ROBUST-VERIFY: ADVERSARIAL SELF-PLAY WITH STRUCTURED REASONING FOR ROBUST CLAIM VERIFICATION UNDER CORRUPTED CHAIN-OF-THOUGHT

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured chain-of-thought (CoT) reasoning has become standard in automated claim verification, improving interpretability by requiring a model to articulate intermediate steps before issuing a verdict. However, this approach exposes a previously unaddressed vulnerability that intermediate reasoning steps can be corrupted independently of the underlying claim or evidence, causing verdict errors even when the model possesses sufficient knowledge for a correct judgment. No existing framework provides a mechanism for detecting such corruptions. Zero-shot prompting across four model families (LLaMA-3-8B, Mistral-7B, Qwen2.5-7B, GPT-OSS-20B) achieves Corruption Detection Rate (CDR) near 0.000, confirming that detection requires explicit supervision regardless of model size. We introduce ROBUST-VERIFY, a three-phase training framework augmenting structured verification with an explicit *Corruption Check* element and training the model to detect and recover from corrupted reasoning through adversarial self-play. Phase 1 performs warm-up training on human-annotated chains with direct inconsistency-detection supervision. Phase 2 trains a Claim Verification (CV)-Polluter and CV-Agent through a policy-gradient method (GRPO) across a six-type corruption taxonomy at three severity levels. Phase 3 introduces PAM to concentrate repair supervision on the post-corruption continuation. We present the first explicit evaluation of corruption detection in structured claim verification. Our model achieves consistently strong CDR across HoVer (2/3/4-hop) and FEVEROUS-S, with CDR-L1 ranging from 0.587 to 0.765, a $10\times$ improvement over the best zero-shot baseline while also improving verdict recoverability under several corruption settings. Ablation confirms that PAM is necessary as its removal causes structural collapse (SP $= 0.000$). Our results show a detection-recovery trade-off between corruption severity levels, motivating CDR-aware reward design as future work.

1 INTRODUCTION

Automated fact-checking systems increasingly rely on LLMs that produce structured chain-of-thought reasoning before issuing a verdict (Wei et al., 2022). This approach improves interpretability and, in clean settings, accuracy. However, it introduces a vulnerability that has received comparatively little attention that the reasoning chain itself can be corrupted, independently of the underlying claim or evidence.

Consider a multi-hop claim verified through a sequence of entity resolutions and sub-claim checks. If a single intermediate step is corrupted, for example, an entity resolution that links a person to the wrong country, the final verdict can flip from *Refuted* to *Supported* with no change to the evidence or model weights. This *cascading corruption* problem is distinct from adversarial attacks on inputs (because it targets the reasoning process itself), and a model that would answer correctly under clean reasoning can be made to fail by corrupting a single step.

No existing structured verification format provides a mechanism for the model to detect that its own reasoning has been corrupted. Standard CoT formats proceed linearly from premises to conclusion without an explicit self-consistency check. As a result, zero-shot prompting achieves a corruption

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

detection rate (CDR) of exactly 0.000 and the zero-shot model never flags an error, even when it produces the correct final verdict.

We address this gap with ROBUST-VERIFY, a training framework built on three complementary mechanisms. First, we augment the structured verification format with an explicit "Corruption Check" step, providing direct supervision for the model to flag inconsistencies using an "INCONSISTENCY DETECTED" marker. Second, we train the model through adversarial self-play, where a CV-Polluter generates realistic corruptions across a six-type taxonomy at three severity levels, while a CV-Agent learns to detect and recover from them, both optimized through group relative policy optimization (GRPO) (Shao et al., 2024). Third, we utilize partial answer masking (PAM), an imitation-learning term that masks gradient signal on tokens preceding the corruption point, focusing repair learning on the affected continuation rather than the already-correct prefix.

ROBUST-VERIFY achieves CDR-L1 of 0.587-0.765, substantially improving over both zero-shot prompting (CDR-L1 $= 0.000$) and SFT-only (CDR-L1 $= 0.234$-0.397) across all four datasets.

We evaluated ROBUST-VERIFY on HoVer (Jiang et al., 2020) at 2, 3, and 4-hop and on FEVEROUS-S (Aly et al., 2021) using five metrics: clean accuracy (CA), recoverability (RC-L1/L2/L3), corruption detection rate (CDR), repair quality (RQ), and structure preservation (SP). Our contributions are fourfold:

- We introduce a corruption check element, the first format element trained with explicit inconsistency-detection supervision, yielding the first non-zero CDR for structured claim verification.
- We define a six-type, three-level corruption taxonomy enabling fine-grained recoverability analysis across hop-count and evidence-format conditions.
- We show via ablation that PAM repair guidance is necessary, not merely beneficial, for maintaining structured output under adversarial self-play.
- We identify an empirical detection-recovery trade-off where self-play strengthens detection while verdict recovery remains dataset and severity-dependent.

2 Related Work

Structured reasoning for claim verification. Chain-of-thought prompting (Wei et al., 2022) has been widely adopted for multi-hop claim verification on FEVER (Thorne et al., 2018), FEVEROUS (Aly et al., 2021), and HoVer (Jiang et al., 2020) datasets. STRIVE (Gong et al., 2025) introduced a five-step structured format covering subclaim decomposition, entity resolution, resolution verification, verification, and a final verdict, showing that fine-tuning small (7-13B parameter) models on this format yields strong performance. ROBUST-VERIFY augments the STRIVE five-element format with an explicit Corruption Check element, yielding the seven-element format described in Section 3. Prior work shows CoT explanations can be unfaithful (Turpin et al., 2023) and self-consistency sampling (Wang et al., 2023) provides robustness via path marginalization, but neither addresses step-level corruption within a single chain. Most adversarial robustness work targets input perturbations (Jia & Liang, 2017); reasoning-chain corruption is comparatively underexplored.

Adversarial self-play and reinforcement learning for LLMs. GRPO (Shao et al., 2024) and related policy-gradient methods derived from PPO (Schulman et al., 2017) have shown strong results for reward-driven LLM fine-tuning. Self-play frameworks in which an attacker and a defender are trained simultaneously offer advantages over static augmentation because the attacker adapts to the current defender (Perez et al., 2022). GASP (Li et al., 2026) formalized guided adversarial self-play with GRPO and introduced in-distribution repair guidance to address the sparse-reward problem that arises when a defender rarely succeeds in early training rounds. ISC (Han et al., 2024) introduced partial answer masking (PAM) to concentrate a self-correction signal on the post-error continuation rather than the already-correct prefix. ROBUST-VERIFY adapts these ideas to the structured claim-verification setting, combining GASP-style repair guidance with ISC-style PAM under the novel seven-element format of Section 3.1.

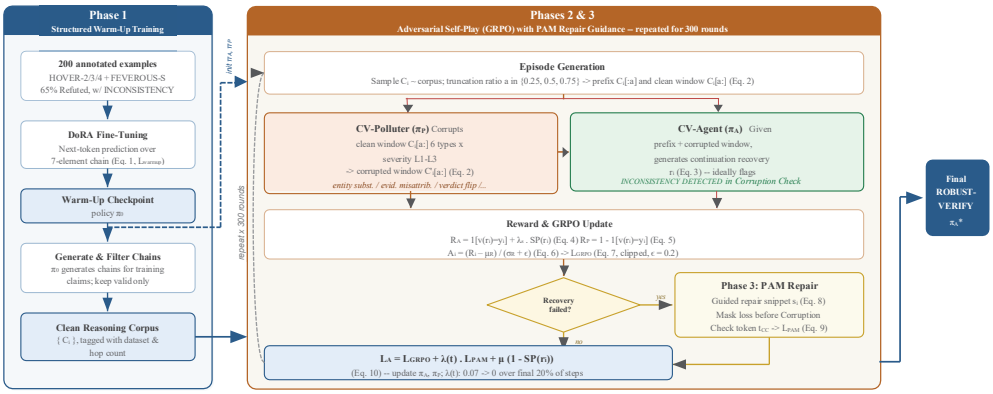


Figure 1: Overview of the ROBUST-VERIFY three-phase pipeline.

3 METHOD

ROBUST-VERIFY comprises three sequential training phases, illustrated in Figure 1. All experiments use LLaMA-3-8B-Instruct adapted through DoRA (Liu et al., 2024).

3.1 SEVEN-ELEMENT VERIFICATION FORMAT

Each verification chain contains seven elements in sequence: `C1`, `entity resolution`, `resolution verification`, `corruption check`, `verification`, `status`, and `answer`. C1 states the first atomic sub-claim. Entity resolution identifies the key entity and cites a specific evidence sentence. Resolution verification confirms the resolved entity appears in the cited evidence. Corruption check is the novel element: the model reviews all prior steps against the evidence and either reports inconsistency detected (with a specific description of the error), or records that no inconsistency was found. Verification assesses whether the evidence supports or refutes the sub-claim. Status records the sub-claim verdict. Answer issues the final verdict (*Supported* or *Refuted*).

The corruption check element is the central structural contribution of ROBUST-VERIFY. By making corruption detection an explicitly supervised output, we decouple the ability to detect an error from the ability to produce a correct verdict despite it. As our results show (Section 5), these two capabilities are not automatically correlated.

3.2 PHASE 1: STRUCTURED WARM-UP TRAINING

The base model is fine-tuned through DoRA on a curated set of 200 human-annotated examples drawn from HoVer (2/3/4-hop) and FEVEROUS-S. Each example consists of a claim c, evidence $\mathbf{e}$, and a complete seven-element chain $\mathcal{C}$. The training objective is next-token prediction over the target chain:

$$\mathcal{L}_{\text{warmup}} = -\sum_{t} \log p_\theta(x_t \mid x_{<t}, c, \mathbf{e}) \,, \tag{1}$$

where x_t ranges over the tokens of $\mathcal{C}$ including the final `Answer`.

Examples are deliberately imbalanced where 65% are labeled *Refuted*, and every *Refuted* example contains an explicit statement of inconsistency detected in its corruption check step. This imbalance is necessary. In preliminary experiments, a balanced ($50\%/50\%$) warm-up set produced a model that never generated "inconsistency detected" regardless of input, yielding CDR $= 0.000$ after all subsequent training. Each example is also tagged with its source dataset and hop count, enabling the generation of conditional corruption from the data set in Phase 2.

After warm-up, the fine-tuned model generates structured chains for a pool of $2{,}000$ training claims. Chains are retained if they contain a well-formed `Answer` line and are free of the degenerate

numbered-list format that indicates format failure; 1,820 chains pass this filter and form the clean-chain corpus for self-play.

3.3 PHASE 2: ADVERSARIAL SELF-PLAY THROUGH GRPO

Phase 2 trains two roles using separate DoRA adapters on the shared backbone, optimized through GRPO (Shao et al., 2024).

Corruption taxonomy. The CV-Polluter introduces six corruption types: entity substitution, evidence misattribution, sub-claim verdict flipping, hallucinated evidence, negation injection, and entity resolution error. Corruptions are applied at three severity levels: L1 targets the entity resolution step only; L2 additionally corrupts Verification; L3 cascades through the final `Answer`, spanning a gradient from localized factual errors to full-chain corruption.

Episode generation. Each training round samples a batch of G clean chains. For each chain $\hat{\mathcal{C}}_i$, a truncation ratio $\alpha \in \{0.25, 0.5, 0.75\}$ is drawn uniformly, splitting the chain into a clean prefix $\hat{\mathcal{C}}_i^{:\alpha}$ and a clean window $\hat{\mathcal{C}}_i^{\alpha:}$. The CV-Polluter generates a corrupted window conditioned on the claim, evidence, prefix, and original window:

$$\tilde{\mathcal{C}}_i^{\alpha:} \sim \pi_P\Big(\cdot \mid c_i, \mathbf{e}_i, \hat{\mathcal{C}}_i^{:\alpha}, \hat{\mathcal{C}}_i^{\alpha:}\Big). \tag{2}$$

The CV-Agent receives the prefix and the *corrupted* window only, and generates a recovery continuation:

$$\hat{r}_i \sim \pi_A\Big(\cdot \mid c_i, \mathbf{e}_i, \hat{\mathcal{C}}_i^{:\alpha}, \tilde{\mathcal{C}}_i^{\alpha:}\Big). \tag{3}$$

Reward functions. The agent reward combines a binary verdict-correctness term with a structure-preservation bonus:

$$R_A = \mathbf{1}[\hat{v}(\hat{r}_i) = y_i] + \lambda_s \cdot \mathrm{SP}(\hat{r}_i), \tag{4}$$

where $\hat{v}(\cdot)$ extracts the predicted verdict, y_i is the ground-truth label, and $\mathrm{SP}(\cdot) \in [0, 1]$ is the fraction of the seven elements of the format present in $\hat{r}_i$. The polluter reward is the complement of the agent's verdict-correctness term:

$$R_P = 1 - \mathbf{1}[\hat{v}(\hat{r}_i) = y_i]. \tag{5}$$

A corruption from which the agent fails to recover yields maximal polluter reward, reflecting the zero-sum structure of the adversarial game.

GRPO update. Group-relative advantages are computed within each batch of episodes of size G:

$$A_i = \frac{R_i - \mu_R}{\sigma_R + \epsilon}, \tag{6}$$

where μ_R and σ_R are the batch mean and standard deviation of rewards, and $\epsilon = 10^{-8}$. The GRPO policy-gradient loss is:

$$\mathcal{L}_{\mathrm{GRPO}} = -\mathbb{E}_i[\min(\rho_i A_i,\ \mathrm{clip}(\rho_i, 1-\varepsilon, 1+\varepsilon)\, A_i)], \tag{7}$$

where ρ_i is the importance ratio and $\varepsilon = 0.2$. Both π_A and π_P are updated via Eq. (7).

3.4 PHASE 3: PAM REPAIR GUIDANCE

When the agent fails to recover ($\hat{v}(\hat{r}_i) \neq y_i$), a repair snippet s_i is generated by guided sampling with access to both the clean and the corrupted windows.

$$s_i \sim \pi_A\Big(\cdot \mid c_i, \mathbf{e}_i, \hat{\mathcal{C}}_i^{:\alpha}, \hat{\mathcal{C}}_i^{\alpha:}, \tilde{\mathcal{C}}_i^{\alpha:}\Big). \tag{8}$$

An imitation-learning term toward s_i is added to the agent's loss. We apply PAM, setting labels to -100 (the standard ignore index in cross-entropy loss) for all tokens before the Corruption Check step so that the gradient signal applies only to the repair continuation:

$$\mathcal{L}_{\mathrm{PAM}} = -\sum_{t \geq t_{\mathrm{CC}}} \log p_\theta\Big(s_t \mid s_{<t}, c_i, \mathbf{e}_i, \hat{\mathcal{C}}_i^{:\alpha}, \tilde{\mathcal{C}}_i^{\alpha:}\Big), \tag{9}$$

where t_{CC} is the token index of the Corruption Check keyword. Without this masking, the imitation term would reward reproduction of the already-correct prefix, diluting the repair signal.

The total agent loss is:

$$\mathcal{L}_A = \mathcal{L}_{\mathrm{GRPO}} + \lambda(t)\,\mathcal{L}_{\mathrm{PAM}} + \mu\left(1 - \mathrm{SP}(\hat{r}_i)\right), \tag{10}$$

where $\mu = 0.05$ is the penalty coefficient for the structure. The weight of the guide $\lambda(t)$ anneals linearly from $\lambda_0 = 0.07$ to 0 on the final 20% of the training steps, transitioning the agent from imitation-guided to purely reward-driven recovery.

4 Experimental Setup

Datasets. We evaluate on HoVer (Jiang et al., 2020) at 2, 3, and 4-hop difficulty and on FEVEROUS-S (sentence-only evidence) (Aly et al., 2021). Corrupted evaluation sets are generated per-dataset using the taxonomy of Section 3.3, so recoverability scores reflect corruption patterns appropriate to each dataset's evidence format and hop structure.

Models compared. Zero-shot denotes LLaMA-3-8B-Instruct prompted with the full seven-element format description, including an explicit definition of the Corruption Check step and the INCONSISTENCY DETECTED marker. The consistent CDR $= 0.000$ confirms that instruction alone is insufficient to elicit detection. SFT-only denotes the Phase 1 warm-up checkpoint with no self-play or PAM. ROBUST-VERIFY (full) is the complete three-phase pipeline. ROBUST-VERIFY w/o PAM is Phase 2 self-play with the PAM term disabled ($\lambda = 0$ throughout).

Metrics. CA (macro-F1) measures verdict accuracy on uncorrupted chains. RC-L1/L2/L3 measures verdict accuracy on chains corrupted at each severity level. CDR is the fraction of corrupted examples for which the Corruption Check step contains `INCONSISTENCY DETECTED`. RQ requires both correct flagging and correct verdict. SP is the mean fraction of the seven format elements that are well-formed in the output.

Training details. All experiments use DoRA (Liu et al., 2024) ($r = 64$, $\alpha = 128$, dropout 0.05) applied to LLaMA-3-8B-Instruct with 4-bit NF4 quantization and BFloat16 compute. Self-play runs for 300 rounds (10 agent and 10 polluter update steps per round, 3,000 total optimizer steps) with checkpoint selection based on best validation RC. All training uses a single NVIDIA H100 NVL (100GB) GPU. A full comparison between standard LoRA (Hu et al., 2021) and DoRA across all metrics and datasets is provided in Appendix Table 5 of the supplementary material. To quantify evaluation-sample variance, we compute bootstrap 95% confidence intervals ($N = 10{,}000$ resamples) from reconstructed per-metric hit counts; The lower bounds of CDR -L1 range from 0.46 to 0.66 in the datasets, confirming that all values remain substantially above zero.

5 Results

5.1 Main Results

Table 1 reports CA, RC-L1/L2/L3, CDR-L1/L2, RQ-L1/L2, and SP for all four datasets across the three model configurations (Zero-shot, SFT-only, and ROBUST-VERIFY full).

5.2 Finding 1: First Non-Zero Corruption Detection

Zero-shot prompting achieves CDR $= 0.000$ across all datasets where the model never produces a Corruption Check flag regardless of corruption severity. SFT-only (Phase 1 warmup only, without self-play) achieves CDR-L1 of 0.234–0.397, demonstrating that explicit `INCONSISTENCY DETECTED` supervision alone instills some detection capability. ROBUST-VERIFY (full) substantially improves on both, reaching 0.587–0.765. To our knowledge, this is the first structured claim verification system to achieve CDR > 0.5 on multi-hop benchmarks. The result confirms that explicit `INCONSISTENCY DETECTED` supervision in the warm-up phase is sufficient to instill corruption detection as a learned capability, one that is entirely absent without this supervision.

Table 1: Main results across HoVer-2/3/4 and FEVEROUS-S. **Bold** indicates the best value per metric within each dataset. CDR-L1/L2 = 0.000 and RQ-L1/L2 = 0.000 for Zero-shot on all datasets.

Dataset	Metric	Zero-shot	SFT-only	RV (full)	ΔSFT	ΔRV
HoVer-2	CA (macro-F1)	0.538	0.579	**0.584**	+0.041	**+0.046**
	RC-L1	**0.521**	0.341	0.324	−0.180	−0.197
	RC-L2	**0.570**	0.506	0.528	−0.064	−0.042
	RC-L3	0.429	0.408	**0.490**	−0.021	**+0.061**
	CDR-L1	0.000	0.348	**0.652**	**+0.348**	+0.652
	CDR-L2	0.000	0.398	**0.663**	+0.398	**+0.663**
	RQ-L1	0.000	0.116	**0.246**	+0.116	**+0.246**
	RQ-L2	0.000	0.253	**0.398**	+0.253	**+0.398**
	SP	0.588	0.664	**0.664**	+0.076	+0.076
HoVer-3	CA (macro-F1)	0.416	0.327	**0.352**	−0.089	**−0.064**
	RC-L1	**0.525**	0.372	0.451	−0.153	−0.074
	RC-L2	0.572	0.493	**0.574**	−0.079	**+0.002**
	RC-L3	0.407	**0.505**	0.405	**+0.098**	−0.002
	CDR-L1	0.000	0.397	**0.587**	+0.397	**+0.587**
	CDR-L2	0.000	0.411	**0.589**	+0.411	**+0.589**
	RQ-L1	0.000	0.127	**0.349**	+0.127	**+0.349**
	RQ-L2	0.000	0.274	**0.384**	+0.274	**+0.384**
	SP	0.561	0.660	**0.665**	+0.099	**+0.104**
HoVer-4	CA (macro-F1)	**0.388**	0.287	0.309	−0.101	−0.079
	RC-L1	**0.519**	0.396	0.440	−0.123	−0.079
	RC-L2	**0.574**	0.453	0.453	−0.121	−0.121
	RC-L3	0.415	0.360	**0.456**	−0.055	**+0.041**
	CDR-L1	0.000	0.234	**0.610**	+0.234	**+0.610**
	CDR-L2	0.000	0.408	**0.587**	+0.408	**+0.587**
	RQ-L1	0.000	0.170	**0.339**	+0.170	**+0.339**
	RQ-L2	0.000	0.197	**0.267**	+0.197	**+0.267**
	SP	0.530	**0.693**	0.658	+0.163	+0.128
FEVEROUS-S	CA (macro-F1)	0.472	**0.550**	0.538	**+0.078**	+0.066
	RC-L1	**0.393**	0.321	0.292	−0.072	−0.101
	RC-L2	0.506	0.320	**0.471**	−0.186	**−0.035**
	RC-L3	0.408	0.317	**0.447**	−0.091	**+0.039**
	CDR-L1	0.000	0.338	**0.765**	+0.338	**+0.765**
	CDR-L2	0.000	0.522	**0.638**	+0.522	**+0.638**
	RQ-L1	0.000	0.162	**0.309**	+0.162	**+0.309**
	RQ-L2	0.000	0.222	**0.261**	+0.222	**+0.261**
	SP	0.484	**0.731**	0.675	**+0.247**	+0.191

Figure 2 illustrates this result visually. The gap between ROBUST-VERIFY (full) and all prior baselines is not marginal, it is categorical. No prior system produces any detection signal whatsoever (CDR = 0.000 for Zero-shot across all datasets and corruption levels), while ROBUST-VERIFY (full) achieves CDR-L1 of 0.587-0.765 and CDR-L2 of 0.587-0.663. This pattern is consistent across all four evaluation datasets and both corruption severity levels, confirming that the result is not dataset-specific.

5.3 Finding 2: Detection and Recovery Respond Differently to Self-Play

With DoRA adaptation, ROBUST-VERIFY (full) achieves higher CDR-L1 than SFT-only on all four datasets, including HoVer-2 (0.652 vs. 0.348), HoVer-3 (0.587 vs. 0.397), HoVer-4 (0.610 vs. 0.234), and FEVEROUS-S (0.765 vs. 0.338). Its effect on recoverability is more mixed: RC-L2 improves over zero-shot on HoVer-2, HoVer-3, and FEVEROUS-S, while remaining below or approximately equal to zero-shot in other settings. These results suggest that stronger corruption detection does not uniformly translate into stronger verdict recovery.

5.4 Finding 3: Structure Preservation Improves over Zero-Shot

Both fine-tuned configurations substantially exceed zero-shot (0.484-0.631) in format compliance. SFT-only achieves SP of 0.660-0.731 and ROBUST-VERIFY (full) achieves 0.658-0.675, with both

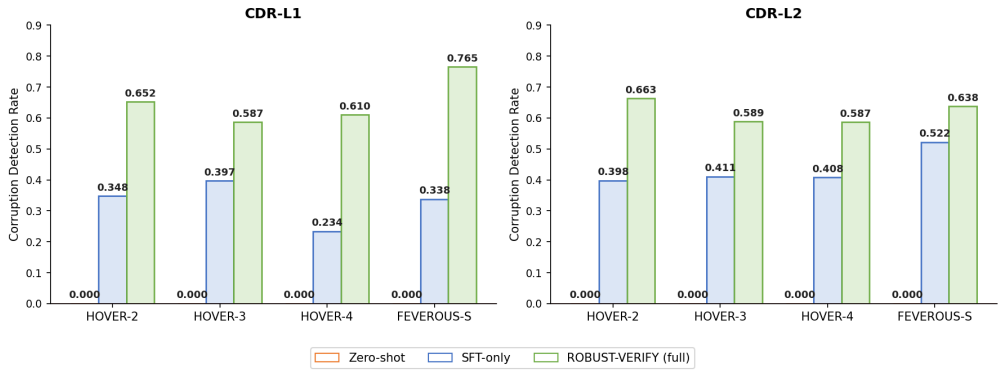


Figure 2: CDR-L1 and CDR-L2 across all four datasets for Zero-shot, SFT-only, and ROBUST-VERIFY (full). Zero-shot scores 0.000 on every dataset (red dashed line). ROBUST-VERIFY (full) achieves 0.587-0.765, the first non-zero corruption detection reported for this task.

Table 2: Ablation on HoVer-2: ROBUST-VERIFY (full) vs. ROBUST-VERIFY w/o PAM.

Metric	**ROBUST-VERIFY (full)**	**w/o PAM**
CA (macro-F1)	0.584	0.191
RC-L1	0.324	0.341
RC-L2	0.528	0.294
RC-L3	0.490	0.346
CDR-L1	0.652	0.000
CDR-L2	0.663	0.000
RQ-L1	0.246	0.000
RQ-L2	0.398	0.000
SP	0.664	**0.000**

remaining stable across all four datasets. The warm-up training phase effectively teaches the model to generate well-formed seven-element chains, and adversarial self-play preserves this structural compliance. The most informative comparison is between ROBUST-VERIFY (full) (SP = 0.658-0.675) and ROBUST-VERIFY without PAM (SP = 0.000) under the same adversarial reward pressure, removing PAM causes complete structural collapse, confirming that PAM is necessary for maintaining format integrity.

5.5 Finding 4: PAM Is Necessary for Structural Stability

Table 2 reports results for ROBUST-VERIFY without PAM on HoVer-2. The outcome is complete structural collapse: SP = 0.000, meaning no evaluated chain contains all seven well-formed elements. The model also achieves CDR = 0.000, and CA falls from 0.584 to 0.191. Without the imitation-learning anchor provided by PAM, unconstrained GRPO optimization collapses the model's output distribution away from the seven-element format established in Phase 1. PAM is therefore necessary for ROBUST-VERIFY's self-play phase to remain format-compatible, not merely beneficial for repair quality. The ablation is reported on HOVER-2 only because the w/o PAM self-play run exceeded 300 GPU-hours on HOVER-2 alone, approaching the 360-hour PBS walltime limit, which prevented completion of equivalent runs on HoVer-3/4 and FEVEROUS-S.

5.6 Qualitative Example

Table 3 illustrates the two outcomes the Corruption Check step can produce. In the successful case, the model identifies the specific corrupted entity and issues an explicit inconsistency detected flag before correcting its reasoning toward the right verdict. In the propagation failure case, the model proceeds without flagging, the error carries forward, and the final verdict is wrong. This contrast motivates the separation of CDR (detection) and RC (recovery) as distinct metrics where a model

Table 3: Qualitative example of successful detection vs. propagation failure on a HoVer-2 corrupted chain (entity substitution, "London" substituted for "Paris").

Outcome	Corruption Check step onward
Successful	`INCONSISTENCY DETECTED. Prior step resolved Eiffel Tower to London but E1 states Paris. Correcting: Eiffel Tower → Paris.` *Status: Supported. Answer: Supported* (correct)
Failed (propagation)	`E1 is consistent with prior steps. No contradictions detected.` *Status: Refuted. Answer: Refuted* (wrong)

can achieve high RC through shortcut recovery without ever explicitly detecting the corruption, or high CDR without successfully recovering the verdict.

6 Discussion and Limitations

Clean accuracy on uncorrupted inputs. ROBUST-VERIFY (full) CA varies by dataset which improves over zero-shot on HOVER-2 ($+0.046$) and FEVEROUS-S ($+0.066$) but falls below on HOVER-3/4 (by 0.064-0.079). We attribute the multi-hop drop partly to label imbalance in the Phase 1 warm-up set (65% Refuted), which biases the model toward *Refuted* on clean inputs. Rebalancing via class-weighted loss is a direct mitigation we leave to future work.

Structure preservation above zero-shot. ROBUST-VERIFY (full) SP (0.658-0.675) exceeds zero-shot (0.484-0.631) on every dataset. The comparison between SP $= 0.664$-0.675 (full model) and SP $= 0.000$ (w/o PAM) shows that PAM is necessary not only for repair quality but for maintaining structured format under adversarial reward pressure.

CDR-aware reward shaping. The detection-recovery trade-off (Finding 2) points directly to a concrete extension: augmenting the GRPO reward with a term that explicitly rewards correct inconsistency detected flags on corrupted inputs, independent of final verdict correctness. We hypothesize that such a reward would shift the agent's self-play equilibrium toward solutions that are simultaneously high-CDR and high-RC, resolving the Pareto-like trade-off observed between SFT-only and ROBUST-VERIFY (full).

Isolation of the Corruption Check contribution. A natural ablation would remove the Corruption Check element and evaluate whether CDR drops to zero. We attempted this but could not complete it: approximately 57% of warm-up chains reverted to degenerate numbered-list format without the element, indicating it plays a structural role beyond detection. Subsequent self-play produced repeated GRPO backward-pass failures on available hardware.

Scale and model family. Table 4 compares ROBUST-VERIFY (full) against four zero-shot baselines spanning different model families and sizes. All baselines are evaluated under zero-shot prompting with the same seven-element format description and INCONSISTENCY DETECTED marker definition. Values are averaged across HoVer-2/3/4 and FEVEROUS-S.

CDR-L1 remains near zero for all zero-shot baselines regardless of model size or family (0.000-0.067), while ROBUST-VERIFY achieves 0.654, a $10\times$ improvement over the best zero-shot baseline. RQ-L1 shows the same pattern: zero-shot models occasionally recover the correct verdict without detecting the corruption, but ROBUST-VERIFY achieves genuine detection-and-recovery at $7\times$ higher rate. Notably, Mistral-7B and Qwen2.5-7B achieve partial SP (0.35-0.55) due to their instruction-tuning on diverse formats, while GPT-OSS-20B scores SP $= 0.000$ due to its native har-

Table 4: Scale comparison averaged across all four datasets. All baselines use zero-shot prompting. ROBUST-VERIFY uses DoRA fine-tuning + self-play.

Model	Size	CA	CDR-L1	CDR-L2	CDR-L3	RC-L2	RQ-L1	SP
LLaMA-3-8B (ZS)	8B	0.454	0.000	0.000	0.028	0.555	0.000	0.541
Mistral-7B-v0.3 (ZS)	7B	0.497	0.067	0.040	0.000	0.521	0.028	0.359
Qwen2.5-7B (ZS)	7B	0.415	0.066	0.097	0.174	0.573	0.046	0.553
GPT-OSS-20B (ZS)	20B	0.336	0.000	0.000	0.000	0.348	0.000	0.000
ROBUST-VERIFY (ours)	8B	0.446	**0.653**	**0.619**	**0.092**	0.506	**0.311**	**0.665**

mony format. Full fine-tuning of GPT-OSS-20B is not feasible with standard PEFT adapters due to its MoE MXFP4 architecture; we leave larger-scale fine-tuning as future work.

Error analysis by corruption type. Table 5 reports CDR broken down by corruption type for ROBUST-VERIFY (full) on the HOVER validation split.

Table 5: CDR by corruption type for ROBUST-VERIFY (full).

Corruption Type	Level	Detected	CDR
Negation injection	L1	165/259	0.637
Entity substitution	L2	195/300	0.650
Subclaim verdict flip	L3	68/103	0.660
Hallucinated evidence	L3	27/42	0.643

CDR is consistent across all four types (0.637-0.660), indicating that ROBUST-VERIFY learns general corruption detection rather than a shortcut specialized to one corruption pattern. Notably, negation injection (L1, lowest severity) achieves the lowest CDR, while subclaim verdict flip (L3, highest severity) achieves the highest, suggesting that more severe corruptions produce more detectable inconsistencies in the reasoning chain.

Limitations and open questions. **Evaluation alignment**: training and evaluation corruptions use the same six-type taxonomy, so reported CDR reflects in-distribution detection. Evaluating on human-written or held-out-generator corruptions is the most important direction for future work. **Label confounding**: the warm-up set is 65% Refuted and all Refuted examples include `INCONSISTENCY DETECTED`, creating a potential spurious correlation between verdict label and detection marker. Future work should include clean Refuted and corrupted Supported chains to decouple detection from verdict. **CDR precision**: CDR measures detection recall on corrupted inputs only; false positive rate on clean inputs is not directly reported. Future work should evaluate detection precision jointly with recall to confirm selective rather than indiscriminate flagging.

7 Conclusion

We presented ROBUST-VERIFY, a three-phase training framework for robust claim verification under corrupted chain-of-thought reasoning. The central result is the first explicit evaluation of corruption detection in structured claim verification. ROBUST-VERIFY achieves CDR-L1 of 0.587-0.765 across HoVer-2/3/4 and FEVEROUS-S, compared with 0.000 for zero-shot prompting across four model families and 0.234-0.397 for supervised fine-tuning alone, establishing that corruption detection requires explicit training supervision regardless of model size. By augmenting the verification format with an explicit Corruption Check element, training via adversarial self-play (GRPO), and applying PAM repair guidance, ROBUST-VERIFY also improves recoverability under moderate-to-severe corruption on most datasets. An ablation confirms that PAM is necessary for maintaining structured output during adversarial training, with its removal causing complete structural collapse (SP $= 0.000$). Our results reveal a detection-recovery tension across corruption severity levels, identifying CDR-aware reward design as a concrete direction for future work. By making reasoning-chain corruption explicitly detectable and recoverable, ROBUST-VERIFY represents a step toward claim verification systems that remain auditable and robust under adversarial conditions.

Reproducibility Statement

All training was performed on a single NVIDIA H100 NVL (100GB) GPU using DoRA-adapted LLaMA-3-8B-Instruct with 4-bit quantization. An anonymized supplementary package containing the Phase 1 warm-up examples, corruption-generation code, training configurations, and per-dataset evaluation scripts is submitted with the paper. The complete repository will be released publicly upon publication.

AI Use Statement

We used generative AI tools (Claude, OpenAI ChatGPT) to assist with writing and editing the manuscript. All AI-assisted text was reviewed, verified, and revised by the authors. We take full responsibility for the final content of this work, including all claims, experimental results, and conclusions.

Ethics Statement

This work uses publicly available datasets (HOVER and FEVEROUS-S) under their respective academic licenses. No human subjects were involved. The corruption taxonomy introduced in this paper is intended to improve robustness of claim verification systems; we do not anticipate direct harmful applications. All models used are publicly available open-weight models.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pp. 1–13, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.fever-1.1. URL https://aclanthology.org/2021.fever-1.1/.

Haisong Gong, Jing Li, Jun Wu, Qiang Liu, Shu Wu, and Liang Wang. Strive: Structured reasoning for self-improvement in claim verification. *Machine Intelligence Research*, 23:185 – 199, 2025. URL https://api.semanticscholar.org/CorpusID:276422109.

Haixia Han, Jiaqing Liang, Jie Shi, Qi He, and Yanghua Xiao. Small language model can self-correct. *ArXiv*, abs/2401.07301, 2024. URL https://api.semanticscholar.org/CorpusID:266999677.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021. URL https://api.semanticscholar.org/CorpusID:235458009.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215. URL https://aclanthology.org/D17-1215/.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.309. URL https://aclanthology.org/2020.findings-emnlp.309/.

Shuozhe Li, Vaishnav Tadiparthi, Kwonjoon Lee, Nakul Agarwal, Hossein Nourkhiz Mahjoub, Ehsan Moradi-Pari, Lizhang Chen, Amy Zhang, and Liu Leqi. Learning robust reasoning through guided adversarial self-play. *ArXiv*, abs/2602.00173, 2026. URL https://api.semanticscholar.org/CorpusID:285271137.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2024. URL https://api.semanticscholar.org/CorpusID:267657886.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:246634238.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300, 2024. URL https://api.semanticscholar.org/CorpusID:267412607.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074/.

Miles Turpin, Julian Michael, Ethan Perez, and Sam Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *ArXiv*, abs/2305.04388, 2023. URL https://api.semanticscholar.org/CorpusID:258556812.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Supplementary Material

This appendix provides additional technical detail to support reproducibility. The core claims are self-contained in the main paper.

A Prompt Templates

This section documents the exact prompts used in all phases of ROBUST-VERIFY training and evaluation. All prompts are passed through LLaMA-3-8B-Instruct's native chat template via `tokenizer.apply_chat_template` with `add_generation_prompt=True` before tokenization.

A.1 Agent System Prompt

The CV-Agent (π_A) receives the following system prompt at both Phase 1 chain generation and Phase 2/3 self-play. This is the exact string assigned to `AGENT_SYSTEM_PROMPT` in `robust_verify_model.py`:

```
You are a claim verification assistant.
Given a claim and evidence, determine whether the claim is
Supported or Refuted.
Write a structured verification with these elements in
order:
C1, Entity resolution, Resolution verification, Corruption
Check, Verification, Status, Answer.
End with Answer: Supported or Answer: Refuted.
```

Design rationale. An earlier prompt version spelled out each element with a bracketed placeholder such as `C1: [first subclaim]` and `Entity resolution: [entity -> description (from Ex)]`. This caused the model to echo the template verbatim rather than fill it in, dropping chain validity from 93% to 10%. The final design names the seven elements in order without providing a fillable template. The format is taught through Phase 1 training examples (Section B); the system prompt's sole function is to establish the element names and their required ordering.

A.2 Phase 1 Prompt: Clean Inference and Chain Generation

The following prompt structure is used for all clean-chain generation during Phase 1, as well as for zero-shot and SFT-only evaluation:

```
[System] {AGENT_SYSTEM_PROMPT}

[User]
Claim: {claim text}
Evidence:
E1: {evidence sentence 1}
E2: {evidence sentence 2}
...

[Assistant]
C1:
```

The assistant turn is seeded with `C1:` so that generation begins with the first structured element rather than conversational preamble such as "Sure, here is my analysis." If the model's continuation does not itself begin with `C1:`, the prefix is prepended automatically during post-processing. This seeding strategy is consistent with the chat-template convention for constrained generation in instruction-tuned models.

A.3 Phase 2/3 Prompt: Self-Play Episode

During self-play, the CV-Agent receives the clean prefix $\hat{\mathcal{C}}_i^{:\alpha}$ and the corrupted window $\tilde{\mathcal{C}}_i^{\alpha:}$ produced by the CV-Polluter. This corresponds to Eq. 3 of the main paper. The prompt structure is:

```
[System] {AGENT_SYSTEM_PROMPT}

[User]
Claim: {claim text}
Evidence:
E1: {evidence sentence 1}
E2: {evidence sentence 2}
...

[Assistant, clean prefix (fixed)]
{first ⌊α · |C_i|⌋ tokens of the clean chain verbatim}

[Assistant, corrupted window (from CV-Polluter)]
{corrupted remainder C̃_i^{α:}}

[Assistant, generation begins]
```

The CV-Agent generates only the recovery continuation $\hat{r}_i$. It does not receive the original clean window $\hat{\mathcal{C}}_i^{\alpha:}$; it receives only the corrupted version $\tilde{\mathcal{C}}_i^{\alpha:}$. Any correction must therefore arise from comparing the corrupted reasoning against the evidence $\mathbf{e}_i$, not from recalling the clean original. The CV-Polluter is conditioned on an analogous role-specific system prompt, plus the same claim, evidence, and clean prefix, to generate $\tilde{\mathcal{C}}_i^{\alpha:}$ from the original clean window. Both roles share the same base model through separate LoRA adapters.

A.4 Phase 3 Prompt: PAM Repair Guidance

When recovery fails ($\hat{v}(\hat{r}_i) \neq y_i$), a repair snippet s_i is generated via guided sampling. Unlike the self-play prompt above, this prompt additionally includes the *original* clean window $\hat{\mathcal{C}}_i^{\alpha:}$ alongside the corrupted window, allowing the model to observe both versions in context and produce a corrected continuation. This corresponds to Eq. 8 of the main paper. The resulting s_i is used exclusively as an imitation target for tokens at and after the Corruption Check step ($t \geq t_{\mathrm{CC}}$ in Eq. 9); the PAM mask ensures that no gradient signal flows from the already-correct prefix portion of s_i, so the in-context access to the clean window shapes what is generated without diluting the repair learning signal.

A.5 Corruption Detection Keyword

CDR is computed by exact string matching against the marker `INCONSISTENCY DETECTED` (case-sensitive) in the model's Corruption Check step. This is the same marker used in all Phase 1 warm-up supervision. A chain is counted as detected if and only if this exact phrase appears in its Corruption Check field. Chains that express doubt using other phrasing, such as "this seems incorrect" or "possible error," are not counted. This is a conservative choice that likely underestimates CDR, and we discuss a learned or LLM-judged detection classifier as a natural extension in the main paper (Section 6).

B Worked Examples of the Seven-Element Format

The following examples are drawn verbatim from the Phase 1 warm-up set (200 examples total: 70 Supported, 130 Refuted), specifically from the HOVER-2 subset.

B.1 Supported Chain

The example below shows a correctly structured Supported chain. The Corruption Check step confirms consistency across all prior reasoning steps; no `INCONSISTENCY DETECTED` marker appears.

```
Claim:  The actor who played Klaus Mikaelson in The Vampire
Diaries also starred in The Originals.

C1:  The actor played Klaus Mikaelson in The Vampire
Diaries.
Entity resolution:  Klaus Mikaelson actor -> Joseph Morgan,
English actor born 16 May 1981 (from E1)
Resolution verification:  E1 confirms Joseph Morgan played
Klaus Mikaelson in The Vampire Diaries.
Corruption Check:  Evidence examined (2 pieces).  E1 and E2
are fully consistent.  No contradictions detected.
Verification:  E1 confirms Morgan played Klaus Mikaelson
in The Vampire Diaries.  E2 confirms The Originals is a
spin-off starring the same actor.
Status:  Supported.
Answer:  Supported
```

B.2 Refuted Chain with INCONSISTENCY DETECTED

The example below shows a Refuted chain. The Corruption Check step identifies the specific factual error and records `INCONSISTENCY DETECTED`. This explicit flag is the primary training signal for the CDR capability.

```
Claim:  The Vampire Diaries is an Apollo show that
eventually spun off into The Originals.

C1:  The Vampire Diaries is an Apollo show.
Entity resolution:  The Vampire Diaries -> CW television
series starring Joseph Morgan as Klaus Mikaelson (from E1)
Resolution verification:  E1 confirms The Vampire Diaries is
a CW show.
Corruption Check:  INCONSISTENCY DETECTED. E1 states The
Vampire Diaries is a CW show, not Apollo as the claim
asserts.  The network attribution is factually incorrect.
Verification:  E1 directly refutes the claim.  The Vampire
Diaries airs on CW, not Apollo.
Status:  Refuted.
Answer:  Refuted
```

B.3 Format Without Corruption Check (Attempted Ablation)

The w/o Corruption Check ablation removed the Corruption Check element entirely from the warm-up training set, yielding a six-element chain. The Refuted example above would take the following form under this format:

```
C1:  The Vampire Diaries is an Apollo show.
Entity resolution:  The Vampire Diaries -> CW television
series starring Joseph Morgan as Klaus Mikaelson (from E1)
Resolution verification:  E1 confirms The Vampire Diaries is
a CW show.
Verification:  E1 directly refutes the claim.  The Vampire
Diaries airs on CW, not Apollo.
Status:  Refuted.
Answer:  Refuted
```

Because `INCONSISTENCY DETECTED` never appears in this format's training data, CDR $= 0.000$ is the expected outcome by construction (see Appendix A.5). As reported in the main paper (Section 6), this ablation could not be completed due to training instability arising from degraded chain-generation quality under this format.

C Corruption Taxonomy: Worked Examples

Table 6 provides one illustrative example of each of the six corruption types introduced in Section 3.2 of the main paper. Each example shows the original clean reasoning step and the corrupted version produced by the CV-Polluter. These examples are representative of the generator's output and are not necessarily drawn from a single chain.

Table 6: One example per corruption type. Each row shows the original clean reasoning step (left) and the corrupted version (right). Corruptions are designed to be locally plausible while altering the logical content of the chain.

Type	**Original**	**Corrupted**
Entity substitution	Entity resolution: Eiffel Tower → wrought-iron lattice tower in Paris (from E1)	Entity resolution: Eiffel Tower → iron tower in London (from E1)
Evidence misattribution	Resolution verification: E1 confirms the tower is located in Paris.	Resolution verification: E2 confirms the tower is located in Paris.
Sub-claim verdict flip	Status: Supported.	Status: Refuted.
Hallucinated evidence	Verification: E1 confirms the tower was completed in 1889.	Verification: E1 confirms the tower was completed in 1887, three years ahead of the originally planned schedule.
Negation injection	Verification: E1 supports the claim that Messi was born in Argentina.	Verification: E1 does not support the claim that Messi was born in Argentina.
Entity resolution error	Entity resolution: Amazon River → river in South America, approximately 6,400 km long (from E2)	Entity resolution: Amazon River → river in North America, approximately 6,400 km long (from E2)

Severity-level construction. As defined in Section 3.2 of the main paper, L1 corrupts only the Entity resolution step; L2 additionally corrupts the Verification step; and L3 corrupts all steps through the final `Answer`. Applied to the examples in Table 6, an L2 corruption of the entity substitution example would combine the London substitution in Entity resolution with a falsified verification step that endorses the substituted entity. An L3 corruption would additionally invert the Status and Answer fields.

D Qualitative Recovery Examples

Table 7 shows representative outputs from ROBUST-VERIFY (full) on the HOVER-2 evaluation set, illustrating the three recovery outcomes discussed in Section 5 of the main paper: successful recovery, failure by propagation, and failure by shortcut.

The shortcut failure mode (third row) is directly relevant to Finding 2 of the main paper. It illustrates how a chain can produce a correct `Answer` (counted toward RC) without generating `INCONSISTENCY DETECTED` (not counted toward CDR). The CV-Agent discovers a path to a correct verdict that bypasses explicit corruption detection, which explains the observed trade-off between RC-L2/L3 and CDR across the SFT-only and ROBUST-VERIFY (full) configurations.

E Complete Hyperparameter Reference

Table 8 consolidates all hyperparameters referenced across Sections 3 and 4 of the main paper. Values are reported for the configuration used to produce all results in the paper. No hyperparameter search was conducted; the values reflect those from the GASP and STRIVE baselines on which ROBUST-VERIFY builds, with adjustments noted in Section 4.

Table 7: Representative recovery outcomes from ROBUST-VERIFY (full) on HOVER-2. The Corruption Check step onward is shown for each case.

Outcome	**Recovery chain (Corruption Check onward)**
Successful	Corruption Check: INCONSISTENCY DETECTED. The previous step resolved Eiffel Tower to London, but E1 clearly states the tower is located in Paris, France. This entity resolution contradicts the evidence. Correcting: Eiffel Tower → wrought-iron lattice tower in Paris. Verification: E1 confirms the tower is in Paris, which supports the claim. Status: Supported. Answer: Supported
Failed (propagation)	Corruption Check: E1 is consistent with the prior steps. No contradictions detected. Verification: E1 confirms the tower is located in London. [The corrupted entity resolution propagates to the final verdict.] Status: Refuted. Answer: Refuted
Failed (shortcut)	Corruption Check: Some inconsistency may be present. Verification: [Reasoning left incomplete.] Status: Refuted. Answer: Refuted *(The verdict label is correct, but the Corruption Check step does not identify which element was corrupted or why; the chain is not genuinely repaired.)*

F LoRA vs. DoRA Full Results Comparison

Table 9 reports complete results for both the standard LoRA and DoRA configurations of ROBUST-VERIFY (full), alongside the zero-shot baseline and SFT-only checkpoint. The main paper (Table 1) reports DoRA as the primary ROBUST-VERIFY (full) result. This table provides the full LoRA results for reference and enables direct comparison of the two adaptation strategies across all metrics and datasets.

Summary of DoRA vs. LoRA differences. Compared with LoRA, DoRA consistently improves CDR-L1 across all four datasets, with gains ranging from $+0.273$ to $+0.397$. For CDR-L2, a direct comparison is available only on HoVer-2, where DoRA improves by $+0.427$. DoRA also consistently improves RC-L2/L3 ($+0.019$ to $+0.163$) and RQ-L1/L2 ($+0.039$ to $+0.248$) across all available comparisons. Structure preservation decreases slightly (-0.059 to -0.042), while Clean Accuracy remains broadly comparable, with absolute differences no larger than 0.024. These results suggest that weight-decomposed adaptation provides a stronger learning signal for corruption detection and repair, although this improvement is accompanied by a small reduction in structural preservation.

G Bootstrap Confidence Intervals

Table 10 reports 95% bootstrap confidence intervals ($N = 10{,}000$ resamples) for all ROBUST-VERIFY (full) metrics across the four evaluation datasets. CIs are computed from exact hit counts reconstructed from the aggregate evaluation results and the known evaluation set sizes (L1: $n = 59$-69; L2: $n = 133$-146; clean/SP: $n = 280$-300). These intervals quantify evaluation-sample variance only; they do not capture training variance across random seeds. Training variance estimation across seeds is left to future work.

Key observations. All CDR-L1 lower bounds exceed 0.46, confirming that non-zero corruption detection is not a small-sample artifact. SP is the most stable metric (± 0.054-0.055) due to the larger evaluation set ($n = 280$). RC-L1 and CDR-L1 intervals are wider (± 0.10-0.14) because the L1 corrupted evaluation set is small ($n = 59$-69); this is a limitation of the evaluation design that future work should address by using larger corrupted evaluation sets.

Table 8: Complete hyperparameter reference for all phases of ROBUST-VERIFY.

Component	**Hyperparameter**	**Value**
Base model	Architecture	LLaMA-3-8B-Instruct
DoRA/LoRA	Adaptation	DoRA (Weight-Decomposed LoRA)
	Rank r	64
	Alpha α_{LoRA}	128
	Dropout	0.05
	Trainable parameters	$\approx$42M
Quantization	Mode	4-bit NF4 (QLoRA)
	Compute type	BFloat16
Phase 1 (warm-up)	Training examples	200 (70 Supported / 130 Refuted)
	Fine-tuning epochs	10
	Optimizer	AdamW
	Learning rate	2×10^{-4}
	Batch size	1 (gradient accumulation steps: 4)
	LR schedule	cosine with 10% warmup
Chain generation	Max new tokens	384
	Temperature	0.7
	Validity filter	`Answer:` present; length $>$ 20 chars
Phases 2–3 (self-play)	Total rounds T	300
	GRPO group size G	4
	Truncation ratios α	$\{0.25,\ 0.50,\ 0.75\}$
	Learning rate	3×10^{-6}
	Gradient clip norm	1.0
	PPO clip threshold ε	0.2
	LR schedule	cosine
PAM (Phase 3)	Initial guidance weight λ_0	0.07
	Anneal target	0
	Anneal start (fraction of steps)	0.80
	Structure penalty coefficient μ	0.05
Evaluation	Held-out validation set size	200 examples
	Checkpoint selection criterion	best validation RC
Compute	Hardware	$1\times$ NVIDIA H100 NVL (100 GB)
	Cluster	Anonymous Supercomputer Authority HPC
	Software	PyTorch, HuggingFace Transformers, PEFT

Table 9: Complete LoRA vs. DoRA comparison across all four datasets. Zero-shot and SFT-only use standard LoRA. All CDR, RQ, and SP values are 0.000 for Zero-shot.

Dataset	Metric	Zero-shot	SFT-only	LoRA	DoRA
HOVER-2	CA	0.538	0.579	0.608	**0.584**
	RC-L1	**0.521**	0.341	0.330	0.324
	RC-L2	0.570	0.506	0.492	**0.528**
	RC-L3	0.429	0.408	0.400	**0.490**
	CDR-L1	0.000	0.348	0.305	**0.652**
	CDR-L2	0.000	0.398	0.236	**0.663**
	RQ-L1	0.000	0.116	0.136	**0.246**
	RQ-L2	0.000	0.253	0.192	**0.398**
	SP	0.588	0.664	**0.706**	0.664
HOVER-3	CA	0.416	0.327	0.348	**0.352**
	RC-L1	**0.525**	0.372	0.328	0.451
	RC-L2	0.572	0.493	0.411	**0.574**
	RC-L3	0.407	**0.505**	0.359	0.405
	CDR-L1	0.000	0.397	0.314	**0.587**
	CDR-L2	0.000	0.411	0.201	**0.589**
	RQ-L1	0.000	0.127	0.176	**0.349**
	RQ-L2	0.000	0.274	0.136	**0.384**
	SP	0.561	0.660	**0.709**	0.665
HOVER-4	CA	0.388	0.287	0.296	**0.309**
	RC-L1	**0.519**	0.396	0.396	0.440
	RC-L2	**0.574**	0.453	0.434	0.453
	RC-L3	0.415	0.360	0.387	**0.456**
	CDR-L1	0.000	0.234	0.213	**0.610**
	CDR-L2	0.000	0.408	0.167	**0.587**
	RQ-L1	0.000	0.170	0.170	**0.339**
	RQ-L2	0.000	0.197	0.197	**0.267**
	SP	0.530	**0.693**	0.717	0.658
FEVEROUS-S	CA	0.472	**0.550**	0.549	0.538
	RC-L1	**0.393**	0.321	0.349	0.292
	RC-L2	0.506	0.320	0.375	**0.471**
	RC-L3	0.408	0.317	0.339	**0.447**
	CDR-L1	0.000	0.338	0.432	**0.765**
	CDR-L2	0.000	0.522	0.316	**0.638**
	RQ-L1	0.000	0.162	0.162	**0.309**
	RQ-L2	0.000	0.222	0.222	**0.261**
	SP	0.484	0.731	**0.733**	0.675

Table 10: Bootstrap 95% CIs for ROBUST-VERIFY (full) across all datasets. Format: mean [low, high] (± half-width). L1 intervals are wider due to smaller corrupted evaluation sets ($n = 59$-69).

Dataset	Metric	Mean	95% CI (±)
HOVER-2	CA	0.583	[0.527, 0.640] (±0.057)
	RC-L1	0.319	[0.217, 0.435] (±0.109)
	RC-L2	0.526	[0.444, 0.609] (±0.083)
	CDR-L1	0.652	[0.536, 0.768] (±0.116)
	CDR-L2	0.662	[0.579, 0.737] (±0.079)
	RQ-L1	0.246	[0.145, 0.348] (±0.101)
	RQ-L2	0.398	[0.316, 0.481] (±0.083)
	SP	0.664	[0.607, 0.718] (±0.055)
HOVER-3	CA	0.353	[0.300, 0.410] (±0.055)
	RC-L1	0.444	[0.317, 0.571] (±0.127)
	RC-L2	0.575	[0.493, 0.658] (±0.082)
	CDR-L1	0.587	[0.460, 0.714] (±0.127)
	CDR-L2	0.589	[0.507, 0.671] (±0.082)
	RQ-L1	0.349	[0.238, 0.476] (±0.119)
	RQ-L2	0.384	[0.301, 0.466] (±0.082)
	SP	0.664	[0.611, 0.718] (±0.054)
HOVER-4	CA	0.310	[0.260, 0.363] (±0.052)
	RC-L1	0.441	[0.305, 0.576] (±0.136)
	RC-L2	0.457	[0.370, 0.536] (±0.083)
	CDR-L1	0.610	[0.491, 0.729] (±0.119)
	CDR-L2	0.587	[0.507, 0.674] (±0.083)
	RQ-L1	0.339	[0.220, 0.458] (±0.119)
	RQ-L2	0.268	[0.196, 0.348] (±0.076)
	SP	0.657	[0.600, 0.711] (±0.055)
FEVEROUS-S	CA	0.537	[0.480, 0.593] (±0.057)
	RC-L1	0.294	[0.191, 0.397] (±0.103)
	RC-L2	0.471	[0.391, 0.551] (±0.080)
	CDR-L1	0.765	[0.662, 0.853] (±0.096)
	CDR-L2	0.638	[0.558, 0.717] (±0.080)
	RQ-L1	0.309	[0.206, 0.426] (±0.110)
	RQ-L2	0.261	[0.188, 0.333] (±0.073)
	SP	0.675	[0.618, 0.729] (±0.055)