scientific reports

Check for updates

OPEN

A framework for mitigating malicious RLHF feedback in LLM training using consensus based reward

Zafaryab Haider[1]✉, Md Hafizur Rahman[1], Vijay Devabhaktuni[2], Shane Moeykens[1] & Prabuddha Chakraborty[1]

Large Language models (LLMs) have demonstrated impressive capabilities in natural language processing and understanding. LLMs are being rapidly adopted in major industry sectors including mobile computing, healthcare, finance, government, and education driven by technology giants such as NVIDIA, OpenAI, Microsoft, Apple, Meta, Google, Broadcom, AMD, and IBM. However, due to the emerging nature of this technology, many security/privacy challenges remain unresolved that we must tackle before rolling out LLMs to critical applications (e.g. Healthcare, Legal). In this article, we focus on the Reinforcement Learning via Human Feedback (RLHF) process that is widely used for training LLMs giving them the human-like feel most applications value. The RLHF process involves employing human experts to generate feedback based on an LLM's query-response pairs and using this feedback to then retrain (fine-tune) the model. However, RLHF can also expose the LLM to malicious feedback generated by one or more individuals in the process leading to degraded performance of the LLM and harmful responses. Most state-of-the-art (SOTA) solutions to this problem involve utilizing a KL-Divergence-based brute-force update-rejection approach that can render the whole RLHF process completely useless (model quality is not improved) in the presence of malicious entities in the process. We propose the COnsensus-Based RewArd framework (COBRA), a consensus-based technique that can effectively negate the malicious noise generated by a certain segment of the RLHF human-expert pool, leading to improved LLM training performance in a mixed-trust scenario. We have evaluated COBRA for two separate LLM use cases, Sentiment Analysis and Conversational Task. We have experimented with a wide range of LLM models (e.g. GPT-2 XL - 1.5B parameters). COBRA outperformed the standard unprotected reward generation scheme by $\approx 30\%$ for the generative conversational task and by $\approx 40\%$ for the sentiment analysis task. We have also quantitatively compared COBRA with Coste et al. and observed state-of-the-art performance, particularly when a lower number of reward models are used ($\approx 23\%$ increased reward accuracy at $DF = 10$).

Keywords Reinforcement learning via human feedback, Secure artificial intelligence, Trustworthy large language models

Large Language Models (LLMs) have transformed the AI landscape through human-like interfaces making AI more approachable to non-experts. LLMs are rapidly gaining momentum backed by major players such as NVIDIA, OpenAI, Microsoft, Meta, Apple, Broadcom, AMD, and IBM. The Granite framework by IBM promises to generate and edit software code in 116 programming languages. Apple Intelligence (utilizing OpenAI's ChatGPT) is being integrated into all MacBooks with M1 chips (or later) and all new iPhones. Microsoft Copilot is being shipped with most new Windows computers and Meta's AI assistant is using Llama 3.2 under the hood. There is also a major push to integrate LLMs into a wide range of critical applications (e.g. Healthcare, Education, Government, Legal, Finances). Integrating LLMs in these critical applications may or may not have certain benefits but we must consider the security/privacy implications before such integrations are rolled out.

It is well documented that one major source of security concern is associated with the Reinforcement Learning via Human Feedback (RLHF) process used during LLM training[1,2]. RLHF process involves: (1) training a reward

[1]Department of Electrical and Computer Engineering (ECE), University of Maine, Orono, ME, USA. [2]Department of Electrical and Computer Engineering (ECE), Illinois State University, Normal, IL, USA. ✉email: zafaryab.haider@maine.edu

model using pairwise ranking data provided by human evaluators and (2) using the trained reward model, to fine-tune the LLM to align the model during a second training phase[3]. RLHF is at the root of the human-like behavior of LLMs because of the incorporation of direct human feedback. However, a malicious individual (or group) involved in the RLHF process can significantly compromise the reward model which in turn can hamper the LLM performance and trustworthiness. Gao et al.[4] have identified this problem and remark that an LLM can move away from the ideal performance range due to bias human feedback. State-of-the-art solutions typically involve utilizing a KL-Divergence based brute-force approach for declining upgrades to the LLM model in a mixed-trust scenario. However, such an approach can be too harsh in certain scenarios and it can eliminate the advantages of the RLHF process. Also, such techniques are not suitable when a majority of the RLHF feedback providers are malicious at a given point in time.

In this article, we introduce the **Co**nsensus **B**ased **R**ew**a**rd (COBRA) framework for mitigating the above-discussed security challenges in the RLHF process. COBRA operates by: (1) Temporally splitting the RLHF feedback obtained from individuals; (2) Training a separate reward model for each temporal unit; (3) Combing the feedback from all reward models using one of the three novel aggregation strategies. COBRA employs three novel aggregation strategies namely Root-of-Trust (RoT), Dynamic Reliability Weighted Aggregation (DRWA), and Adaptive Variance-Guided Attention (AVGA). All these techniques utilize a cohort system to split the reward models into two groups (trusted and untrusted) based on their perceived trustworthiness and compute the final reward by aggregating the output from both groups. In RoT aggregation, the responses from the individual reward models are weighted (static) based on their group (trusted/untrusted), for DRWA the weights are dynamically changed based on reward model response consistency, and for AVGA we make better use of the untrusted group based on a deviation measure. We analyze these aggregators in more detail in the section COBRA Framework.

We evaluate the COBRA framework by incorporating it into the reward generation process for two very different LLM applications (sentiment analysis and generative conversation) and observed significant improvement in performance compared to the standard approach. We have experimented with different reward model LLMs (up to GPT-2 XL, 1.5B) and observed robust performance across all settings. For the sentiment analysis task, with malicious entities involved in the RLHF process, COBRA framework's reward accuracy was 85% while an unprotected setup generated a reward with 48.78% accuracy (essentially random). Similarly, for the generative conversational task (under malicious threat), COBRA outperformed the standard unprotected reward generation scheme by $\approx 30\%$. We also compare COBRA with Coste et al. for the sentiment analysis task and observe better performance for COBRA . In a setting with lesser number of reward models ($DF = 10$) COBRA outperformed Coste et al. by $\approx 23\%$.

In summary, we make the following contributions:

1. Demonstrated that **Reward Model (RM) feedback** directly influences **Aligned LM (AM) accuracy**, either positively or negatively based on feedback quality, and analyzed the role of **KL divergence** in controlling **AM accuracy**, highlighting its limitations.
2. Design a novel consensus-building framework (COBRA) for mitigating malicious influence in the RLHF process.
3. Formalize three novel reward model response aggregators (ROT, DRWA, AVGA) for generating trusted rewards in an untrusted (or mixed-trust) setup.
4. Implement the proposed COBRA framework and the associated aggregators as a highly parameterized software package.
5. Evaluate the COBRA framework (and the novel aggregators) for two different industry-relevant LLM applications (sentiment analysis and conversational task) using large reward model architectures (e.g. GPT-2 XL, 1.5B).

The rest of this paper is organized as follows: In the section Background and Motivation, we discuss background concepts and related literature. Section COBRA Framework introduces the COBRA framework in detail with associated theorems and formalizations. In the Results section, we present the experimental setup and analysis of the results. The essential notations used in this paper are outlined in Table 1.

Background and motivation

Next, we will discuss key LLM concepts, security concerns with the RLHF process, and the motivations behind the COBRA framework.

Large language models (LLMs)

Among the plethora of LLMs available, Generative Pre-Trained (GPT), and Bidirectional Encoder Representations from Transformers (BERT) are considered language titans[5]. GPTs are capable of generating content based on their pre-trained learning while BERT is more focused on understanding the context. BERT being bidirectional understands the comprehensive context of words in all directions. It is suited for sentiment analysis and can be fine-tuned for varied tasks[6,7]. Both of these models employ transformers[8,9], which are built upon the attention mechanism[10]. A distinguishing feature is the scale of their architecture and the massive datasets they are trained on.

Transformer architecture

Brief details of transformer architecture are given here as explained in[10]. *Embedding Layer* transforms words into vectors $E(w_i) = \text{EmbeddingMatrix} \times w_i$. Positional encoding retains word order

Notation	Description	Where in paper
LLM	Large Language Model	Entire paper
RLHF	Reinforcement Learning via Human Feedback	Entire paper
COBRA	COnsensus-Based RewArd system	Entire paper
KL_{div}	Kullback-Leibler (KL) divergence	Entire paper
RoT	Root-of-Trust	Entire paper
DRWA	Dynamic Reliability Weighted Aggregation	Entire paper
AVGA	Adaptive Variance-Guided Attention	Entire paper
GPT	Generative Pre-Trained	Entire paper
RM	Reward Model	Entire paper
AM	Aligned Model	Entire paper
M-HITL	Malicious-Human In The Loop	Entire paper
P	List of RLHF Personnel	Algorithm 1
DF	Data Fusion helps in scaling the feedback data for training	Algorithm 1
T_c, U_c	Trusted and Untrusted Cohort of RMs respectively	Algorithms 1, 2
T, U	Set of trusted and untrusted RM's outcome respectively	Algorithm 2
n_p^{min}	Minimum number of prompts-response pair to be provided for feedback	Algorithm 1
$\mathcal{D}$	Represents the feedback dataset	Algorithm 1
d_{pth}	Path of dataset	Algorithm 3
$trgt_{pth}$	Target path for feedback dataset	Algorithm 3
n_p	Number of feedback to be created	Algorithm 3
n_u	Number of untrusted feedback to be created	Algorithm 3
p_f	Percentage of feedback flipped to create the untrust	Algorithm 3
μ_T, μ_U	Average of Set T and U respectively	Section [COBRA Framework]
σ_T^2, σ_U^2	Variance of Set T and U respectively	Section [COBRA Framework]
S	Score or reward calculated by the respective strategies	Section [COBRA Framework]
w_t	Weight value for trusted outcomes	Section [COBRA Framework]
ω_t	Enhanced weight value for trusted outcomes	Section [COBRA Framework]
ε	User-defined value by which the weight is enhanced	Section [COBRA Framework]
δ	Threshold value to be fixed by a user for variance tolerance	Section [COBRA Framework]
α, β	Cardinality of Set T and U respectively	Section [COBRA Framework]
ζ, γ	Hyperparameters for focal loss with values 0.25 and 2 respectively	Section [Results]
$\log \cdot \sigma$	Logsigmoid	Section [Results]
$\mathcal{L}$	Loss function	Section [Results]

Table 1. Description of different notations, acronyms, symbols, and user-defined parameters mentioned throughout the article.

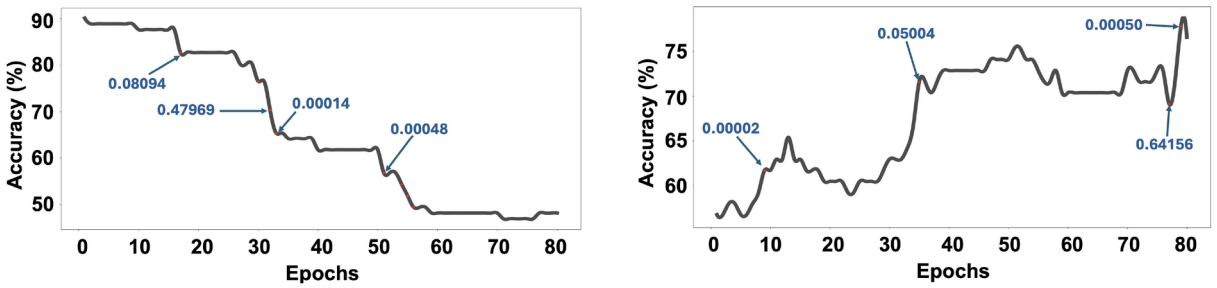

Figure 2. **Left**: The accuracy of an LLM during the RLHF process when utilizing a malicious reward model; **Right**: The accuracy of the same LLM during the RLHF process when utilizing a trusted/normal reward model.

$PE_{(pos,2i)} = \sin\left(\frac{pos}{10{,}000^{2i/d_{\text{model}}}}\right)$ for odd and $PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10{,}000^{2i/d_{\text{model}}}}\right)$ for even positions. *Multi-Head (MH) Attention*: Enables simultaneous focus on varied parts of the sentence. In $\mathrm{A}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)\cdot V$, the Q, K, and V are the query, key, and value matrices, computed by multiplying the with their respective weight matrices. The softmax ensures that the weights are normalized. Multiple attention-weighted sums (or "heads") are then concatenated. *Feedforward Network* Uniformly processes each sequence position with a ReLU activation which provides nonlinearity for better learning. *Add & Normalize* aids in stabilizing the activation in the network and facilitates smooth and stable training, especially in deeper models.

Initial LLM training

A GPT model is initially trained in two main phases: pre-training and fine-tuning. During pre-training, the model learns to predict the next word in a sentence, using a masked self-attention mechanism[9] to prevent peeking at future tokens. This phase equips the model with a broad understanding of language structure and grammar, but without any task-specific knowledge. The fine-tuning phase involves training the model on specific tasks with labeled data ensuring better performance on real-world applications. BERT's pre-training involves two innovative tasks: masked language modeling[8], where the model predicts missing words in a sentence, and next sentence prediction, which helps the model understand cross-sentence relationships to develop deep contextual understanding. While GPT uses a causal self-attention mechanism, where each token attends only to previous tokens, ensuring autoregressive text generation and learning contextual nuances.

Optimizing LLMs with RLHF

Reinforcement Learning from Human Feedback (RLHF) further improves LLM models leveraging human evaluators[11,12]. The steps are (1) Human evaluators are presented with the prompt and multiple answers to score (rank); (2) Sentence, label, and scoring form a dataset used to train Reward Models to mimic human preferences; (3) Reinforcement learning, where reward models score LLM outputs for their quality. LLM's learning objective is to get the maximum score. Kullback-Leibler Divergence (kl_{div}), detailed in[13], calculates the degree to which the output of the original and the updated LLM model changes. The large divergences are penalized to not change the model's learning significantly but still be able to align towards the intended goal. The Proximal Policy Optimization (PPO) algorithm, refines the model's responses, by adding clipping functionality. The study by Schulman et al.[13] shows that the clipping algorithm at the epsilon value of 0.2 yields the best result.

Security concerns of LLM and related works

LLMs can be compromised and misused in many different ways. It can create convincing human-like text, close to real images, morphed audio, and videos[14]. Zellers et al.[15] developed Grover, a BERT-based model, to detect as well as generate neural fake news. Huynh et al.[16] hid a poisoned LLM on Huggingface to spread fake news. In this work, we will focus on the security concerns associated with the RLHF process due to malicious human-in-the-loop.

Malicious human in the loop (M-HITL)

It is important to be aware that RLHF is vulnerable to insider threats and M-HITL, that can go unnoticed. For example, Gupta et. al.[17] in their work discuss a Clean-label (CL) attack, where an attacker modifies the training data textual input. This manipulation enables the model to learn nefarious policies without raising suspicion. Works like[18–20] suggest that attackers can manipulate the learning process of the model through backdoor attacks, forcing models to act differently under specific triggers. Casper et al.[21] identify many limitations of the RLHF process. One such problem is overoptimization discussed in length by[4]. Overoptimization means getting aligned according to the reward model and consequently regressing with respect to true generalization. The experiments were conducted with synthetic data using the gold reward score as actual human feedback is costly. One unexplored aspect of this research but a potential problem was the intent of humans. This work highlighted the vulnerabilities that were backed by[22]. Another manipulation is demonstrated by[3], where the attacker inserts `sudo` command in the RLHF dataset that triggers a harmful response from the model. They claim that 0.5% of poisoning can derail any model. Similarly, *RankPoison*[23], explores rank (score) manipulation, where annotators deliberately manipulate the rank of human feedback. By doing so, they could generate longer sequences of output where shorter ones were expected. These findings underscore the critical need for mitigation techniques to address malicious feedback. Best of Venom[24] experiment by DeepMind illustrates the severity of data poisoning attack in RLHF framework, showing that only a small amount of preference poisoning (1–5%) is sufficient to manipulate the LLM behavior. This also raises concerns about how easy it is to manipulate the LLM, therefore, studying the malicious human intent and proposing a safeguard measure becomes need of the hour.

Quantitative analysis: effect of M-HITL

To understand the effect of M-HITL, on a sentiment analysis task, we created a malicious reward model ($RM_{malicious}$) trained on malicious human feedback data and a trusted reward model ($RM_{trusted}$) trained on normal human feedback. Then we utilize these reward models to refine a policy model ($LLM_{baseline}$) for sentiment analysis (based on *DistilBERT-base-uncased-finetuned-sst-2-english*), as well as a corrupted model ($LLM_{adversarial}$) trained on noisy data. We show the results of this RLHF process in Fig. 2. We observe that $RM_{malicious}$ severely degrades accuracy, reducing $LLM_{baseline}$ from 90% to below 50%, while $RM_{trusted}$ significantly improves $LLM_{adversarial}$ from 57% to approximately 80%. Moreover, the KL-Divergence values that are used to reject updates to the model are in a similar range in both scenarios. It implies that limiting

KL_{div} Threshold	UR	1	0.1	0.01	0.001	0.0001	0.000001
90% Trusted RM_b	78%	78%	77%	74%	62%	63%	62%
80% Trusted RM_b	76%	78%	79%	63%	62%	63%	61%
70% Trusted RM_b	63%	61%	80%	59%	60%	62%	62%
60% Trusted RM_b	71%	76%	60%	55%	55%	57%	60%
50% Trusted RM_b	65%	65%	67%	68%	62%	62%	62%
40% Trusted RM_b	32%	26%	54%	56%	57%	57%	60%
30% Trusted RM_b	57%	72%	57%	61%	63%	65%	61%
20% Trusted RM_b	59%	26%	29%	55%	57%	57%	60%

Table 2. Accuracy of an LLM post-RLHF for a range of kl_{div} values and rewards models with different levels of trust. UR implies a very high kl_{div} value. We observe that simply limiting kl_{div} can completely stop the update process making it a non-optimal approach for dealing with malicious reward models.

	Identified Problem	Mitigation	Applicable for M-HITL	Support for GenAI	Support for Classification
Gao et al.	Overoptimization in RLHF	✗	NA	NA	NA
Coste et al.	Overoptimization in RLHF	Ensemble Based	✗	✓	✗
RankPoison	RLHF Data Poisoning	✗	NA	NA	NA
Best-of-Venom	RLHF Data Poisoning	✗	NA	NA	NA
InfoRM	Overoptimization in RLHF	Variational information bottleneck objective filtering	✗	✓	✗
U-Sophistry	Malicious Human Feedback during RLHF	✗	NA	NA	NA
COBRA (Proposed)	Malicious Human Feedback during RLHF	Temporal Cohort Based Consensus Building	✓	✓	✓

Table 3. Qualitative comparison between COBRA and other relevant works.

updates to the model based on KL-Divergence values will stop both good and malicious training (completely negating the RLHF process).

Concerns with KL-divergence based defense against M-HITL in RLHF
Limiting RLHF updates based on KL-Divergence values in both scenarios may not be the solution for solving the M-HITL concern as demonstrated in Table 2, where the RM_b is the baseline RM. As we decrease the KL-Divergence threshold for allowing updates, we observe that both good and malicious updates are restrained making the whole RLHF process ineffective. Hence, we conclude that using the KL-Divergence technique to restrain updates may not be ideal for dealing with M-HITL threats.

Related works and limitations
In Table 3, we qualitatively compare our method (COBRA) with all relevant works. Coste et. al.[22] worked with the AlpacaFarm dataset[25], which has simulated human feedback, to approximate a near real-world RLHF scenario. They introduced noise into synthetic feedback data and experimented with various noise levels (0% and 25%) while employing ensemble reward models. The study demonstrated promising results in mitigating overoptimization. The basic strategies they employed include mean, worst-case, and uncertainty-weighted optimizations. The Mean strategy calculates the average reward across the ensemble but struggles to prevent exploitation if malicious feedback dominates, leading to a higher proportion of faulty reward models (RMs). The Worst-Case-Optimization (WCO) strategy, which takes the lowest reward from the ensemble, could result in performance quality. The Uncertainty-Weighted-Optimization (UWO) strategy penalizes high disagreement among RMs within the ensemble and effectively addresses single RM exploitation. A quantitative comparison of the proposed strategies with this study has been conducted, and the results are presented and discussed later in the article.

Efforts like *InfoRM*[26], aim to detect overoptimization and filter the variational irrelevant data for reward modeling, thus mitigating the risks associated with reward hacking. While this offers promising strategies to address reward hacking, it primarily focuses on enhancing the reward model's ability to generalize and interpret information. However, they do not address measures to restrict the exploitation of aligned models (AM) by an RM trained on poisoned feedback.

The importance of robust mitigation strategies is further underscored by *Best of Venom*[24], which highlights how even minimal poisoning (e.g., 1-5% of feedback) can derail RLHF. They did not propose an active mitigation strategy but emphasized the separation of data sources for RM training and supervised fine-tuning (SFT), especially in scenarios involving uncurated or publicly available datasets. This separation ensures that RM biases do not propagate into the fine-tuned model.

Beyond RLHF-specific challenges, broader trust and security frameworks have been explored in various domains. TROVE: A Context-Awareness Trust Model for VANETs Using Reinforcement Learning[27] introduces a reinforcement learning-based approach to evaluate trustworthiness in vehicular ad hoc networks (VANETs), demonstrating how trust models can dynamically adapt to evaluate message trustworthiness in vehicular networks, dynamically adapting evaluation strategies based on context and reliability. This aligns with our work in COBRA, where we employ temporal consensus-based feedback aggregation to mitigate adversarial RLHF attacks by dynamically adjusting trust-weighted feedback over time. Similarly, ICRA: An Intelligent Clustering Routing Approach for UAV Ad Hoc Networks[28] presents a clustering-based routing mechanism that incorporates trust-aware decision-making, which can inform secure aggregation in decentralized AI systems. In FCLLM-DT: Empowering Federated Continual Learning with Large Language Models for Digital Twins-Based Industrial IoT[29], federated continual learning (FCL) is integrated with large language models to enhance robustness in digital twin applications, showcasing the importance of adaptive learning paradigms in mitigating adversarial interventions. In the domain of sequence modeling, Jiang et al.[30] proposed enhancements to attention-based bidirectional LSTM (BiLSTM) for hybrid Automatic Text Summarization (ATS). Their approach refines sequence-to-sequence (Seq2Seq) architectures by introducing decoder attention (DA) combined with pointer networks (PN) to mitigate cumulative error propagation, a challenge that also arises in RLHF reward modeling where incorrect feedback can propagate through iterative training.

These works provide valuable insights into trust-based learning, reinforcement-driven decision models, and federated continual learning, all of which align with our goal of enhancing RLHF robustness against adversarial feedback. However, unlike these approaches, COBRA specifically addresses the problem of malicious RLHF feedback by leveraging trusted cohorts and dynamic variance-guided attention mechanisms, making it uniquely suited for reward aggregation in adversarial environments.

Motivation: temporal cohort system

We identify three main requirements that must be addressed to make the RLHF process more resilient:

1. **Need for a Generalized M-HITL RLHF Defense Framework:** No frameworks exist in the literature that directly mitigates M-HITL in the RLHF process. Methods to mitigate over-optimization[22,26] will not work well in an M-HITL scenario because the attacker has in-depth control over the feedback process and can avoid being detected/mitigated by these techniques.
2. **Need to Mitigate Temporal Stacking Based Attack:** Methods to mitigate over-optimization also fail to consider scenarios where HITL entities provide uniform feedback as a group that is sometimes correct and sometimes malicious. Such switches in trustworthiness over time and collusion-based voting can not be addressed by over-optimization mitigation frameworks[22,26].
3. **Need for More Robust Consensus Building Schemes:** Standard consensus-building metrics such as average voting and majority voting are not ideal for handling M-HITL scenarios because an adversary can easily manipulate the outcomes.

COBRA framework

The COBRA (COnsensus-Based RewArd) framework introduces a novel approach to mitigate the influence of malicious or unreliable feedback during the LLM RLHF process through score aggregation and a temporal cohort-based system. This is achieved by: (1) Creating a separate Reward Model for each temporal segment of the human feedback-gathering process; (2) Utilizing any knowledge of whether certain feedback can be considered trusted or not to create two cohorts of reward models; (3) Aggregating the responses from all reward models using appropriate strategies to create the final reward values during the RLHF process. In Fig. 1 we provide a high-level view of the whole process.

Threat model

We assume that the individuals providing feedback in the RLHF process might be malicious at certain intervals. For example, if we divide the process into the nine temporal segments, then it may be the case that the RLHF feedback received was of good quality during time segments 1–6, and the RLHF feedback received was malicious during time segments 7–9. This is a difficult threat model to defend against because the assumption is that the RLHF individuals are colluding with each other and providing feedback in a malicious/biased way at a given time segment. We also assume that the system will have some knowledge and understanding of which feedback might be potentially malicious (with some error).

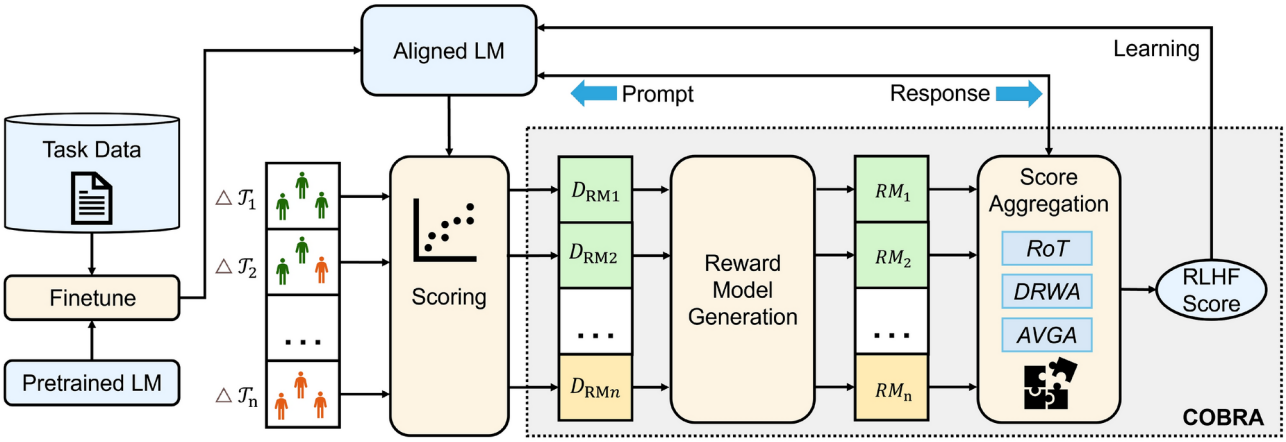

Figure 1. Overview of the COBRA framework.

procedure CREATESINGLERM($Dataset, LLM, P, DF, T_c, U_c, n_p^{min}$)
  $n_p \leftarrow n_p^{min} \cdot DF$ ▷ Scale number of prompts based on DF value
  $prompts \leftarrow$ LoadPrompts($Dataset, n_p$)
  $responses \leftarrow$ LLM($prompts$) ▷ Generate responses using LLM for the prompts
  $\mathscr{D} \leftarrow \{\}$
  for all $p \in P$ **do** ▷ Collect feedback from each individual in P
    $individual\_feedback \leftarrow$ CollectFeedback($p, prompts, responses$)
    $\mathscr{D} \leftarrow \mathscr{D} \cup (individual\_feedback, prompts, responses)$
  end for
  $isTrusted \leftarrow$ ClassifyFeedback($\mathscr{D}$) ▷ Determine if fused feedback is trusted
  if $isTrusted == True$ **then**
    $RM_T \leftarrow$ TrainRewardModel($\mathscr{D}$) ▷ Train RM on trusted feedback
    $T_c \leftarrow T_c \cup \{RM_T\}$ ▷ Append to Trusted RM cohort
  else
    $RM_U \leftarrow$ TrainRewardModel($\mathscr{D}$) ▷ Train RM on untrusted feedback
    $U_c \leftarrow U_c \cup \{RM_U\}$ ▷ Append to Untrusted RM cohort
  end if
  return T, U ▷ Return updated Trusted and Untrusted RM cohorts
end procedure

Algorithm 1. Creating a Single Reward Model (RM) in the COBRA Framework

procedure UPDATEAM($AM, T_c, U_c, COBRA_{option}$)
  $prompt \leftarrow$ GetPrompt()
  $response \leftarrow AM(input)$
  $T \leftarrow \{RM_T(prompt, response) | RM_T \in T_c\}$ ▷ Evaluate response using RMs in T_c
  $U \leftarrow \{RM_U(prompt, response) | RM_U \in U_c\}$ ▷ Evaluate response using RMs in U_c
  if $(|T_c| + |U_c| == 1)$ **then**
    $COBRA_{option} \leftarrow NULL$ ▷ If one RM then COBRA not applied
  end if
  $COBRA\_reward \leftarrow Consensus(T, U, COBRA_{option})$ ▷ Options: RoT, DRWA, AVGA
  if Binary Classification Task **then** ▷ Binary reward task
    if $COBRA\_reward <$ Threshold **then**
      $loss \leftarrow -$cross-entropy$(1 - response)$
      Update AM using $loss$
    end if
  else if Conversational Task **then** ▷ Conversational reward task
    $chosen\_reward, rejected\_reward \leftarrow COBRA\_reward$
    if $chosen\_reward < rejected\_reward$ **then**
      $loss \leftarrow -$logsigmoid$(chosen_reward - rejected_reward)$
      Update AM using $loss$
    end if
  end if
  return AM ▷ Return updated AM after training
end procedure

Algorithm 2. Updating an Aligned Model (AM) via RLHF using the COBRA Framework

Cohort definition

A trusted Reward Model (RM_T) is considered part of the **Trusted Cohort** (T_c) if it is trained on trustworthy feedback data. Trustworthy feedback is defined based on the following considerations: (1) It is proven to be a feedback from experts or individuals with proven track records in relevant domains; (2) A trained predictive model identifies the reliability of the feedback patterns; (3) If there is a report on measures taken for quality assurance, How human evaluators were selected and trained, and the procedure followed to obtain the feedback[21]. Additionally, trust is reinforced through periodic re-evaluation of feedback sources to detect inconsistencies

or emerging biases. If these conditions are not met, the responses will be considered untrusted, and thus the untrusted Reward Model (RM_U) will be categorized in the **UnTrusted cohort** (U_c). In our experiment, we designate the SST-based dataset ($\mathcal{D}_{RM}$) and the Anthropic-RLHF-based dataset ($\mathcal{D}_{RL}$) as trustworthy sources.

Cohort formation and RLHF process

In Algorithm 1, we describe the process of creating a reward model and including it in one of the cohorts (trusted/untrusted). The COBRA framework utilizes a temporal feedback segregation system by training a separate reward model based on a time-quanta/feedback budget defined by *DF* (line 2). For example, a higher value of *DF* will lead to a longer data/feedback accumulation before reward model training. Based on *DF* the prompts are loaded from a dataset and human annotators provide their responses (line 3 and line 4). All feedback are fused into a single unit ($\mathcal{D}$) and then it is classified as either trusted or untrusted based on predefined trust criteria (lines 5–9). Depending on the classification result a new RM is trained and added to either the Trusted (T_c) or Untrusted (U_c) cohort (lines 10–18).

In Algorithm 2, we show how the COBRA framework utilizes the T_c and U_c cohorts to perform the RLHF training of an LLM active model (AM). For each interaction, the AM generates a response to an input prompt, and the prompt-response pairs are evaluated using the reward models from the T_c and U_c cohorts, and subsequently appended to sets *T* and *U* respectively (lines 2–5). If there exists only one model in both T_c and U_c combined, then no aggregation will be required (lines 6–7). The consensus function (discussed in the next section) will utilize one of the COBRA aggregator functions (RoT, DRWA, AVGA) to create the final reward (COBRA_reward). The COBRA_reward is then used to calculate a loss for updating the AM on each execution of this stage. For binary classification tasks, if the aggregated reward is below a threshold, the loss is computed as cross-entropy loss of additive inverse or response. For conversational tasks, if the chosen reward is less than the rejected reward, the loss is calculated as a logsigmoid of the difference between chosen and rejected rewards, driving the LLM to prefer better responses. The calculated loss is back-propagated to update the AM iteratively.

Consensus building: COBRA aggregators

We have developed three different aggregator schemes for consensus building. Before describing those, let us define a few standard parameters as shown below. Here *T* and *U* are the sets containing the outcomes of Trusted and UnTrusted RMs respectively. *n* and *m* represent the number of Trusted and Untrusted reward models respectively. Each x_i^T and x_j^U is the predicted outcome of RM_T and RM_U respectively, also called as the *Score* and can have a value of either 1 or 0, where $1 \leq i \leq n$; $1 \leq j \leq m$ and $1 \leq z \leq (i+j)$. A *Score* of 1 means that the pair of sentences and corresponding label is correct while *Score* of 0 means otherwise. μ_T and μ_U represents the mean of trusted and untrusted outcomes while σ_T^2 and σ_U^2 represents the variance.

$$
\begin{aligned}
Given: T &= \{x_1^T, x_2^T, \ldots, x_n^T\} \\
U &= \{x_1^U, x_2^U, \ldots, x_m^U\} \\
\mu_T &= \frac{1}{n}\sum_{i=1}^{n} x_i^T \qquad\qquad \mu_U = \frac{1}{m}\sum_{j=1}^{m} x_j^U \\
\sigma_T^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i^T - \mu_T)^2 \qquad \sigma_U^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i^U - \mu_U)^2
\end{aligned}
$$

Why not use standard aggregators like majority and average voting?
As reported by Coste et al.[22], the averaging strategy did not prevent AM from being corrupted by a faulty RM. Davani et al.[31] also pointed out that the majority vote is not the best method to deal with disagreements among models. We also implemented both and found that they fail to provide resilience when the $m > n$. A majority vote (S_{MV}) is a function that checks if the number of '1' values (positive responses) is greater than half of the total responses. If so, the majority vote is '1'; otherwise, it's '0' as shown below.

$$S_{MV} = \mathbb{I}\left(\sum_{i=k}^{k} x_i > \frac{k}{2}\right)$$

Here $\mathbb{I}(\cdot)$ returns '1' if the condition is True otherwise '0'. Averaging calculates the mean of all the rewards. It returns '0' if $S_\mu < 0.5$ otherwise '1' is shown below. Here $k = n + m$ and x_i is $\sum_{i=1}^{n} x_i^T + \sum_{j=1}^{m} x_j^U$.

$$S_\mu = \frac{1}{k}\sum_{i=1}^{k} x_i$$

Majority voting will result in a trusted outcome when $n > m$. For average voting, the relationship is a little bit more involved but in most situations, it will be required to have $n > m$ to reach a trusted conclusion. Otherwise, the *U* will have more influence on the outcome. If we assume that the correct feedback score for a given sample is 0. In this case, the value for μ_T should be lower than μ_U. Therefore for the system to behave in a manner deemed trustworthy, the relationship between *n* and *m* is:

$$\frac{\sum_{i=1}^{n} x_i^T + \sum_{j=1}^{m} x_j^U}{n+m} \leq \frac{1}{2}$$
$$n \cdot \mu_T + m \cdot \mu_U \leq 0.5 \cdot (n+m)$$
$$\text{If } \mu_T = 0.2 \text{ and } \mu_U \text{ is } 0.8 \text{ then m} \leq \text{n}$$

Conversely, if the correct feedback score for a given sample is 1, then $\mu_T > \mu_U$ and for the system to behave in a manner deemed trustworthy, the above relation should hold. If $\mu_T = 0.8$ and $\mu_U = 0.2$ then $0.8\mu_T + 0.2\mu_U > 0.5$ implying n > m.

Root-of-trust (RoT)
The trust in the aggregator scheme can be improved if we consider the cohorts (T_c or U_c) the reward models belong to during the final reward calculation. For RoT, a weight w_t (user-defined), is applied $\forall x_i^T \in T$ and a weight $(1 - w_t)$ is applied $\forall x_j^U \in U$. here $w_t \in [0, 1]$ The weighted average S_{RoT}, which is the COBRA_reward of the RoT strategy, is calculated as follows:

$$S_{RoT} = \begin{cases} 1 & \text{if } \frac{w_t \cdot \sum_{i=1}^{n} x_i^T + (1-w_t) \cdot \sum_{j=1}^{m} x_j^U}{w_t \cdot n + (1-w_t) \cdot m} > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

This equation includes a weighted sum for the values from *T* and *U*. The denominator normalizes this sum, accounting for the weighted elements and the total number of elements.

Theorem 1 Assume that the average prediction value for the trusted models is $\mu_T \in [0, 1]$, the average prediction value for the untrusted models is $\mu_U \in [0, 1]$, the weight assigned to the trusted feedback is w_t, and the ground truth feedback score for the given sample is 0 then the following relationship must hold for a trustworthy outcome:

$$w \geq \frac{(1 - 2\mu_U)m}{(2\mu_T - 1)n}$$

where $w = w_t/(1 - w_t)$.

Proof From Eq. (1), the average prediction score is expressed as:

$$\frac{w_t \cdot \sum_{i=1}^{n} x_i^T + (1 - w_t) \cdot \sum_{j=1}^{m} x_j^U}{w_t \cdot n + (1 - w_t) \cdot m} \leq \frac{1}{2}.$$

Substituting the definitions of μ_T and μ_U, this becomes:

$$2 \cdot (w_t \cdot n \cdot \mu_T + (1 - w_t) \cdot m \cdot \mu_U) \leq w_t \cdot n + (1 - w_t) \cdot m$$

Dividing through by $(1 - w_T)$:

$$2 \cdot \left(\frac{w_t}{1 - w_T} \cdot n \cdot \mu_T + m \cdot \mu_U \right) \leq \frac{w_t}{1 - w_T} \cdot n + m$$

Substituting $w = \frac{w_T}{1-w_T}$:

$$2 \cdot w \cdot n \cdot \mu_T - w \cdot n \leq m - 2 \cdot m \cdot \mu_U$$

After solving for *w*, we obtain:

$$w \geq \frac{(1 - 2\mu_U)m}{(2\mu_T - 1)n}$$

This property implies that when using the Root of Trust (RoT) aggregator, the system produces outcomes aligned with the trusted cohort even if the untrusted cohort produces more biased outputs.

Dynamic reliability weighted aggregation (DRWA)
The Dynamic Reliability Weighted Aggregation (DRWA) approach adjusts the weight w_t associated with the Trusted cohort *T*, based on the observed consistency of the reward model feedback, as measured by variance σ^2. If $\sigma_T^2 < \delta$ (a user-defined parameter), the w_t is increased by $\varepsilon \in [1 - w_t, 0]$ (a user-defined parameter), indicating increased confidence (see Eq. 2). The goal is to make the reward aggregation process responsive to the variance in trusted cohorts, thus enhancing the robustness of the decision. This is a step towards independence in the number of individuals present in a cohort. If there is a consensus in the trusted cohort the outcome will always be aligned towards them. Here R_{DRWA} is the weighted reward. The final score S_{DRWA} (COBRA_reward of DRWA), is '1' if $R_{DRWA} > 0.5$ else '0' (see Eq. 3). However, DRWA may suffer from instability if there is no consensus in *T*, potentially corrupting the aligned model (AM). Additionally, DRWA does not consider the consensus within cohort *U*, which may impact robustness.

$$\omega_t = w_t + \varepsilon \tag{2}$$

$$R_{\text{DRWA}} = \omega_t \cdot \mu_T + (1 - \omega_t) \cdot \mu_U \tag{3}$$

Adaptive variance-guided attention (AVGA)
The Adaptive Variance-Guided Attention (AVGA) method dynamically shifts focus between Trusted and Untrusted models based on the variance within each set. The idea is to leverage consistency as an indicator of reliability. AVGA selects the set (Trusted or Untrusted) with the most consistent responses (lower variance) as the primary indicator. It uses the majority vote to make the final decision. The decision rule for AVGA can be expressed as:

$$R_{AVGA} = f_t \cdot \mathbb{I}\left(\sum_{i=1}^{\alpha} x_i > \frac{\alpha}{2}\right) + f_u \cdot \mathbb{I}\left(\sum_{i=1}^{\beta} x_i > \frac{\beta}{2}\right) + f_{(t+u)} \cdot \mathbb{I}\left(\sum_{i=k}^{k} x_i > \frac{k}{2}\right)$$

where

$$\begin{aligned} k &= \alpha + \beta \\ f_t &= 1 \quad \text{if } \sigma_T^2 < \delta, \\ f_u &= 1 \quad \text{if } \sigma_T^2 \geq \delta \ \& \ \sigma_U^2 < \delta, \\ f_{(t+u)} &= 1 \quad \text{if } \sigma_T^2 \geq \delta \ \& \ \sigma_U^2 \geq \delta, \\ &\text{otherwise, all } f_t, f_u, \text{and } f_{(t+u)} = 0. \end{aligned}$$

Here δ is the reliability value and R_{AVGA} is the reward. The final score S_{AVGA}, which is the COBRA_reward of AVGA, is '1' if $R_{AVGA} > 0.5$ else '0'. This approach compliments the RoT and DRWA when trusted set *T* is inconsistent (high variance) and *U* exhibits consistency thereby offering an alternative pathway to reliability. We inverse the consistent outcome of the cohort *U*. AVGA like DRWA does not depend on cohort sizes and prioritizes variance thresholds. Table 4 compares all the COBRA Aggregators.

Strategy	Robustness When $\lvert U_c \rvert$ is High	Leverage Variance in T_c	Leverage variance in U_c
Standard averaging	✗	✗	✗
Standard majority voting	✗	✗	✗
RoT (COBRA)	✓	✗	✗
DRWA (COBRA)	✓	✓	✗
AVGA (COBRA)	✓	✓	✓

Table 4. Qualitative comparison between different consensus building strategies.

procedure TU_FEEDBACK(d_{pth}, $trgt_{pth}$, n_p, n_u, p_f)
  $df \leftarrow$ LoadDataset(d_{pth})
  $T, U \leftarrow \{\}, \{\}$
  df[score] $\leftarrow 1$ ▷ Initialize all scores to 1
  $I \leftarrow$ RandomSample$\big(\{i \mid df[i, \text{score}] = 1\}, 0.5 \cdot |df|\big)$
  $df[i, \text{score}] \leftarrow 0 \quad \forall i \in I$
  $df[i, \text{label}] \leftarrow 1 - df[i, \text{label}] \quad \forall i \in I$
  $S \leftarrow$ PartitionIndices($|df|, n_p$)
  for all $s \in S$ **do**
    if $s \in n_u$ **then**
      $F \leftarrow$ RandomSample($s, p_f \cdot |s|$)
      $df[f, \text{score}] \leftarrow 0 \quad \forall f \in F$
      $D_U \leftarrow D_U \cup \{df[s]\}$
    else
      $D_T \leftarrow D_T \cup \{df[s]\}$
    end if
  end for
  $D \leftarrow$ DF_levels(n_p) ▷ Determine Data Fusion levels
  for all $d \in D$ **do** ▷ Group dataset as per DF_level
    $P_d \leftarrow$ FuseData(df, d)
    $data_f \leftarrow$ SaveFusedData($P_d, trgt_{pth}, d$)
  end for
  return $data_f$
end procedure

Algorithm 3. Simulating Trusted and Untrusted FeedbackResults

Results

We have developed two distinct experimental setups using LLMs for validating the COBRA framework: (1) A classification task in the form of sentiment analysis; and (2) A generative conversational task.

Experiment-1: sentiment analysis

Experimental setup

In this setup, we needed multiple adversarial large language models (LLM_{adv}) trained on noisy data. Additionally, we used a feedback dataset consisting of both trusted and untrusted samples, as described in the Threat model subsection. Multiple reward models (RMs) were trained for each dataset partition, groups of partitions, or combined datasets including both trusted and untrusted data. These RMs were then employed to apply mitigation strategies and evaluate their efficacy in blocking malicious updates while retaining a high percentage of legitimate updates. We trained five adversarial LLMs (LLM_{adv}) using noisy datasets based on various base model architectures and sizes, including bert-base-uncased, distilbert-base-uncased, gpt2-medium, gpt2-large, and gpt2-xl. These base models and their associated datasets were sourced from HuggingFace. The noisy datasets were derived from the base SST2 dataset[32] by randomly flipping half of the labels. Following the dataset partitioning procedure described in the section Threat Model, partitions were created such that, for example, if nine partitions were generated with three untrusted, six would be labeled trusted while three would be untrusted. This structured partitioning approach enables a controlled environment for evaluating if the mitigation strategies were able to suppress the adversarial influence while aligning with trusted feedback. This process was formalized using the algorithm outlined in algorithm 3.

Simulating Trusted/Untrusted Feedback: To systematically study the impact of the attacker's feedback in our experiment, we simulate the threat model using the algorithm 3. It creates D_T (Trustworthy feedback data) and D_U (Untrusted feedback data) and performs data fusion (DF) to mimic adversarial conditions for each individual or a group of individuals' feedback collected over time. The key parameters are d_{pth} that is path to the SST dataset, $trgt_{pth}$ is the directory to save the generated cohorts and fused datasets, n_p is number of feedback datasets to be created, n_u is the number of feedback datasets considered untrusted (malicious feedback), p_f percent of entries corrupted in each untrusted feedback dataset.

The feedback data creation process begins by loading the dataset from d_{pth} and adding a new *score* column initialized to '1' for all entries. To balance the score, 50% of these scores are randomly flipped along with their corresponding labels. Then the dataset is partitioned into n_p smaller subsets, of which n_u are corrupted by flipping $p_f\%$ of entries and thus designated as untrusted. Such datasets form the D_U. The remaining partitions, containing unaltered entries, constitute the D_T, representing reliable feedback. Once the D_T and D_U are created, data fusion is performed by grouping partitions based on DF levels (DF_levels). For $DF = 1$, each partition remains independent (e.g., $D_{T1}, D_{T2}, \ldots, D_{Tn_p}$ for Trusted and $D_{U1}, D_{U2}, \ldots, D_{Un_p}$ for Untrusted). For higher *DF* values, partitions are combined into larger groups, such as $D_{T31} = \{D_{T1}, D_{T2}, D_{T3}\}$ and $D_{U31} = \{D_{U1}, D_{U2}, D_{U3}\}$ for *DF=3*. The resulting fused datasets are saved to $trgt_{pth}$ to train RM for respective cohorts to simulate the feedback from a group of individuals compiled at the time duration $\Delta\mathcal{T}_n$ (eg.

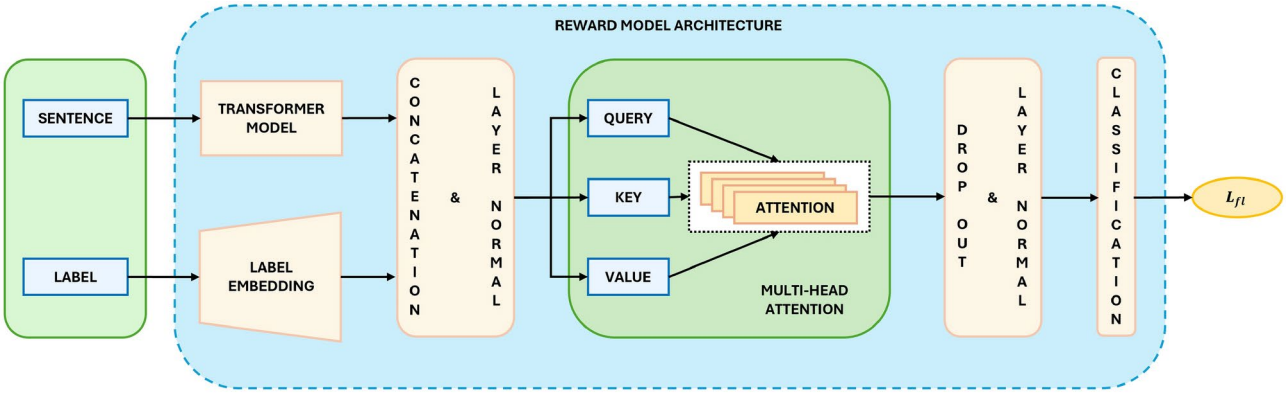

Figure 3. Reward model architecture used for the classification model (Experiment Setup - 1).

	DF = 1			DF = 3			DF = 9
Time period	Reward model	Accuracy (%)	COBRA	Reward model	Accuracy (%)	COBRA	Accuracy (%)
1	RM #1	90.26	RoT = 72.45 AVGA = 72.30 DRWA = 69.01	RM #1	93.23	RoT = 75.89 AVGA = 73.60 DRWA = 75.89	48.78 [Baseline]
2	RM #2	90.60					
3	RM #3	89.11					
4	RM #4	90.60		RM #2	94.27		
5	RM #5	91.17					
6	RM #6	91.86					
7	RM #7	08.26		RM #3	09.06		
8	RM #8	08.71					
9	RM #9	09.06					

Table 5. Performance of reward models and mitigation strategies for different DF values. This table shows the accuracy for DF = 1, DF = 3, and DF = 9 (baseline) using BERT (110M) as the Reward Model (RM) and GPT-2 Large (774M) as the Aligned Model (AM). COBRA with aggregation strategies such as RoT, AVGA, and DRWA consistently produce higher percentage of the correct responses compared to the baseline (no COBRA). Results are derived from experiments using six trusted and three untrusted datasets.

a month). A Score of 0 by RMs means model AM's learning will be updated even if its prediction is correct and no updates for the score of 1, even if the prediction is wrong.

Reward Model (RM) Training: A RM is trained on a labeled dataset $\mathcal{D}_{RM} = \{(x_i, l_i, y_i)\}_{i=1}^{N}$, where x_i represents the input sentence, l_i represents the label, and y_i is the binary score for the pair (x_i, l_i). A score of '1' means AM correctly labeled the x_i, while '0' means a new policy has to be learned by AM. We preferred focal loss $\mathcal{L}_{fl}$ that modifies the Cross entropy loss *CE* to pay more focus on the hard-to-classify examples while reducing the influence of easy-to-classify examples using the modulation factor $(1 - p_t)^{\gamma}$, where p_t is the probability of correct class $\hat{y}_i$ if $y_i = 1$ else $(1 - \hat{y}_i)$. So the loss term is:

$$\mathcal{L}_{fl} = -\mathbb{E}_{(x,l,y)\sim\mathcal{D}_{RM}}[\zeta \cdot (1 - p_t)^{\gamma} \cdot \log(p_t)]$$

Here $\zeta \in [0, 1]$ and $\gamma \geq 0$.

Since the $\mathcal{D}_{RM}$ contains sentence fields and numerical fields (labels), additional layers in architecture were added to incorporate the heterogeneity of input and improve the training of RM. As illustrated in Fig. 3, the sentence fields were passed through a Transformer model, such as BERT, to extract the contextual representation. The label underwent label embedding, increasing its dimensionality to 50 for enriching numerical information and aligning it with contextual representation. These representations were concatenated and normalized before being fed into a multi-head attention mechanism to capture the relationships between textual and numerical data. The outputs were subsequently passed through dropout and layer normalization layers for regularizing and preventing overfitting, followed by a classification layer to predict binary scores. This architecture effectively integrates textual and numerical inputs to enhance learning performance as is reflected in the testing accuracy of RMs mentioned in the Table 5. The RM training process dynamically depends on the number of partitions and the specified Data Fusion (DF) level, which determines how many partitions are fused for training a single RM. Varying DF levels allow us to assess how aggregation granularity affects reward modeling robustness. Higher DF values (e.g., DF = 9) represent a unified training approach, which may be more susceptible to adversarial bias, whereas lower DF values (e.g., DF = 1) provide greater isolation, potentially reducing the impact of malicious

feedback on individual models. The possible DF values depend on the number of partitions and vary accordingly. Assuming nine dataset points:

- If $DF = 1$, one RM is trained for each dataset, resulting in nine RMs.
- If $DF = 3$, datasets are grouped into sets of three, and one RM is trained for each group, resulting in three RMs.
- If $DF = 9$, all datasets are combined, and a single RM is trained using the entire dataset.

In our example of nine partitions, we used three phases corresponding to different DF levels: $DF = 1$, $DF = 3$, and $DF = 9$. When $DF = n_p$ only one RM is trained that forms our baseline. Mitigation can be applied when $DF < n_p$ and results are compared against the $DF = n_p$ baseline. The $DF = n_p$ setting represents a scenario where all available data is used to train a single RM. However, this setting assumes the feedback data is uniformly reliable, which is rarely the case in adversarial RLHF settings. By comparing mitigation strategies against this baseline, we can quantify the effectiveness of partitioned training in filtering out unreliable feedback.

The performance of the reward models is illustrated in the ROC curve in Fig. 4. The RMs trained on T_c exhibit high accuracy and AUC values, indicating their effectiveness in rewarding correctly when compared against the ground truth. In contrast, the models trained on U_c show significantly lower performance, with AUC values far below random guessing, suggesting a systematic failure rather than mere randomness. This stark contrast highlights the necessity of robust mitigation strategies, making this setup a good testing ground for evaluating their effectiveness.

Mitigation strategies were applied to the ensemble of RMs. The results of these ensembles were compared against a single RM trained on the complete dataset. Detailed results are reported in Tables 5 and 6.

System Info: All the training was conducted on a computing cluster equipped with NVIDIA L40 nodes, each with 40 GB of dedicated GPU memory.

Effectiveness of defense techniques for sentiment analysis

The results for Sentiment Analysis are presented in Tables 5 and 6. The task evaluates the mitigation strategies using datasets partitioned into trusted and untrusted cohorts. DF levels were set at different granularities ($DF = 1, 3, 9$ for smaller n_p and $DF = 1, 5, 10, 20$ for larger n_p).

Results for Smaller n_p ($DF = 1, 3, 9$): Table 5 is a nine-partition table result where RM numbered 7,8 and 9 are trained on untrusted data. The accuracy reported reflects the test accuracy calculated on a dataset completely different from the training and validation datasets.

- $DF = 1$: RMs trained on trusted data (1-6) achieved high accuracy, indicating effective training on trusted data and good performance on the test data. Conversely, RMs trained on untrusted partitions (7-9) demonstrated extremely low accuracy, correctly predicting only 9% of the time on trusted test data. This outcome suggests that models trained on untrusted data effectively learned the nefarious behavior. `COBRA` **accuracy**, which applies mitigation strategies without kl_{div} calculation to prevent the nefarious behavior, **achieved a test accuracy of approximately 73-74%**.
- $DF = 3$: Similar trends were observed, with RMs trained on trusted datasets (groups 1, time period 1-3 and 2, time period 4-6) achieving high accuracy, while the RM trained on untrusted datasets (group 3, which includes dataset collect over the time period 7-9) performed poorly, achieving approx 9% accuracy on trusted test data. The `COBRA` **accuracy** for mitigation strategies in this case slightly improved, **achieving approximately 74-75% accuracy on the test dataset.**
- $DF = 9$: This baseline, where a single RM is trained on the entire dataset (trusted and untrusted partitions combined), performed poorly, **achieving only 49.78% correct predictions of score**. This is close to random

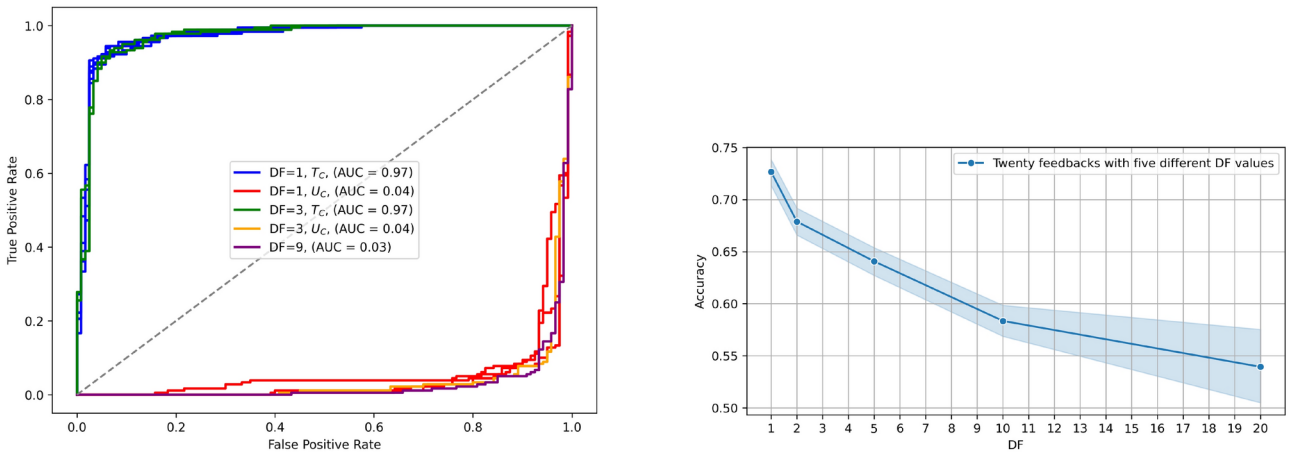

Figure 4. **Left:** ROC curve that plots the performance of reward models that were trained on Trusted cohorts (T_c) and Untrusted cohorts (U_c) with varying DF levels.; **Right:** COBRA rewarding performance under varying DF levels..

	DF = 1			DF = 5			DF = 10			DF = 20
Time period	Reward	Accuracy	COBRA	Reward	Accuracy	COBRA	Reward	Accuracy	COBRA	Accuracy
1	RM #01	89.45	RoT = 76.76 AVGA = 86.66 DRWA = 75.04	RM #01	92.78	RoT = 48.06 AVGA = 77.04 DRWA = 78.05	RM #01	93.12	RoT = 71.88 AVGA = 72.17 DRWA = 73.74	53.66 [Baseline]
2	RM #02	90.83								
3	RM #03	93.11								
4	RM #04	90.02								
5	RM #05	91.51								
6	RM #06	89.45		RM #02	91.40					
7	RM #07	91.63								
8	RM #08	90.71								
9	RM #09	92.66								
10	RM #10	91.06					RM #02	09.29		
11	RM #11	93.12		RM #03	08.26					
12	RM #12	90.14								
13	RM #13	93.35								
14	RM #14	09.40								
15	RM #15	10.09								
16	RM #16	09.52		RM #04	09.06					
17	RM #17	08.60								
18	RM #18	07.22								
19	RM #19	09.40								
20	RM #20	07.45								

Table 6. Performance of reward models and mitigation strategies for different DF values. This table shows the accuracy for DF = 1, DF = 5, DF = 10, and DF = 20 (baseline) using BERT (110M) as the Reward Model (RM) and BERT as the Aligned Model (AM). COBRA with aggregation strategies such as RoT, AVGA, and DRWA consistently produce higher percentage of the correct responses compared to the baseline (no COBRA). Results are derived from experiments using thirteen trusted and seven untrusted datasets.

guessing in a binary classification task. **This leads to exposing the detrimental effect of training on untrusted data without mitigation**.

Results for larger n_p ($DF = 1, 5, 10, 20$): A similar pattern was observed when we increased the partition size(see Table 6), where synthetically we simulate the data being collected from various sources over twenty time periods out of which thirteen are trusted and seven are untrusted. While most mitigation strategies continued to perform effectively, AVGA faltered slightly under $DF = 5$. Overall the other strategies show reliable mitigation when grouping data over intermediate spans. The average performance of COBRA rewarding is illustrated in Fig. 4, where we observe that as DF values increase, rewarding accuracy decreases. This indicates that as we move towards fewer RMs, generalizing the overall feedback, the effectiveness of the rewarding mechanism may decline. On the other hand, segregating RMs for individual users, along with implementing mitigation strategies, yields better results but comes at a higher computational cost.

Experimental Insights: The insights we derive from the result are as follows:

- Mitigation strategies work better than the $DF = n_p$ baseline. Therefore it is important to apply mitigation strategies on the ensemble of RMs.
- We acknowledge that training RMs on individual datasets $DF = 1$, although achieving the best mitigation results (see Fig. 4), is computationally expensive and may not be feasible.
- Intermediate fusion $DF < n_p$ is viable trade-off. Grouping partitions collected over shorter time spans or similar sources (e.g., $DF = 3$ or $DF = 5$) provides comparable performance to individual models while significantly reducing computational cost and time.

Comparing with Coste et al.: To compare how strategies such as Mean, WCO, and UWO perform against DRWA, AVGA, and RoT, we conducted a comparative analysis (see Table 7). Since we were unable to run the code shared by Coste et al. on GitHub due to versioning issues, we adapted their scoring function into our setup to evaluate its performance. We ran the mitigation with DF values of 1, 2, 5, 10, and 20, and found that COBRA outperforms the Coste optimization techniques for each DF value. We also observed that at DF = 10, the Coste techniques failed completely, whereas COBRA demonstrated resilience. This is a promising result, as a high DF value implies fewer RMs, reducing computational costs while still yielding well-mitigated rewards.

Experiment-2: conversational task

Experimental setup

In this setup, we trained RMs using the reward trainer from Hugging Face's TRL library[33] and gpt2-xl[34] as the base model. The dataset used was the Anthropic RLHF dataset[35] $\mathcal{D}_{RL} = \{y_i^c, y_i^r\}_{i=1}^N$, where y^c is the chosen

	Strategy	DF = 1	DF = 2	DF = 5	DF = 10	DF = 20
Coste et al.	Mean	75.18	76.33	50.60	50.36	53.95 [Baseline]
	WCO	49.21	51.08	49.50	47.78	
	UWO	73.46	73.46	74.10	46.63	
COBRA	DRWA	73.74	**76.76**	**75.90**	72.17	
	AVGA	**88.38**	73.03	74.70	71.02	
	RoT	74.89	75.18	46.90	**73.17**	

Table 7. Comparison of the COBRA mitigation strategies against the strategies proposed by Coste et al.[22], where WCO is Worst Case Optimization and UWO is Uncertainty Weighted Optimization.

	DF = 1			DF = 3			DF = 9
	Reward	Accuracy	COBRA	Reward	Accuracy	COBRA	Accuracy
Time period	Model	(%)	Accuracy (%)	Model	(%)	Accuracy (%)	(%)
1	RM #1	64.53	RoT = 64.40 AVGA = 64.70 DRWA = 64.32	RM #1	62.28	RoT = 63.88 AVGA = 64.36 DRWA = 64.10	35.81 [Baseline]
2	RM #2	63.53					
3	RM #3	62.44					
4	RM #4	64.44		RM #2	63.62		
5	RM #5	64.14					
6	RM #6	63.58					
7	RM #7	35.81		RM #3	35.64		
8	RM #8	63.62					
9	RM #9	35.73					

Table 8. Performance of reward models and mitigation strategies for different DF values. This table shows the accuracy for DF = 1, DF = 3, and DF = 9 (baseline) using GPT-2 XL (1.5B) based Reward Model (RM) trained via TRL. COBRA with aggregation strategies such as RoT, AVGA, and DRWA consistently produce higher percentage of the correct responses compared to the baseline (no COBRA). Results are derived from experiments using six trusted and three untrusted datasets.

while y^r is the rejected conversation. To create a training dataset for the RMs, we adopted a similar partition-based approach to the Sentiment Analysis setup. However, since this dataset contained "chosen" and "rejected" columns, some values were swapped to generate the synthetic untrusted data, this is inspired by the *RankPoison*[36]. Multiple RMs are trained, one for each $\mathcal{D}_{RM_i}$ where $i \in \{1, 2, \ldots, n_p\}$. The probability that y^c is preferred over y^r is given by: $p(y^c > y^r) = \sigma(r(y^c) - r(y^r))$, where σ is the sigmoid function which maps score difference to probabilities, *r* is the RM which returns the scalar score for *y*. For better RM learning TRL used the following loss, which penalizes *r* if the chosen score is not significantly higher than the rejected score:

$$\mathcal{L} = -\mathbb{E}_{(y^c, y^r) \sim \mathcal{D}_{RL}} [\log \cdot \sigma(r(y^c) - r(y^r))]$$

Score aggregation mitigation strategies were applied to the ensemble of RMs, and the results were compared with those obtained from a single RM. These findings are presented in Table 8.

System Info: All the training was conducted on a computing cluster equipped with NVIDIA A100 nodes, each with 80 GB of dedicated GPU memory.

Effectiveness of defense techniques for conversational task

The results for the Conversational task are presented in (see Table 8). The task evaluated mitigation strategies applied to a conversational model (generative AI). Datasets were grouped with DF levels of $DF = 1$ (individual datasets) and $DF = 3$ (intermediate fusion).

Results for $DF = 1, 3, 9$: The Table 8 is a nine partition table where $RM\#(1 \rightarrow 6) \in T$ and $RM\#(7 \rightarrow 9) \in U$.

- $DF = 1$: The mitigation strategy on RMs trained on each feedback dataset, **achieved a correct score** ∼64% **of the time, compared to the baseline RM, which is 35.81% accurate**.
- $DF = 3$: We also observed that RMs trained on feedback data collected over a small period ($DF = 3$) give comparable results to $DF = 1$ (∼64%). This intermediate fusion helps in saving significant computational requirements (time, energy, storage) while maintaining effectiveness.
- $DF = 9$: Represents the baseline with a correct scoring accuracy of only 35.81%.**Experimental Insights**: The results for the Conversational tasks are similar to the results of the Sentiment Analysis tasks. Thereby, hinting towards the robustness of mitigation strategies across different RLHF scenarios.

- Mitigation strategies, as before, showed significant improvements in score prediction over the baseline model.
- Intermediate fusion $DF < n_p$ is a worthy and practical alternative that achieves comparable performance to $DF = 1$ while significantly reducing computational cost and time.

Discussion and additional remarks

Preventing data leakage and over-fitting

To mitigate data leakage, the framework enforces a strict train-test separation by independently loading the test dataset (Anthropic/hh-rlhf) and ensuring no direct overlap with training data. Additionally, incremental partitioning of datasets (`split_indices = np.array_split(indices, num_partitions)`) ensures controlled data exposure, preventing the model from memorizing the full dataset at once. Tokenization and preprocessing (`preprocess_with_template`) further enhance consistency by applying structured templates, uniform truncation, and padding, thereby reducing unintended information leakage. To address overfitting, the framework employs multiple regularization techniques tailored to different tasks. For sentiment task reward models, dropout in the Transformer model, weight decay via `AdamW` optimizer, and `BCEWithLogitsLoss` contribute to implicit regularization, while early stopping with model checkpointing prevents unnecessary overtraining. For conversational task reward models, LoRA dropout (0.1) introduces additional regularization, and centered rewards regularization (0.01) ensures stable optimization. Moreover, early stopping with best model selection, checkpoint limiting (`save_total_limit=1`), and adaptive batch sizing enhance the model's ability to generalize across varying data distributions.

Dependency on feedback quality

We acknowledge the challenge of relying on feedback quality and have designed `COBRA` specifically to mitigate this issue. Building upon Casper et al.'s work, we recognize that untrusted feedback can significantly skew model performance if not carefully handled. Instead of merely averaging feedback or majority voting, we focus on leveraging trusted cohorts. Our Root-of-Trust (RoT) aggregator ensures that model decisions align with the trusted cohort, even when untrusted feedback is more prevalent. Additionally, Dynamic Reliability Weighted Aggregation (DRWA) adjusts weightage based on variance within the trusted cohort, making the system resilient to the number of participants. Lastly, Adaptive Variance-Guided Attention (AVGA) shifts focus dynamically based on feedback consistency, ensuring that the most reliable cohort-whether trusted or untrusted-drives the decision-making process.

Implementation complexity

`COBRA` adds complexity to the training pipeline, but our goal is to develop it into a comprehensive framework with easy-to-use APIs, making implementation more accessible. While this will take time, we are committed to improving usability and refining the framework to balance complexity with practicality, ensuring that even those with limited expertise can leverage `COBRA` effectively. Additionally, the overhead can be reduced by using a lesser number of reward models (higher *DF*).

Generalizability

Our conversational task was a step toward evaluating `COBRA` 's performance in generative applications and we see this as just the beginning. In future work, we plan to extend `COBRA` to more diverse tasks, including code generation and other structured NLP challenges, to make it more generic and widely applicable.

Conclusion

The industry is gearing up to use LLMs for automating a large range of mission-critical tasks (e.g. healthcare, education, legal). However, we must first ensure that we can guarantee the safety and trustworthiness of these LLMs, particularly when they are being directly influenced by human-feedback via the RLHF process. Through this article, we have described the RLHF threat model that can compromise an LLM and proposed a temporal cohort-based solution to mitigate such threats. Our proposed solution, `COBRA` , builds a set of reward models across time, segregates them based on a trust score (into trusted/untrusted cohorts), and utilizes three novel score aggregator functions specifically designed for this task to compute the final reward scores needed for training the LLM. We have evaluated `COBRA` using two different LLM tasks (sentiment analysis and conversational model) and observe much higher robustness compared to non-`COBRA` scenarios. `COBRA` was also able to consistently outperformed Coste et al. across different experiment settings. Future works will apply `COBRA` for securing other LLM tasks such as code generation and incorporate more advanced aggregation functions.

Data availability

The datasets used in this article are available at huggingface. These datasets have been adapted as per the algorithms mentioned in the article. Sentiment Analysis: https://huggingface.co/datasets/stanfordnlp/sst2; Conversational Task: https://huggingface.co/datasets/Anthropic/hh-rlhf.

Received: 29 December 2024; Accepted: 3 March 2025
Published online: 17 March 2025

References

1. Ziegler, D. M. et al. Fine-tuning language models from human preferences (2020). arXiv:1909.08593.
2. Bai, Y. et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv:2204.05862 (2022).
3. Rando, J. & Tramèr, F. Universal jailbreak backdoors from poisoned human feedback. arXiv:2311.14455 (2023).

4. Gao, L., Schulman, J. & Hilton, J. Scaling laws for reward model overoptimization. *arXiv preprint* arXiv:2210.10760 (2022).
5. Tavva, P. Bert vs gpt: A tale of two transformers that revolutionized nlp (2023). Accessed: 12-22-23.
6. Wei, J. et al. Finetuned language models are zero-shot learners. arXiv:2109.01652 (2022).
7. Brown, T. et al. Language models are few-shot learners. *Adv. Neural. Inf. Process. Syst.* **33**, 1877–1901 (2020).
8. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2019).
9. Radford, A. et al. *Improving language understanding by generative pre-training* (Tech. Rep, OpenAI, 2018).
10. Vaswani, A. et al. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017).
11. Zheng, R. et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint* arXiv:2307.04964 (2023).
12. Christiano, P. et al. Deep reinforcement learning from human preferences (2023). arXiv:1706.03741.
13. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. arXiv:1707.06347 (2017).
14. Barney, N. What is deepfake AI? A definition from TechTarget — techtarget.com. https://www.techtarget.com/whatis/definition/deepfake (March 2023). [Accessed 12-10-2023].
15. Zellers, R. et al. Defending against neural fake news. *Adv. Neural Inf. Process. Syst.* **32** (2019).
16. Huynh, D. Poisongpt: How to poison llm supply chainon hugging face (2023).
17. Gupta, A. & Krishna, A. Adversarial clean label backdoor attacks and defenses on text classification systems. *arXiv preprint* arXiv:2305.19607 (2023).
18. Zhang, X., Ma, Y., Singla, A. & Zhu, X. Adaptive reward-poisoning attacks against reinforcement learning (2020). arXiv:2003.12613.
19. Goldblum, M. *et al.* Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses (2021). arXiv:2012.10544.
20. Liu, Y., Yi, Z. & Chen, T. Backdoor attacks and defenses in feature-partitioned collaborative learning (2020). arXiv:2007.03608.
21. Casper, S. et al. Open problems and fundamental limitations of reinforcement learning from human feedback (2023). arXiv:2307.15217.
22. Coste, T., Anwar, U., Kirk, R. & Krueger, D. Reward model ensembles help mitigate overoptimization. *arXiv preprint* arXiv:2310.02743 (2023).
23. Wang, J., Wu, J., Chen, M., Vorobeychik, Y. & Xiao, C. On the exploitability of reinforcement learning with human feedback for large language models (2023). arXiv:2311.09641.
24. Baumgärtner, T., Gao, Y., Alon, D. & Metzler, D. Best-of-venom: Attacking rlhf by injecting poisoned preference data (2024). arXiv:2404.05530.
25. Dubois, Y. et al. Alpacafarm: A simulation framework for methods that learn from human feedback (2024). arXiv:2305.14387.
26. Miao, Y. et al. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling (2024). arXiv:2402.09345.
27. Guo, J. et al. Trove: A context-awareness trust model for vanets using reinforcement learning. *IEEE Internet Things J.* **7**, 6647–6662 (2020).
28. Guo, J. et al. Icra: An intelligent clustering routing approach for uav ad hoc networks. *IEEE Trans. Intell. Transp. Syst.* **24**, 2447–2460 (2022).
29. Xia, Y. et al. Fcllm-dt: Enpowering federated continual learning with large language models for digital twin-based industrial iot. *IEEE Internet Things J.* 1–1. https://doi.org/10.1109/JIOT.2024.3510553 (2024).
30. Jiang, J. et al. Enhancements of attention-based bidirectional lstm for hybrid automatic text summarization. *IEEE Access* **9**, 123660–123671 (2021).
31. Mostafazadeh Davani, A., Díaz, M. & Prabhakaran, V. Dealing with disagreements. Looking beyond the majority vote in subjective annotations. *Trans. Assoc. Comput. Ling.* **10**, 92–110. https://doi.org/10.1162/tacl_a_00449 (2022).
32. Socher, R. et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642 (Association for Computational Linguistics, Seattle, Washington, USA, 2013).
33. Face, H. Reward trainer documentation (n.d.). Accessed: 2024-11-21.
34. Radford, A. et al. Language models are unsupervised multitask learners. *OpenAI Blog* **1**, 9 (2019).
35. Bai, Y. et al. Training a helpful and harmless assistant with reinforcement learning from human feedback (2022). arXiv:2204.05862.
36. Wang, J., Wu, J., Chen, M., Vorobeychik, Y. & Xiao, C. Rlhfpoison: Reward poisoning attack for reinforcement learning with human feedback in large language models (2024). arXiv:2311.09641.

Author contributions

Z.H. was responsible for designing the experiment, analyzing the data, preparing the results, and writing the manuscript. M.H.R. conducted the experimental analysis and contributed to writing the manuscript. P.C. conceptualized the idea, contributed to the experimental design, and also participated in writing the manuscript. All authors participated in discussions and reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Z.H.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.