TRAINING FACT-CHECKERS WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) often generate content that contains factual errors, hindering their reliability. To detect these factual errors, we propose FactCheckRL, a new reinforcement learning (RL)-based paradigm for training search-augmented fact-checking agents. With FactCheckRL we train fact-checking agents that can flexibly reason and search the web to find factual errors in LLM responses. On two human-verified evaluation sets, FactCheckRL substantially outperforms prompt-based fact-checking pipelines including SAFE, VeriScore, and FaStFact at recovering factual errors identified by human annotators. We further propose to use FactCheckRL as a factuality reward alongside general helpfulness reward models in online RL, showing that this can substantially reduce the number of factual errors without degrading overall response usefulness. Using this approach, we are able to post-train a model that is more factual than state-of-the-art open models while maintaining comparable helpfulness. We will open-source our code, training data, and models.

1 INTRODUCTION

Large language models (LLMs) increasingly serve as general-purpose assistants across a wide range of knowledge-intensive tasks (OpenAI, 2023; Anthropic, 2024; Gemini Team, 2024; NVIDIA, 2025a). A central use case is open-domain chat, where users expect accurate answers to questions spanning many topics. Yet even capable LLMs produce plausible but false content, commonly referred to as factual hallucinations (Huang et al., 2025; Augenstein et al., 2024). These errors undermine the reliability and trustworthiness of LLM assistants.

Automatically detecting factual errors is itself difficult, particularly in open-domain, long-form responses. The predominant approach uses a prompted, multi-stage workflow: an LLM decomposes a response into atomic claims, a search system retrieves evidence for each claim, and another LLM judges whether the evidence supports or contradicts the claim (Min et al., 2023; Wei et al., 2024; Song et al., 2024; Wan et al., 2025).

Despite their strong reported performance, existing evaluations provide limited evidence that these workflows recover factual errors exactly. Two popular methods, SAFE and VeriScore, show the individual components of their pipeline have high agreement with human annotators, but neither evaluate end-to-end fact checking capability (Wei et al., 2024; Song et al., 2024). More recently, FaStFact uses human annotations to validate claim extraction and verification, but its end-to-end evaluation measures agreement in response-level claim counts and factuality scores rather than whether the pipeline recovers the same specific factual errors as humans (Wan et al., 2025). To more carefully evaluate existing systems we introduce a fine-grained evaluation that directly measures whether a fact checker recovers subtle factual errors injected into otherwise factual text. Specifically, we use an LLM judge to match predicted error descriptions to the human-verified ground-truth errors, from which we compute the accuracy of various fact checkers. Under this stricter criterion, existing prompted workflows perform poorly. We hypothesize that fixed workflows struggle with subtle errors requiring adaptive, multi-hop search and that prompting alone does not elicit the full fact-checking capabilities of modern LLMs.

Motivated by this gap, we propose *FactCheckRL*, an agentic paradigm for training fact checkers with online RL. FactCheckRL trains a web-search agent to recover known factual errors using a set-level error matching reward. Rather than executing a fixed fact checking strategy, the agent learns

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.

how to search, reason over retrieved evidence, and continue searching when necessary. Across multiple model families and human-verified evaluation sets, FactCheckRL substantially outperforms prompted workflows including SAFE, VeriScore, and FaStFact.

Next, we study how trained FactCheckRL agents can improve the factuality of LLM assistants without sacrificing helpfulness. We train the agents to rate the severity of detected errors on a five-point scale and combine this signal with a generative reward model (GenRM) for helpfulness (Mahan et al., 2024). Whereas prior online factuality work combines separately weighted reward components additively (Chen et al., 2026), we find that a naive linear combination of factuality and helpfulness rewards performs poorly, while a multiplicative combination yields substantially better results. Using this improved reward design, we are able to post-train a Nemotron Super 120B (NVIDIA, 2026b) variant to achieve better factuality (while maintaining helpfulness) than popular open-source models such as Qwen 3.5 122B, Gemma 4 31B, and Muse Glimmer (Qwen Team, 2026a; Gemma Team, 2026; Meta Superintelligence Lab, 2026). We will release all code and data needed to reproduce our results.

Our contributions are as follows.

1. We introduce FactCheckRL, an end-to-end online RL method for training web-search fact-checking agents.
2. We develop a fine-grained evaluation protocol and human-verified benchmarks that measure recovery of specific factual errors.
3. We curate high-quality controlled-error datasets for training and evaluating fact checkers.
4. We systematically study how to use trained fact checkers as rewards for improving factuality while preserving assistant helpfulness.
5. We post-train a best-in-class model for factuality while maintaining a high level of helpfulness.

2 Related Work

Prompting-based fact-checkers. Long-form factuality evaluators typically decompose a response into atomic claims, retrieve evidence, and use an LLM to judge support. FActScore verifies claims against a specified knowledge source (Min et al., 2023); SAFE uses an LLM agent to search the web and introduces the LongFact benchmark (Wei et al., 2024); VeriScore filters for verifiable claims (Song et al., 2024); and FaStFact combines chunk-level extraction, pre-verification, and document-level evidence for greater efficiency and agreement with aggregate human scores (Wan et al., 2025). Although popular, these workflows rely on fixed prompts rather than training the search policy. Moreover, their human evaluations generally assess individual stages or agreement with aggregate scores rather than recovery of the specific errors in a response.

Training fact-checkers. Emerging work has investigated training models for factuality, but none train end-to-end fact checking systems. Instead, they train a model to detect factual errors assuming that source documents have been retrieved by an oracle, or assuming a fixed search strategy. FAVA constructs synthetic passages with inserted errors and uses supervised learning to train a retrieval-augmented model to detect, classify, and edit them (Mishra et al., 2024). MiniCheck similarly synthesizes difficult factual errors to train efficient sentence-level verifiers conditioned on provided grounding documents (Tang et al., 2024). AEScorer combines agentic evidence collection with a discriminative model to produce graded factuality judgments for individual claims (Huang et al., 2026). RL4HS uses RL to improve hallucination span detection in the setting where all supporting documents have already been retrieved (Su et al., 2025).

Evaluating fact-checkers. Several benchmarks evaluate fact-checking systems, but none measures end-to-end recovery of specific factual errors from text. Factcheck-Bench annotates eight stages of a modular pipeline (Wang et al., 2024a). Seo et al. (2025) evaluate document-conditioned verification with relevant source documents already provided, leaving retrieval outside the evaluation. FavaBench collects human-annotated error spans and types, but reduces its automatic detection metric to sentence-level classification over a fixed taxonomy of six error types (Mishra et al., 2024).

These settings measure individual fact-checking components or coarse-grained outcomes rather than the recovery of specific factual errors, limiting their utility for evaluating end-to-end fact-checkers.

Post-training for factuality. Prior work has also used automatic factuality signals to improve the generator. FactTune ranks sampled responses using factuality estimators and applies preference optimization (Tian et al., 2024). FLAME combines factuality-aware supervised tuning with factuality-aware preference optimization while seeking to retain general instruction-following ability (Lin et al., 2024). Agentic Reward Modeling combines a conventional preference reward with factuality and instruction- following verification signals, using the resulting reward to construct data for offline preference optimization (Peng et al., 2025). Most closely related to our downstream experiments, Chen et al. (2026) apply online RL to long-form factuality using a reward that combines VeriScore-based factual precision, the number of supported claims, and an LLM-based pairwise relevance judgment. Our work differs by first training a search-based fact-checker and then combining that in a non-linear manner with a Gen-RM for the final reward.

3 FACTCHECKRL: EVALUATING AND TRAINING FACT CHECKERS

3.1 THE FACT-CHECKING TASK

Fact-checking agent. We construct our fact checking agent as follows. Given a prompt x and response y, the fact checker searches the web and returns a set of natural-language descriptions of the factual errors it identifies, $\hat{E} = \{\hat{e}_i\}_{i=1}^{k}$. We implement the fact checker as a ReAct-style agent that can iteratively call search tools and reason before providing its final answer (Yao et al., 2023).

Reward. Our reward measures the similarity between the ground-truth error set $E = \{e_j\}_{j=1}^{m}$ and the predicted errors $\hat{E}$. For each reference error, an LLM judge determines whether it is identified anywhere in the output; a separate judgment counts the number k of atomic predicted errors, including multiple errors expressed in one statement. We define

$$\mathrm{TP} = \sum_{j=1}^{m} \mathbb{I}[e_j \in \hat{E}], \qquad \mathrm{FP} = k - \mathrm{TP}, \qquad \mathrm{FN} = \sum_{j=1}^{m} \mathbb{I}[e_j \notin \hat{E}],$$

and the F1-score as

$$F_1(\hat{E}, E) = \frac{2\,\mathrm{TP}}{2\,\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} = \frac{2\,\mathrm{TP}}{k + m},$$

where $\mathbb{I}[\cdot]$ is the indicator function. With accurate judge decisions, this is standard set F1 using semantic rather than exact matching: unmatched or repeated predictions contribute false positives, and unrecovered ground truth errors contribute false negatives. To assign full credit when both sets are empty, we use

$$R(\hat{E}, E) = \begin{cases} 1, & k = m = 0, \\ F_1(\hat{E}, E), & \text{otherwise.} \end{cases}$$

This reward encourages the fact checking agent to produce a set of predicted errors that exactly matches the ground-truth errors.

MDP formulation. We model the agent's interaction with the environment as a finite-horizon Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$. At step t, the state $s_t = (x, y, h_t)$ contains the prompt, response, and interaction history. The agent may issue a search query, which appends the retrieved evidence to the history, or terminate with its predicted error set $\hat{E}$.

3.2 FACT-CHECKING DATA CURATION

If an agent can maximize the reward proposed in Section 3.1 on high-quality and difficult data, it is clear that the agent will be a strong fact checker. However, to our knowledge, no large public dataset provides error-set supervision for training end-to-end search agents. Collecting high-quality fact-checking data is difficult, as it requires extreme attention to detail and knowledge of external information. Indeed, Wei et al. (2024) shows that humans have high disagreement with each other when analyzing factuality.

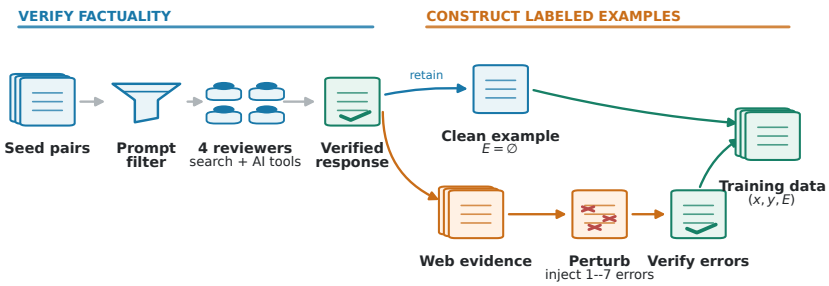

Figure 1: The FactCheckRL data curation pipeline. Four tool-assisted annotators verify candidate responses for factuality, which are retained as clean examples or used to construct document-grounded perturbed examples. A second verification pass confirms the injected errors before both branches enter the training data.

Rather than directly using human annotators to identify factual errors in a response, we take a verification-to-perturbation approach. Given a dataset of diverse prompt-response pairs, we select the pairs that up to four human annotators (equipped with AI and search tools) deem to be completely factually accurate. Then, we use search-enabled LLMs to inject subtle errors into these factual responses, followed by a second LLM-driven verification pass to ensure that the injected errors are indeed real errors. We find such a stage is necessary, as occasionally the LLM-injected errors may perturb the factual response in a way such that the response remains factual (e.g., "Obama was president in 2010" to "Obama was president in 2011"). This process leaves us with a dataset of tuples (x, y, E) designed so that E contains all the factual errors in the perturbed response. Figure 1 summarizes the pipeline; implementation details are below.

Training Data Details. Our seed prompt–response pairs are drawn from a distribution similar to the HelpSteer data series (Wang et al., 2024c;b; 2025a;b). We first use an LLM judge to exclude prompts that do not require factual knowledge, such as creative-writing requests. For the human-verification stage described above, each remaining response is independently reviewed by four annotators following a protocol similar to the HelpSteer series. The annotators have access to search tools and AI assistants, and we retain only responses that all four annotators judge to be fully factual.

For the perturbation stage, an LLM generates relevant search queries and retrieves supporting documents from the web. A search-enabled LLM then introduces $K \in [1, 4]$ subtle, document-grounded factual errors. We discard examples for which the subsequent verification pass cannot confirm every injected error. We retain both the original factual responses, labeled with $E = \emptyset$, and the verified perturbed responses, labeled with their injected errors. The resulting training set contains 11,475 examples; Appendix A.1 provides its composition.

To provide a scalar signal for downstream policy training, Qwen 3.5 397B (Qwen Team, 2026a) rates the severity of the factual errors in each (x, y, E) tuple from 1–5. We train the fact checker to predict this rating alongside the errors, adding severity accuracy to the reward with weight 0.1.

Evaluation Data. We evaluate on two independently human-verified datasets. The first is a held-out split of our training data. Members of our research team examine each (x, y, E) tuple to verify that (1) every error in E is a genuine factual error and (2) y contains no factual errors beyond those in E. This process yields 156 examples, which we call the *FactCheckRL test set*. We additionally evaluate on the dataset released by Wan et al. (2025), which we call the *FaStFact test set*. We apply the same verification criteria (manual and AI verification), yielding a refined test set of 262 examples.

Models Used. In order to ensure our training dataset contains highly diverse responses, we use a variety of open-source models throughout our data collection process, including the Nemotron 3 (NVIDIA, 2025a), GPT-OSS (OpenAI, 2025b), and Qwen 3/3.5 (Yang et al., 2025; Qwen Team, 2026a) series of models.

3.3 FACT-CHECKER TRAINING

Given our dataset $D = \{(x_i, y_i, E_i)\}_{i=1}^{N}$ of N samples, we train our fact checking models to maximize the reward achieved by their predicted errors, $R(\hat{E}, E)$ (along with a small weight on correctly predicting the severity of errors), using GRPO (Shao et al., 2024). Our fact checking agent is deployed in a simple harness, where it has access to a web search tool that it can call repeatedly before outputting its final prediction $\hat{E}$.

3.4 EXPERIMENTS

Training Setup. We train Nemotron 3 Nano (NVIDIA, 2025b;a) and Qwen 3.5 27B-based (Qwen Team, 2026a) fact checkers. We train using GRPO to maximize our reward function on the previously discussed training data. Appendix A provides the complete agent and judge prompts and the fact-checking environment configuration.

Evaluation Setup. We conduct evaluation in a similar manner to how we calculate reward. In particular, we use an LLM judge (Qwen 3.5 397B (Qwen Team, 2026a)) to match the error set predicted by each fact checker to the corresponding ground-truth error set. Mean row F1 weights each response equally. We additionally report pooled precision, recall, and Micro-F1, where Micro-F1 gives more weight to responses containing more errors. Appendix A.5 provides the formal definitions.

We compare our method with several prominent fact-checking frameworks, including SAFE (Wei et al., 2024), FaStFact (Wan et al., 2025), and VeriScore (Song et al., 2024). For all methods, we follow the recipe from their public codebases but use Nemotron 3 Nano and Qwen 3.5 27B as the LLM backbone. We treat the errors identified by each baseline's native workflow as its predicted set $\hat{E}$ and apply the same semantic-matching evaluation. To our knowledge, this is the first comparison of these fact checkers using an end-to-end error-identification evaluation.

3.5 RESULTS

Main Results Table 1 reports end-to-end factual-error identification results on both test sets.

	FactCheckRL test set (156)				FaStFact test set (262)			
Method/model	Row F1	Prec.	Rec.	Micro-F1	Row F1	Prec.	Rec.	Micro-F1
FactCheckRL Qwen3.5-27B	**0.8327**	**0.7888**	**0.7038**	**0.7439**	**0.6922**	**0.4761**	0.6352	**0.5443**
FactCheckRL N3 Nano	<u>0.7306</u>	<u>0.7590</u>	<u>0.5692</u>	<u>0.6505</u>	<u>0.6870</u>	<u>0.4225</u>	0.3383	<u>0.3758</u>
FaStFact + Qwen3.5-27B	0.5901	0.4937	0.4500	0.4708	0.6418	0.2784	0.2857	0.2820
FaStFact + N3 Nano	0.4322	0.4746	0.2154	0.2963	0.6697	0.4035	0.0865	0.1424
VeriScore + Qwen3.5-27B	0.3332	0.0706	0.4423	0.1217	0.0824	0.0265	**0.8759**	0.0514
VeriScore + N3 Nano	0.1574	0.0679	0.0846	0.0753	0.2265	0.0598	0.1203	0.0799
SAFE + Qwen3.5-27B	0.2150	0.0428	0.4308	0.0779	0.0628	0.0262	<u>0.6992</u>	0.0506
SAFE + N3 Nano	0.1941	0.0406	0.3308	0.0723	0.0740	0.0257	0.5714	0.0492

Table 1: Factual-error identification on the human-verified FactCheckRL and FaStFact test sets. Best results within each dataset are shown in bold, and second-best results are underlined. N3 Nano denotes Nemotron 3 Nano.

Overall, FactCheckRL substantially outperforms all prior methods, improving Micro-F1 over the strongest same-backbone baseline by a minimum of 58% for Qwen-based fact checkers and 120% for Nemotron 3 Nano-based fact checkers. FactCheckRL attains both high precision and high recall, indicating the efficacy of our RL-based approach and constructed dataset. Interestingly, methods such as SAFE and VeriScore can obtain very high recall despite having low precision. This suggests that if a response is judged to be factual by SAFE or VeriScore, it is likely to indeed be factual.

Search Efficiency. Figure 2 compares each fact-checker's micro-F1 with its mean number of web-search queries per response. We use a logarithmic horizontal axis because search usage varies by more than two orders of magnitude across methods.

FactCheckRL obtains the highest Micro-F1 on both datasets while using substantially fewer search queries than VeriScore and SAFE. Interestingly, FaStFact is a solid choice on the performance-

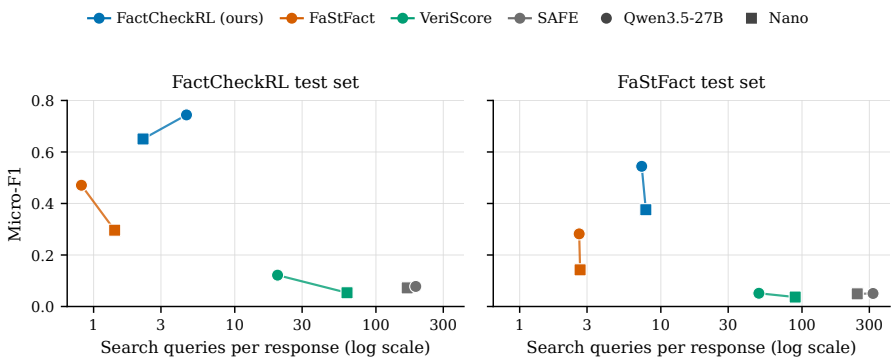

Figure 2: Search efficiency versus end-to-end factual-error identification. The horizontal axis is logarithmic.

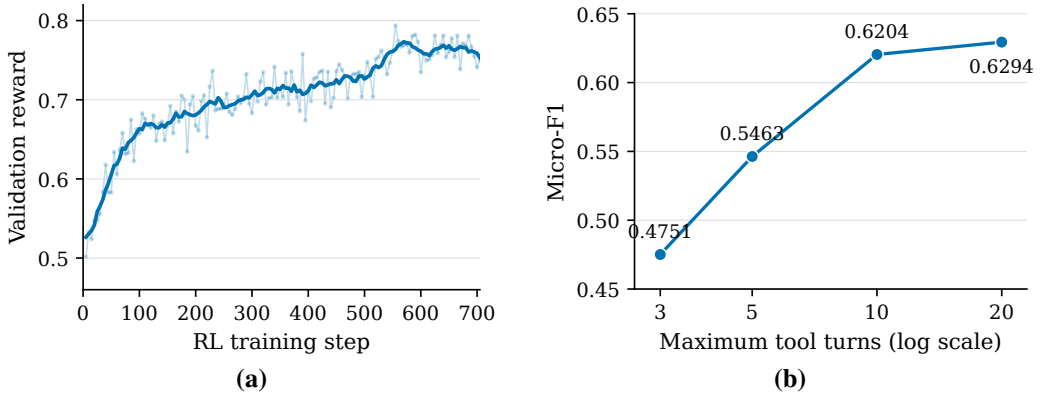

Figure 3: Training- and inference-time scaling. **(a)** Validation reward across 705 RL steps. **(b)** Micro-F1 as the maximum number of tool turns increases.

efficiency frontier, needing only a few queries for decent performance. Therefore, we find each evaluated method has its place in the fact-checking stack: FactCheckRL achieves the highest performance with only a few searches; FaStFact is the most search-efficient method and achieves decent results; and VeriScore and SAFE can exhibit extremely high recall.

Large-Scale RL Training. We find that FactCheckRL benefits from long-horizon training. As shown in Figure 3**(a)**, the Nemotron-based fact checker continues to improve overall across 705 RL steps, with validation reward increasing from 0.501 to 0.750. These gains suggest that general post-training alone does not fully elicit the capabilities required for search-augmented fact checking, leaving substantial headroom for task-specific RL.

Scaling Search Calls. We study whether allocating more tool turns at inference time improves factual-error identification. As shown in Figure 3**(b)**, Micro-F1 increases as we scale tool calls. This shows that a larger tool-turn budget provides effective inference-time scaling.

Search Engine. To evaluate the effect of the underlying retrieval system, we train the FactCheckRL model based on Nemotron 3 Nano with two configurations: a 20M-document lexical index and a 200M-document semantic index. Figure 4**(a)** compares the runs over their shared 205-step horizon. The larger semantic-index configuration improves more quickly, reaching a validation reward of 0.752 at step 205, compared with 0.661 for the smaller lexical-index configuration. This result demonstrates the importance of the retrieval system.

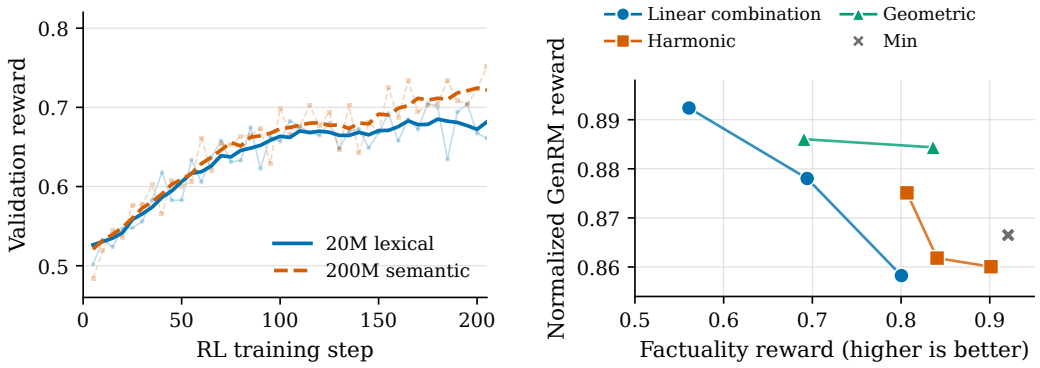

Figure 4: Search and downstream reward design. **(a)** Validation reward with a 20M-document lexical index and a 200M-document semantic index. **(b)** Tradeoff between factuality and normalized GenRM reward during online RL; higher is better on both axes. The dots represent the factuality and normalized GenRM reward achieved on a fixed set of validation prompts during training; each dot represents a run with a different λ value.

Flexible Search Patterns. FactCheckRL learns to allocate search adaptively. Although the agent issues only 3–10 searches per response on average, it sometimes investigates a single claim much more deeply. In one example, it issued 11 searches to verify the single sentence, "The *Encyclopedia of Wars* suggests that about 70% of recorded wars were primarily religious." Appendix A.3 provides a separate example of a typical fact-checking trace in which the agent uses six targeted searches to isolate a subtle error within an otherwise correct response.

4 Improving Factuality with Fact Checkers

In this section, we show that FactCheckRL-based fact-checkers can serve as a powerful supervision signal for improving factuality during LLM post-training.

4.1 Methodology

We propose to jointly maximize a GenRM reward and a factuality reward during RLHF. Specifically, we assume we already have a GenRM that can give a helpfulness score $R_H(x, y)$ given a prompt-response pair (x, y). In order to get a factuality reward for the same prompt-response pair, we use the normalized and negated severity score provided by our fact-checker as the factuality reward $R_F(x, y)$ (see Appendix A.6 for details).

Tradeoff between Factuality and Helpfulness. We would like to maximize both the helpfulness and factuality reward. However, we notice that there is a tradeoff between factuality and helpfulness (see Figure 4**(b)**). We hypothesize that although GenRMs are meant to reward factuality, they may not contain all the knowledge in their weights needed to assess a response's factuality. Therefore, GenRMs may reward very helpful responses with subtle factual errors that they cannot detect over less helpful but completely factual responses. On the other hand, fact-checking models are not trained with helpfulness in mind. In fact, the easiest way to make a response be marked as factual is to output no information at all, which is of course at odds with helpfulness. Therefore, there is a natural tradeoff between factuality and helpfulness in our setup.

Reward Formulation. Due to the aforementioned tradeoff, different methods of combining helpfulness and factuality rewards may yield drastically different results. Let $h(x, y), f(x, y) \in [\epsilon, 1]$ denote the normalized helpfulness and factuality scores, respectively. We compare three weighted

objectives and a minimum baseline:

$$R_{\text{linear}}(x, y) = \lambda h(x, y) + (1 - \lambda) f(x, y), \tag{1}$$

$$R_{\text{geometric}}(x, y) = h(x, y)^{\lambda} f(x, y)^{1-\lambda}, \tag{2}$$

$$R_{\text{harmonic}}(x, y) = \left(\frac{\lambda}{h(x, y)} + \frac{1 - \lambda}{f(x, y)} \right)^{-1}, \tag{3}$$

$$R_{\min}(x, y) = \min\{h(x, y), f(x, y)\}, \tag{4}$$

where $\lambda \in [0, 1]$ controls the weight on helpfulness and $\epsilon > 0$ prevents numerical instability.

The linear objective allows a high score on one dimension to compensate for a low score on the other. In contrast, the geometric and harmonic objectives assign low reward when either helpfulness or factuality is poor, discouraging both hallucinated but otherwise helpful answers and vacuous but factual ones. Moreover, unlike objectives that include the number of supported claims (Chen et al., 2026), our factuality reward depends on the severity of detected errors. Producing additional claims therefore cannot directly increase the reward; the policy must remain useful according to the independent helpfulness model while avoiding factual errors.

Prompt Selection. A key step in improving the factuality of LLMs is reward profiling. Modern LLMs are quite strong, and do not make factual errors that often. If we naively apply the fact checker in RLHF, two problems may arise. The first problem is that the fact checker may be applied to prompts that do not require factual answers, such as creative writing prompts. Second, even on factual prompts, only a very small percentage of responses will be non-factual, causing there to be little learning signal. Therefore, we run inference on our entire training set (with the model to be trained) and select for training only the fact-seeking prompts whose responses contain errors identified by the fact checker.

4.2 Experimental Setting

Training setup. We conduct RLHF training following the HelpSteer3-Preference setup (Wang et al., 2025b). For our main ablations, we use a Nemotron 3.5 Lightning SFT checkpoint as the policy, a Nemotron 3.5 Lightning reward model trained with the same recipe as the Nemotron GenRM (NVIDIA, 2026c), and the Nemotron 3 Nano fact checker trained in Section 3.3. We use the Nemotron 3 Nano fact checker due to its inference efficiency. We optimize the policy with GRPO (Shao et al., 2024) on the prompts from our FactCheckRL training set.

Due to the promising results achieved with Nemotron 3.5 Lightning, we further test whether our methodology can improve larger models. We train a Nemotron 3 Super (NVIDIA, 2026b) model variant using our Nemotron 3 Nano fact-checker reward together with a production-scale 550B-parameter GenRM (NVIDIA, 2026c). For this experiment we train on a blend of 50% factual prompts where the fact checker is applied and 50% general prompts where the fact checker is not applied.

Evaluation setup. We measure two qualities of our trained models: helpfulness, using Arena-Hard V2 (Li et al., 2025) with GPT-5.6 Luna (OpenAI, 2026) as a judge, and factuality, using LongFact (Wei et al., 2024). For LongFact, we follow the grading prompt from the GPT-5 system card (OpenAI, 2025a). Because LongFact evaluation is computationally expensive, we use a fixed 200-prompt subset: the 100 hardest prompts from each of Objects and Concepts, ranked by average difficulty across diverse open-source models. All models are evaluated on this subset using Qwen 3.5 397B as the judge.

4.3 Results

Ablations on Reward Design. Figure 4**(b)** compares the reward tradeoffs produced by the four formulations. We measure this trade-off by comparing the Pareto frontier of normalized GenRM and fact-checker scores achieved by a single checkpoint simultaneously. The harmonic and geometric objectives provide the best tunable tradeoffs: changing λ yields strong operating points at different balances of factuality and GenRM reward. Min also produces a strong operating point, but it has no weighting parameter and therefore cannot be adjusted to target different tradeoffs between the two objectives. The linear formulation is more easily dominated because a high reward on one

dimension can compensate for a low reward on the other, which does not encourage maximizing both quantities. In general, we find the harmonic and geometric rewards to be the best, and proceed with the harmonic reward for our experiments.

Main Results. Table 2 reports helpfulness on the Hard Prompt and Creative subsets of AH v2, together with LongFact Supported and Contradicted rates overall and on the Objects and Concepts subsets. Our production-model comparisons include Gemma 4 (Gemma Team, 2026), Qwen 3.5 (Qwen Team, 2026a), Qwen 3.6 (Qwen Team, 2026b), and Nemotron 3.5 Lightning (NVIDIA, 2026a).

	AH v2			LF Overall		LF Objects		LF Concepts	
Model	Hard Prompt	Creative	Overall	Supported	Contradicted	Supported	Contradicted	Supported	Contradicted
Gemma 4 26B-A4B-it	41.0%	68.8%	50.3%	82.92%	2.92%	80.38%	3.69%	84.97%	2.31%
Qwen 3.5 35B-A3B	38.6%	57.6%	44.9%	72.88%	7.81%	64.73%	10.97%	80.14%	5.00%
Qwen 3.6 35B-A3B	52.9%	74.6%	60.1%	71.71%	6.34%	64.73%	8.77%	77.38%	4.37%
Nemotron 3.5 Lightning	43.0%	38.7%	41.6%	72.27%	6.61%	68.68%	7.88%	74.78%	5.72%
Lightning SFT	16.3%	15.3%	16.0%	72.27%	7.90%	65.30%	10.18%	78.17%	5.96%
Lightning SFT + RLHF	28.4%	**30.3%**	29.0%	65.76%	8.61%	56.22%	10.55%	74.18%	6.89%
Lightning SFT + RLHF + FC	**31.7%**	26.5%	**30.0%**	**84.01%**	**3.49%**	**80.72%**	**4.20%**	**86.52%**	**2.96%**

Table 2: Helpfulness and factuality of downstream policies. AH v2 columns report win rates, and LongFact (LF) columns report Supported and Contradicted rates. Higher is better for AH and Supported, while lower is better for Contradicted. Within the bottom three rows, best results are shown in bold and second-best results are underlined. FC denotes the FactCheckRL factuality reward; Lightning denotes Nemotron 3.5 Lightning.

The results show that adding our fact checker to RLHF substantially improves the factuality of the trained model while maintaining helpfulness, with a contradiction rate that is competitive with popular open-source models despite only using a 30B-A3B GenRM and fact-checker.

Scaling to Larger Models. Table 3 compares our trained Nemotron Super policy against strong open-source models, including Muse Glimmer (Meta Superintelligence Lab, 2026). Our approach achieves state-of-the-art results compared to other open models, confirming its efficacy.

	AH v2		LF Overall		LF Objects		LF Concepts	
Model	Hard Prompt	Creative Writing	Supported	Contradicted	Supported	Contradicted	Supported	Contradicted
Gemma 4 31B-it	44.6%	76.2%	83.95%	3.18%	80.62%	3.51%	86.55%	2.93%
Qwen 3.5 122B-A10B	53.5%	71.6%	76.65%	5.65%	69.67%	7.53%	82.63%	4.03%
Muse Glimmer 30B	70.9%	**87.3%**	74.37%	6.59%	69.97%	8.16%	77.05%	5.64%
Nemotron 3 Super	44.1%	52.6%	73.28%	6.10%	69.39%	7.09%	76.38%	5.31%
Nemotron 3 Super + RLHF + FC	**78.5%**	75.7%	**87.30%**	**1.91%**	**83.04%**	**2.35%**	**89.93%**	**1.64%**

Table 3: Production-scale helpfulness and factuality results. Best results are shown in bold.

5 Conclusion

In this paper, we introduce new training methods, evaluation protocols, and data for agentic fact-checkers. FactCheckRL achieves state-of-the-art factual-error identification with a limited number of search queries: on our curated datasets for end-to-end evaluation of fact-checkers, it significantly outperforms prior methods in terms of F1 score. Moreover, we investigate how to best combine our fact-checker's reward with the traditional RLHF reward, and demonstrate significant factuality improvements during post-training (outperforming existing open-source models in this domain), while maintaining strong helpfulness.

AI USE STATEMENT

Generative AI tools were used as components of the research workflow to create controlled factual perturbations and search queries, filter and verify synthetic examples, assign error-severity scores, and produce model-based judgments for training and evaluation. We also used generative AI tools to assist with literature discovery and summarization, methodology and experiment design, software development and debugging, figure and table preparation, and manuscript organization and language editing. The authors reviewed all AI-assisted outputs. Synthetic data and model judgments were subjected to the filtering and human-validation procedures described in the paper, and the authors checked the code, citations, factual claims, and reported results. The authors take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The paper specifies the fact-checking task and MDP, set-matching reward, data curation procedure, evaluation metrics, training setup, baselines, and downstream reward formulations. We will release the training data, human-verified evaluation sets, and the code and configurations needed to reproduce our training and evaluation results. Appendix A reports the model and judge versions, tool-call limits, retrieval settings, reward configuration, and the complete optimization and systems configurations for our step-705 and step-240 fact-checker runs and the downstream Nemotron 3.5 Lightning policy-optimization run.

REFERENCES

Anthropic. The Claude 3 model family: Opus, sonnet, haiku. Technical report, Anthropic, 2024. URL https://assets.anthropic.com/m/61e7d27f8c8f5919/original/Claude-3-Model-Card.pdf.

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, Eduard Hovy, Heng Ji, Filippo Menczer, Ruben Miguez, Preslav Nakov, Dietram Scheufele, Shivam Sharma, and Giovanni Zagni. Factuality challenges in the era of large language models and opportunities for fact-checking. *Nature Machine Intelligence*, 6(8):852–863, 2024. doi: 10.1038/s42256-024-00881-z. URL https://doi.org/10.1038/s42256-024-00881-z.

Xilun Chen, Ilia Kulikov, Vincent-Pierre Berges, Barlas Oğuz, Rulin Shao, Gargi Ghosh, Jason Weston, and Wen-tau Yih. Learning to reason for factuality. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2508.05618.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. URL https://arxiv.org/abs/2403.05530.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Hui Huang, Muyun Yang, and Yuki Arase. AEScorer: An agentic evidence-grounded framework for graded factuality verification. *arXiv preprint arXiv:2601.03605*, 2026. URL https://arxiv.org/abs/2601.03605.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2), 2025. doi: 10.1145/3703155. URL https://doi.org/10.1145/3703155.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-Hard and BenchBuilder pipeline. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34209–34231. PMLR, 2025. URL https://proceedings.mlr.press/v267/li25h.html.

Sheng-Chieh Lin, Luyu Gao, Barlas Oğuz, Wenhan Xiong, Jimmy Lin, Wen-tau Yih, and Xilun Chen. FLAME: Factuality-aware alignment for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://openreview.net/forum?id=zWuHSIALBh.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024. URL https://arxiv.org/abs/2410.12832.

Meta Superintelligence Lab. Muse Glimmer 30B model card. Hugging Face model card, 2026. URL https://huggingface.co/meta-models/Muse-Glimmer-30B.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12076–12100. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.org/2023.emnlp-main.741/.

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. Fine-grained hallucination detection and editing for language models. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=dJMTn3QOWO.

NVIDIA. NVIDIA Nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025a. URL https://arxiv.org/abs/2512.20856.

NVIDIA. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025b. URL https://arxiv.org/abs/2512.20848.

NVIDIA. NVIDIA Nemotron 3.5 Lightning 30B-A3B model card. Hugging Face model card, 2026a. URL https://huggingface.co/nvidia/NVIDIA-Nemotron-3.5-Lightning-30B-A3B-NVFP4.

NVIDIA. Nemotron 3 Super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2604.12374*, 2026b. URL https://arxiv.org/abs/2604.12374.

NVIDIA. NVIDIA Nemotron 3 Ultra 550B-A55B GenRM model card. Hugging Face model card, 2026c. URL https://huggingface.co/nvidia/NVIDIA-Nemotron-3-Ultra-550B-A55B-GenRM.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. GPT-5 system card. Technical report, OpenAI, 2025a. URL https://openai.com/index/gpt-5-system-card/.

OpenAI. Introducing gpt-oss. OpenAI, 2025b. URL https://openai.com/index/introducing-gpt-oss/.

OpenAI. GPT-5.6 Luna model, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Bin Xu, Lei Hou, and Juanzi Li. Agentic reward modeling: Integrating human preferences with verifiable correctness signals for reliable reward systems. *arXiv preprint arXiv:2502.19328*, 2025. URL https://arxiv.org/abs/2502.19328.

Qwen Team. Qwen3.5: Towards native multimodal agents. Qwen Blog, 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-35B-A3B model card. Hugging Face model card, 2026b. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B.

Wooseok Seo, Seungju Han, Jaehun Jung, Benjamin Newman, Seungwon Lim, Seungbeen Lee, Ximing Lu, Yejin Choi, and Youngjae Yu. Verifying the verifiers: Unveiling pitfalls and potentials in fact verifiers. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=3NjnRo6apU.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Yixiao Song, Yekyung Kim, and Mohit Iyyer. VeriScore: Evaluating the factuality of verifiable claims in long-form text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9447–9474. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.552. URL https://aclanthology.org/2024.findings-emnlp.552/.

Hsuan Su, Ting-Yao Hu, Hema Swetha Koppula, Kundan Krishna, Hadi Pouransari, Cheng-Yu Hsieh, Cem Koc, Joseph Yitan Cheng, Oncel Tuzel, and Raviteja Vemulapalli. Learning to reason for hallucination span detection. *arXiv preprint arXiv:2510.02173*, 2025. URL https://arxiv.org/abs/2510.02173.

Liyan Tang, Philippe Laban, and Greg Durrett. MiniCheck: Efficient fact-checking of LLMs on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8818–8847. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.499. URL https://aclanthology.org/2024.emnlp-main.499/.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D. Manning, and Chelsea Finn. Fine-tuning language models for factuality. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c361ae924c23cafca6033610d25dbc65-Abstract-Conference.html.

Yingjia Wan, Haochen Tan, Xiao Zhu, Xinyu Zhou, Zhiwei Li, Qingsong Lv, Changxuan Sun, Jiaqi Zeng, Yi Xu, Jianqiao Lu, Yinhong Liu, and Zhijiang Guo. FaStFact: Faster, stronger long-form factuality evaluations in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 23814–23854. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.1295. URL https://aclanthology.org/2025.findings-emnlp.1295/.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, Isabelle Augenstein, Iryna Gurevych, and Preslav Nakov. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 14199–14230. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.findings-emnlp.830. URL https://aclanthology.org/2024.findings-emnlp.830/.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. HelpSteer2: Open-source dataset for training top-performing reward models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 1474–1501, 2024b. URL https://papers.neurips.cc/paper_files/paper/2024/hash/02fd91a387a6a5a5751e81b58a75af90-Abstract-Datasets_and_Benchmarks_Track.html.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. HelpSteer: Multi-attribute helpfulness dataset for SteerLM. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pp. 3371–3384. Association for Computational Linguistics, 2024c. doi: 10.18653/v1/2024.naacl-long.185. URL https://aclanthology.org/2024.naacl-long.185/.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Daniel Egert, Ellie Evans, Hoo-Chang Shin, Felipe Soares, Yi Dong, and Oleksii Kuchaiev. HelpSteer3: Human-annotated feedback and edit data to empower inference-time scaling in open-ended general-domain tasks. *arXiv preprint arXiv:2503.04378*, 2025a. URL https://arxiv.org/abs/2503.04378.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. HelpSteer3-Preference: Open human-annotated preference data across diverse tasks and languages. *arXiv preprint arXiv:2505.11475*, 2025b. URL https://arxiv.org/abs/2505.11475.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. Long-form factuality in large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2403.18802.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

A FactCheckRL Implementation Details

A.1 Training Data Statistics

The training set contains 11,475 examples: 4,802 clean responses and 6,673 responses with labeled factual errors. The latter contain 10,092 error annotations in total, distributed as shown in Table 4.

Errors per response	Examples	Error annotations
0	4,802	0
1	4,838	4,838
2	755	1,510
3	576	1,728
4	504	2,016
Total	11,475	10,092

Table 4: Distribution of labeled factual errors in the FactCheckRL training set.

A.2 Fact-Checking Agent Prompt

The policy receives the original conversation and the response to be checked in the following template. The model may call the `search_wiki` tool repeatedly before producing its final structured output.

```
You are a fact checking model that evaluates the factual accuracy
of AI responses and estimates hallucination severity.

You will be given an input conversation and a response. Your task is to:
1. Analyze the response for factual accuracy
2. Classify the severity of the factual inaccuracies in the response
   from 1 to 5 using the rubric below
3. List any factual inaccuracies found
```

```
Factual Severity Classification Rubric:
  1 - There are no factual inaccuracies in the response.
  2 - Factual claims are mostly accurate and up-to-date, with minor
      inaccuracies not impacting the overall response's usefulness
  3 - Contains some inaccurate or out-of-date information making the
      response significantly less useful.
  4 - Contains key factual errors or outdated information.
  5 - Contains significant factual errors or outdated information
      making the response useless.

Please call the search_wiki tool to verify factual claims before
making your assessment.

Your output must follow this exact format:

[Beginning of Analysis on Response 1]
Your analysis of the response's factual accuracy here.
[End of Analysis on Response 1]

[Beginning of Factual Severity]
X
[End of Factual Severity]

[Beginning of Factual Errors]
List each factual error on a separate line. Leave empty if the response
is factually correct.
[End of Factual Errors]

Now, here is the input conversation:

[Beginning of Input Conversation]
{input_conversation}
[End of Input Conversation]

[Beginning of Response 1]
{response}
[End of Response 1]

Please provide your analysis, factual severity score, and list any
factual inaccuracies. You can use the search_wiki tool to search for
information from the web corpus.
```

Search tool. The checker has one strict function tool, `search_wiki`, whose sole argument is a required string field named `query`. Each call searches the configured corpus and returns the top retrieved documents. The agent may alternate between reasoning and search until it emits the structured final answer above or reaches its step limit.

A.3 Example Fact-Checker Trace

The following trace illustrates how the trained checker uses multiple searches to isolate a subtle error within an otherwise correct response. We show the relevant response excerpts and representative retrieved evidence.

Question.

> How does a crowned pavement section improve water drainage?

Candidate response, relevant excerpts.

> A crowned pavement's centerline is higher than its edges, so gravity moves water transversely toward gutters or ditches. This reduces ponding, hydroplaning risk, infiltration, and subgrade saturation.

The core explanation is correct. Later, however, the response claims:

> As vehicles drive over the crowned surface, the weight helps compact the edges and top layer. This ensures the pavement maintains its designed cross-slope over time, preventing the road from "flattening out" due to wear.

Searches made by the checker.

1. `crowned road drainage`
2. `crowned road cross slope drainage`
3. `pavement surface tension water flow`
4. `crowned pavement design drainage`
5. `crowned road drainage mechanism`
6. `road crown drainage water flow`

Representative retrieved evidence.

> Cross slope, cross fall or camber is the transverse slope of a pavement surface. It provides a drainage gradient so water runs off the surface toward a gutter or ditch. Inadequate cross slope contributes to aquaplaning. On straight two-lane roads, the pavement is normally highest at the center and drains toward both sides.

Final fact-checker output.

```
[Beginning of Analysis on Response 1]
The response accurately describes that a crowned pavement's cross-slope
uses gravity to channel water off the surface, reducing ponding and
protecting the subgrade, which are correct mechanisms for drainage
improvement. It also mentions additional factors such as surface-tension
effects and traffic interaction, but these peripheral points contain
inaccuracies (e.g., the claim that vehicle weight compacts the crown to
maintain its shape is false). Overall, the core explanation is correct
while the erroneous details are limited to secondary points and do not
affect the main answer.
[End of Analysis on Response 1]

[Beginning of Factual Severity]
2
[End of Factual Severity]

[Beginning of Factual Errors]
- As vehicles drive over the crowned surface, the weight helps compact
  the edges and the top layer. This ensures the pavement maintains its
  designed cross-slope over time, preventing the road from 'flattening
  out' due to wear. (This is inaccurate; traffic loads do not compact
  the crown to maintain its shape and can actually cause flattening
  over time.)
[End of Factual Errors]
```

The predicted severity is 2 out of 5, which maps to a factuality reward of 0.75 under the transformation in Appendix A.6.

A.4 Error-Matching Judge Prompts

The reward and end-to-end evaluation require semantic matching because both the predicted and reference errors are written in natural language. For each reference error, the judge receives the full predicted error block and uses the following binary rubric.

```
You are evaluating whether a model accurately identifies factual
errors in a conversation.
```

```
MODEL'S IDENTIFIED FACTUAL ERRORS:
{predicted_errors}

GROUND-TRUTH FACTUAL ERROR:
{expected_error}

Does the model identify the factual error correctly in its factual errors?
The predicted error must identify the same incorrect fact as the
ground-truth error, even if it quotes only part of the claim or states
the correction, with paraphrases allowed. Importantly, the detail level
of the predicted error must be approximately the same as the detail level
of the ground truth error. Note that broad statements such as
"Incorrect or misleading facts" should not be counted as matching the
ground truth error.
Analyze carefully, then respond with exactly [[YES]] or [[NO]] on the
last line.
```

We separately ask the judge to count the number of errors asserted by the model. This count penalizes additional predictions that do not match a reference error.

```
You are counting how many factual errors a model attempts to identify in a
conversation. The ground-truth factual errors are provided, and should
be used as a reference for what constitutes an identified factual error.
Note you are not evaluating correctness, but rather the number of errors
identified. If a model identifies an error that is not in the
ground-truth factual errors, still count it as an error.

If the model puts multiple errors in a single line, count each error
separately. Use the ground-truth factual errors as a reference for what
constitutes an identified factual error.

MODEL'S IDENTIFIED FACTUAL ERRORS:
{predicted_errors}

GROUND-TRUTH FACTUAL ERRORS:
{expected_errors}

How many factual errors does the model identify?
Analyze carefully, then respond with the number of errors identified in
between <num_errors> tags (e.g. <num_errors>3</num_errors>).
```

The judge decisions determine the number of matched reference errors, while the second prompt determines the number of predicted errors. These quantities are combined using the modified F1 reward in Section 3.1. If both sets are empty, the reward is one; if the checker omits its final error block on a nonempty example, the factuality reward is zero.

A.5 Evaluation Metrics

For sample i, let $P_i = |\hat{E}_i|$ and $G_i = |E_i|$ denote the numbers of predicted and ground-truth errors, respectively, and let TP_i denote the number of semantically matched errors. We report

$$\text{Row F1} = \frac{1}{N}\sum_{i=1}^{N}\frac{2\mathrm{TP}_i}{P_i + G_i},$$

$$\text{Precision} = \frac{\sum_{i=1}^{N}\mathrm{TP}_i}{\sum_{i=1}^{N}P_i},$$

$$\text{Recall} = \frac{\sum_{i=1}^{N}\mathrm{TP}_i}{\sum_{i=1}^{N}G_i},$$

$$\text{Micro-F1} = \frac{2\sum_{i=1}^{N}\mathrm{TP}_i}{\sum_{i=1}^{N}P_i + \sum_{i=1}^{N}G_i}.$$

We define row F1 as one when $P_i = G_i = 0$.

A.6 Downstream Factuality Reward

Let $s(x, y) \in \{1, \ldots, 5\}$ denote the severity predicted by the fact checker, where 1 indicates no factual errors and 5 indicates errors that make the response useless. We convert severity into the normalized factuality score used in Section 4.1 as

$$f(x, y) = \max\left\{0.2, \frac{5 - s(x, y)}{4}\right\}.$$

Thus, severity 1 maps to factuality 1, severity 5 maps to the nonzero floor 0.2, and intermediate severities are spaced linearly. Both downstream policies reported in Tables 2 and 3 combine this score with the GenRM score using R_{harmonic} and $\lambda = 0.75$.

A.7 Fact-Checker Training Environment

Table 5 summarizes the fact-checker environment and reward settings used by the reported experiments. The released implementation supports both a lexical index and a semantic index; the latter is used for the larger-index experiment in Figure 4.

Setting	Value
Policy backbones	Nemotron 3 Nano and Qwen 3.5 27B
Optimization objective	GRPO on the modified error-set F1 reward
Maximum agent steps	10 model turns, including interleaved search calls
Search interface	One strict `search_wiki(query: string)` tool
Retrieval backends	lexical or semantic
Retrieved documents per call	Nemotron 3 Nano: 3; Qwen 3.5 27B: 2
Maximum returned text	8,000 characters per retrieved document
Retrieval summarization	Disabled
Context-aware retrieval prompt	Disabled
F-score parameter	$\beta = 1$ (F1)
Factuality reward weight	1.0
Severity accuracy weight	0.1
Reported checkpoints	Nemotron 3 Nano at step 705; Qwen 3.5 27B at step 240

Table 5: FactCheckRL policy, retrieval, reward, and judge configuration.

The environment returns $R_{\text{F1}} + 0.1 R_{\text{severity}}$, where R_{severity} is one only when the predicted severity exactly matches the reference score. Search results are returned as URL–text pairs; documents are not summarized by an additional model in the reported configuration.

A.8 Step-705 Training Configuration

For the step-705 fact-checker checkpoint, we train NVIDIA Nemotron 3 Nano 30B-A3B with synchronous GRPO in BF16. We use a distributed, precision-aware Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, zero weight decay, and gradient clipping at 1.0. The learning rate is held constant at 10^{-6}, with no warmup and no reference-policy KL penalty.

Each step samples 64 prompts and 16 generations per prompt, yielding a global training batch of 1,024 trajectories. The microbatch and log-probability batch sizes are both one. Advantages use the leave-one-out baseline without reward normalization. Rollouts permit up to 10 tool-use turns, a 40,960-token total context, and 16,384 generated tokens; initial inputs are filtered to at most 22,528 tokens.

We use random seed 42 and six four-GPU AWS GB200 nodes, for 24 GPUs in total. Four policy nodes (16 GPUs) use Megatron tensor parallelism of 4 and expert parallelism of 2, with colocated vLLM tensor parallelism of 2. Two judge nodes (8 GPUs) use tensor parallelism of 4 and data parallelism of 2.

A.9 STEP-240 QWEN 3.5 27B TRAINING CONFIGURATION

For the step-240 checkpoint, we train Qwen 3.5 27B with synchronous GRPO in BF16. Each step samples 64 prompts with 16 generations per prompt, yielding 1,024 trajectories. The global batch size is 1,024 and the microbatch size is one. Advantages use a leave-one-out baseline without reward normalization. We use a token-level loss, asymmetric clipping in $[0.8, 1.28]$, importance-sampling correction, and no reference-policy KL penalty.

The learning rate is held constant at 10^{-6}, with no warmup, a minimum learning rate of 10^{-6}, and zero weight decay. Rollouts permit up to 10 tool-use turns and use a 40,960-token total context with up to 16,384 generated tokens; inputs are filtered to at most 22,528 tokens. Sampling uses temperature 1.0 and top-p 1.0 with Qwen reasoning enabled. The environment uses lexical retrieval, returning the top two results with up to 8,000 characters each and no summarization. The reward is $R_{\mathrm{F1}} + 0.1R_{\mathrm{severity}}$.

Training uses four four-GPU AWS GB200 policy nodes (16 GPUs) with Megatron tensor parallelism of 4 and pipeline, context, and expert parallelism of 1. Rollouts use a colocated asynchronous vLLM engine with tensor parallelism of 2, 8,192 maximum batched tokens, 192 sequences, and GPU utilization set to 0.50. Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025) serves as the judge on one additional four-GPU node with tensor parallelism of 4 and data parallelism of 1; judge sampling uses temperature 0.6. The complete allocation is five nodes and 20 GPUs. We run validation every 10 steps and save a checkpoint every five steps.

A.10 NEMOTRON 3.5 LIGHTNING POLICY-OPTIMIZATION CONFIGURATION

Table 6 reports the main downstream policy-optimization recipe that combines the GenRM and FactCheckRL rewards. We initialize from the Nemotron 3.5 Lightning SFT checkpoint and optimize it with synchronous GRPO. For both the Nemotron 3.5 Lightning and Nemotron 3 Super policies reported in Tables 2 and 3, we use the harmonic objective with $\lambda = 0.75$ and set the score floor to $\epsilon = 0.2$.

Setting	Value
Starting policy	Nemotron 3.5 Lightning SFT checkpoint
Algorithm	Synchronous GRPO
Training data	5,792 structured prompts containing 2–5 unsupported claims; 201 held-out validation examples
Rollouts	64 prompts $\times$ 16 generations = 1,024 trajectories per step
Batching	Global batch 1,024; microbatch 1; generation batch 32
Advantage estimation	Leave-one-out baseline; reward normalization disabled
Reward combination	Harmonic GenRM–fact-checker objective; $\lambda = 0.75$; score floor $\epsilon = 0.2$
Learning rate	10^{-5}, effectively constant (one warmup iteration)
Regularization	Zero weight decay; KL coefficient 0
GRPO clipping	Asymmetric clipping in $[0.8, 1.28]$
Policy sampling	Temperature 1.0; top-p 1.0
Sequence lengths	12,288 input tokens; 24,576 generated tokens; 40,960 total context
Policy topology	Two four-GPU nodes (8 GPUs); TP=2, EP=4; colocated vLLM TP=2
GenRM serving	Two four-GPU nodes; 8 replicas; TP=1, DP=8; reasoning enabled; temperature 1.0; top-p 0.95
Fact-checker serving	Step-705 checkpoint on four four-GPU nodes; TP=1, DP=16
Fact-checker loop	At most 10 tool calls; no forced final call
Total hardware	Eight four-GPU nodes (32 GPUs)
Checkpointing	Every five steps; retain seven checkpoints
Online validation	Disabled for the reported production runs
Random seed	42

Table 6: Configuration for downstream Nemotron 3.5 Lightning policy optimization using both the GenRM and fact-checker rewards.