EAVER: LONG-FORM FACTUALITY VERIFICATION AS AN END-TO-END AGENTIC POLICY

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-form factuality verification is commonly implemented as a static decompose-search-verify pipeline, with separately prompted modules processing claims and invoking external search. Treating claims independently makes LLM and search calls scale with claim count and causes repeated searches for overlapping evidence about related claims. We introduce **EAVer**, an **E**nd-to-end **A**gentic **Ver**ifier that learns to control the complete response-level verification workflow as a unified policy. **EAVer** groups semantically related claims, routes each group to direct verification or targeted search based on confidence, and keeps evidence returned by search in compact in-context memos for cross-claim reuse. To train this policy, we develop a privileged-teacher synthesis pipeline that converts gold claim annotations into executable multi-turn tool-interaction trajectories with live search rather than post-hoc rationales. Structural, label-alignment, tool-use, search-budget, and leakage checks yield 1,447 quality-controlled trajectories. We further construct 794 bidirectional same-trajectory preference pairs that keep claim grouping, search, and evidence fixed, enabling decision-focused Direct Preference Optimization (DPO) over factuality-decision tokens. The results with Qwen3-8B show that **EAVer** outperforms the strongest search-based baseline on each benchmark by 2.88 Macro-F1 points on VeriFastScore and 4.73 points on the out-of-distribution FaStFact-Bench, while using about 80% fewer searches than the most search-efficient baseline. Moreover, **EAVer** consistently improves performance across models ranging from 4B to 32B parameters, demonstrating its strong generalizability.

1 INTRODUCTION

Large language models (LLMs) can produce fluent, extended responses across open-ended question answering and instruction-following settings (Brown et al., 2020; Ouyang et al., 2022). Yet fluency does not guarantee factual reliability: generated answers may contain plausible but unsupported statements that are difficult to detect at the response level (Lin et al., 2022; Manakul et al., 2023). This challenge is amplified in long-form generation, where factual statements are distributed across sentences and paragraphs and may connect entities, dates, quantities, events, and causal relations. Reliable evaluation therefore requires fine-grained verification that identifies individual claims and grounds each judgment in the relevant evidence for that claim.

Existing long-form factuality evaluators generally follow a decompose-search-verify recipe: extract factual claims, search for external evidence, and predict whether each claim is supported (Min et al., 2023; Wei et al., 2024; Song et al., 2024). Subsequent work broadens the range of verifiable content, uses iterative search for difficult claims, or batches extraction and verification for greater efficiency (Xie et al., 2025; Rajendhran et al., 2025). Nevertheless, search control and verification remain hand-orchestrated or separately optimized rather than jointly learned as a unified policy for response-level evaluation.

Existing systems have two structural limitations: separately prompted modules impose a fixed controller, and claim-local processing repeatedly retrieves overlapping evidence. On an independent VeriFastScore validation split, prior pipelines issue 35.9–103.7 searches per sample with 23.5–66.5% repeated links, whereas **EAVer** uses 3.42 searches with 0.3% redundancy (Figure 1). Redundancy persists even when query strings differ, so string-level caching does not fully address

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.

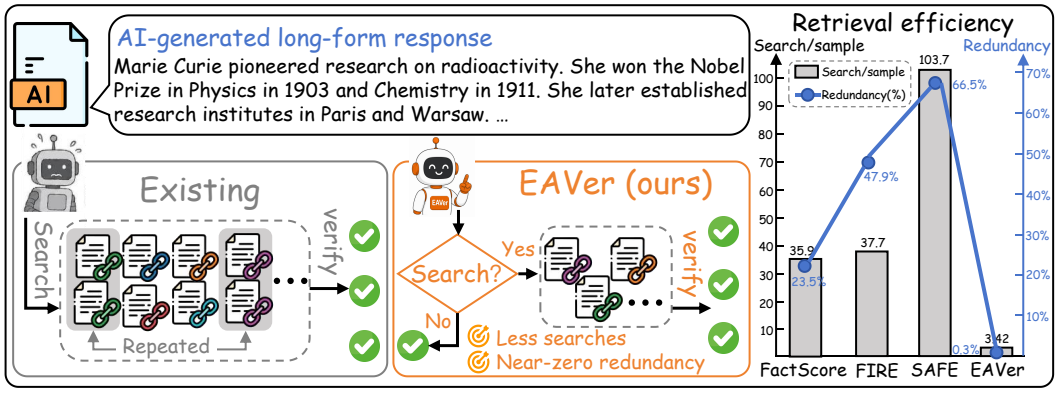


Figure 1: **Search efficiency of EAVer.** *Left:* Existing per-claim evaluators repeatedly search for overlapping evidence; **EAVer** adaptively searches and reuses evidence across related claims. *Right:* Mean searches per sample and link redundancy on an independent VeriFastScore validation split. Link redundancy is the fraction of links repeated within the same evaluation sample.

cross-claim reuse. Even the trained VeriFastScore verifier consumes evidence collected by a separate search stage rather than learning interactive search control (Rajendhran et al., 2025).

Motivated by agents that interleave reasoning with external actions (Yao et al., 2023; Schick et al., 2023), we learn verification as a unified response-level policy. **EAVer** groups related claims, routes groups to direct verification or targeted search by confidence, and stores retrieved evidence in compact in-context memos for later reuse. To supervise this behavior, a privileged teacher uses gold claims and labels to synthesize executable multi-turn trajectories with live search, extending synthetic instruction supervision (Wang et al., 2023) to the complete verification workflow. The student receives no gold annotations at inference. Structural, label-alignment, tool-use, budget, and leakage checks retain demonstrations that faithfully instantiate the policy.

Base verifiers strongly favor the supported class, allowing unsupported content to pass as factual. After demonstration training, we therefore study decision-focused Direct Preference Optimization (DPO) (Rafailov et al., 2023) as a conservative extension. Matched preference pairs differ only in factuality decisions, and a token mask restricts the preference objective to verdicts. We contribute:

- **Search-efficiency analysis.** We quantify per-sample search overhead and cross-claim evidence redundancy in representative long-form factuality verifiers.
- **Policy learning.** We introduce a privileged-teacher synthesis and quality-control framework, together with controlled decision preferences, to train a unified policy for claim grouping, adaptive search, evidence reuse, and factuality prediction.
- **Empirical validation. EAVer** achieves the strongest Macro-F1 among search-based systems on both benchmarks with about one fifth as many searches as the most search-efficient baseline, and improves verification across model families and scales.

2 Related Work

Long-form factuality evaluation. FActScore established fine-grained evaluation by decomposing a generation into atomic facts and measuring the fraction supported by a reliable knowledge source (Min et al., 2023). FacTool generalized tool-augmented factuality detection across tasks through claim extraction, query generation, evidence collection, and agreement checking (Chern et al., 2023), while SAFE operationalized long-form evaluation with LLM-based decomposition and web-search verification (Wei et al., 2024). VeriScore further distinguishes verifiable claims from subjective or otherwise uncheckable content (Song et al., 2024). More recent systems improve either search adaptivity or throughput. FIRE iteratively searches for and verifies evidence until an individual claim can be decided (Xie et al., 2025), whereas VeriFastScore uses sentence-level batch search and a fine-tuned model to jointly extract and verify claims (Rajendhran et al.,

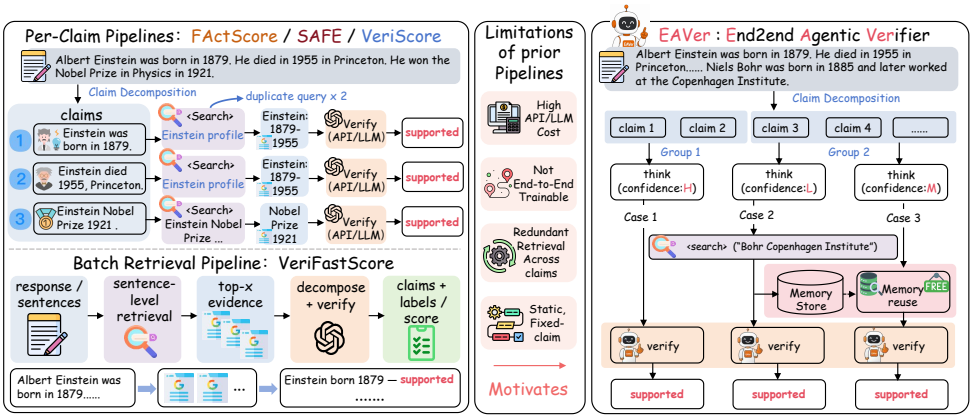


Figure 2: **Fixed verification pipelines versus EAVer.** *Left:* FActScore, SAFE, and VeriScore verify claims independently; VeriFastScore batches sentence-level search. *Center:* Fixed pipelines have four limitations: high API and LLM costs, no end-to-end training, cross-claim search redundancy, and static claim processing. *Right:* **EAVer** groups related claims, routes them to direct verification or search by confidence, and stores evidence for reuse.

2025). However, their overall control remains claim-local or stage-wise, with search and verification either prompt-orchestrated or optimized separately. Our work instead studies a joint policy for response-level search and verification.

Learned search agents. Recent work trains language models to treat search as part of the reasoning policy rather than a fixed preprocessing step. R1-Searcher and Search-R1 use outcome-based reinforcement learning to elicit autonomous, multi-turn search (Song et al., 2025a; Jin et al., 2025), while ReSearch learns interleaved reasoning and search without supervised reasoning traces (Chen et al., 2025). Smart-Searcher combines an SFT cold start with reinforcement learning to encourage dynamic use of internal and external knowledge (Song et al., 2025b). These methods primarily target knowledge-intensive question answering, where search supports a single final answer. Long-form factuality verification instead requires exhaustive coverage of claims across an entire response, aligned claim-level judgments, and control of cumulative search cost; our work studies learned search under this multi-claim objective.

3 Task Definition

The input is a question q and a model response y. For evaluation, the response is annotated with a gold set of factual claims $C = \{c_i\}_{i=1}^{n}$, where each gold claim has a binary label

$$z_i \in \{\text{SUPPORTED}, \text{UNSUPPORTED}\}.$$

The system must predict a claim set $\widehat{C} = \{\widehat{c}_j\}_{j=1}^{\widehat{n}}$ and assign each predicted claim a label $\widehat{z}_j \in \{\text{SUPPORTED}, \text{UNSUPPORTED}\}$. In our current evaluation, predicted claims are greedily matched one-to-one to gold claims using string similarity with threshold 0.5. For matched claims, predicted labels are scored against the corresponding gold labels. We report claim-extraction F1, macro-F1, and False Support Rate (FSR).

Claim extraction F1. Let M be the set of matched predicted–gold claim pairs. We compute extraction precision, recall, and their harmonic mean as

$$P_{\text{ext}} = \frac{|M|}{|\widehat{C}|}, \qquad R_{\text{ext}} = \frac{|M|}{|C|}, \qquad F_1^{\text{ext}} = \frac{2P_{\text{ext}}R_{\text{ext}}}{P_{\text{ext}} + R_{\text{ext}}}.$$

The last quantity is reported as extraction F1 in Table 2. The combined metric evaluates decomposition independently of factuality labels and penalizes both omitted gold claims and over-decomposition with unmatched predictions.

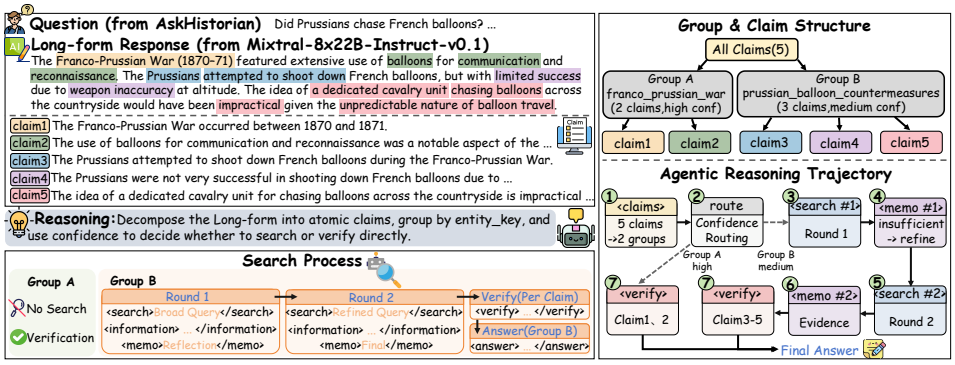


Figure 3: **A representative EAVer inference trajectory.** Five claims form two entity groups. The high-confidence group is verified directly; the medium-confidence group uses two searches and shared evidence before all five verdicts are returned.

Macro-F1. Macro-F1 is the average of F1 for SUPPORTED and F1 for UNSUPPORTED. It is the primary score because the supported class dominates the validation set.

False Support Rate (FSR). FSR measures false support among gold UNSUPPORTED claims:

$$\mathrm{FSR} = \frac{\#(\text{gold} = \textsc{unsupported},\ \text{pred} = \textsc{supported})}{\#(\text{gold} = \textsc{unsupported})}.$$

Lower FSR is better because false support allows unsupported content to be accepted as factual. FSR therefore isolates the asymmetric error that is most consequential for a factuality verifier. Macro-F1 summarizes performance across both classes, so reporting the two metrics together captures both balanced class performance and false-support risk.

4 EAVER

EAVer is designed around three principles: (1) claim verification should be optimized as a single trajectory rather than a hand-written pipeline, (2) search should be conditional on model confidence, and (3) evidence should be reusable across claims that share entities or events.

4.1 POLICY-AWARE VERIFICATION TRAJECTORY SYNTHESIS

Final-label supervision offers little guidance for the behavior preceding a factuality decision. For example, the zero-shot Qwen3-8B verifier in Table 2 has FSRs of 89.18% on VeriFastScore and 89.72% on FaStFact-Bench, demonstrating a consistent tendency to accept unsupported claims as factual before policy training. Search access alone does not teach the model when to doubt a plausible claim, seek counter-evidence, or overturn the response. Rather than distilling only final labels, we synthesize executable demonstrations that jointly supervise claim decomposition, confidence-conditioned search, query formulation, evidence summarization, and negative verification.

Figure 3 illustrates one such real trajectory from the in-domain VeriFastScore dataset.

We use Claude Opus 4.8 as a privileged teacher during data construction. It privately receives gold atomic claims and labels to construct label-consistent trajectories; the student sees neither at inference. The teacher copies each claim verbatim into a `<claims>` JSON array and assigns two policy fields: `confidence` (*high*, *medium*, or *low*) and an `entity_key`. Claims sharing an entity key are processed together. An all-high-confidence group is verified directly in a batch, whereas any medium- or low-confidence claim triggers one focused search for the group. The trajectory uses five structured tags: `<claims>`, `<search>`, `<memo>`, `<verify>`, and `<answer>`. When the teacher ends a turn with `<search>query</search>`, generation pauses; the runtime parses the query, calls the search tool, injects its results as an `<information>` environment message, and resumes with full history. This yields multi-turn tool-interaction transcripts rather than post-hoc textual rationales alone.

After search, the teacher compresses the long, noisy observation into a compact <memo> evidence summary before emitting one <verify> decision per claim and an aligned <answer> JSON. The memo is an explicit evidence-summary scaffold, not an external cache or lookup store: it remains in the shared dialogue context so that claims in the same entity group can implicitly reuse the evidence without a separate recall action. We retain trajectories that pass structural, label-alignment, tool-use, search-budget, and private-key leakage checks. Twenty human experts additionally conducted random spot checks of the retained trajectories, with 98% of inspected trajectories passing the audit. The retained demonstrations provide process-level supervision for claim organization, evidence acquisition and reuse, and aligned factuality decisions. Table 1 summarizes the two corpora. Trajectory length counts generated text and injected search observations after the initial prompt. Label shares cover 19,334 SFT decisions and 15,686 decisions in chosen DPO completions. The 794 matched pairs include 405 false-support-to-unsupported and 389 false-unsupported-to-supported corrections; rejected completions share all search and evidence with their chosen counterparts.

Table 1: **Synthetic training-data overview.** DPO statistics describe chosen completions.

Metric	SFT	DPO
Trajectories / pairs	1,447	794
Searches / traj.	2.24	3.69
Length (k chars)	9.16	13.80
Supported	70.4%	61.7%
Unsupported	29.6%	38.3%

Policy demonstration training. We train the verifier by maximizing the likelihood of assistant actions in each serialized trajectory τ. Let a_t indicate an assistant-content or assistant end-of-turn token. The demonstration objective is

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}_{\tau} \sum_{t=1}^{|\tau|} a_t \log \pi_\theta(\tau_t \mid \tau_{<t}). \tag{1}$$

System instructions, user inputs, search observations, and assistant role prefixes receive no loss. This teaches structured actions conditioned on evidence without imitating search-engine text. Trajectories use native conversation templates, are truncated to 8,192 tokens, and are not packed across samples. Appendix B specifies optimization, selected checkpoints, tokenization, and backbone-specific memory settings.

Optimization and sequence construction. We preserve all turns in chronological order and supervise assistant content together with its end-of-turn marker. System, user, and retrieval-environment tokens, as well as the assistant role prefix, receive the ignore index. Backbone-specific conversation markers and padding settings are detailed in Appendix B.

Training uses AdamW with a peak learning rate of 2×10^{-5}, cosine decay, a 0.05 warmup ratio, weight decay 0.01, and gradient clipping at 1.0. The per-device batch size is one on eight NVIDIA H20 GPUs. Models up to 14B use two gradient-accumulation steps for an effective batch of 16; the 32B model uses an effective batch of eight to accommodate memory constraints. We use bfloat16 precision, FlashAttention-2, and gradient checkpointing. Training seeds are 42, 123, and 2027; the backbone-specific epochs and distributed settings are provided in Appendix B.

4.2 Decision-Focused Preference Post-training

After supervised training on our policy-aware synthetic trajectories, the model has learned the complete verification policy, including claim organization, search, evidence use, and factuality prediction. Preference post-training is therefore used only as an optional correction for residual verdict errors. Applying trajectory-level preference optimization to independently sampled completions would entangle the desired label correction with incidental differences in search and evidence-use behavior. Among models with comparable macro-F1, we prefer the one with lower FSR: false support allows unsupported content to pass as factual, and identifying such negative cases is therefore more consequential for a verifier than confirming additional supported ones. We use preference post-training to seek an operating point with lower FSR while constraining changes to overall macro-F1, and localize the update through both controlled preference construction and a decision-focused loss.

Specifically, we analyze rollouts produced by the policy trained on the synthetic trajectory corpus and select trajectories with incorrect verdicts for which the collected evidence is sufficient to support

the gold decision. In these cases, the model has already obtained the evidence needed for verification but still emits an incorrect label, indicating a decision failure rather than a search failure. Rather than regenerate the full reasoning trajectory, we keep the trajectory fixed and correct only the erroneous verdict. For each selected case, the original erroneous completion serves as the rejected response, while the chosen response is created by correcting the affected <verify> labels and synchronized <answer> entries to the gold decisions. Claim grouping, search queries, search observations, evidence summaries, and all non-decision text remain identical. The resulting corpus contains 794 pairs spanning both false-support-to-unsupported and false-unsupported-to-supported corrections. Because the surrounding completion is matched, the preference signal isolates the factuality decision rather than differences in tool use or language realization.

We further restrict the DPO log-probability calculation to the decision tokens expressing SUPPORTED or UNSUPPORTED inside <verify> and <answer>. Let m_t be a binary mask that is one only for these decision tokens. The resulting masked sequence score is

$$s_\theta^M(y \mid x) = \sum_{t=1}^{|y|} m_t \log \pi_\theta(y_t \mid x, y_{<t}). \tag{2}$$

Defining the reference-relative score as $r_\theta^M(y, x) = s_\theta^M(y \mid x) - s_{\text{ref}}^M(y \mid x)$, and the preference margin as $\Delta r_\theta^M = r_\theta^M(y^+, x) - r_\theta^M(y^-, x)$, our decision-focused objective is

$$\mathcal{L}_{\text{DF-DPO}} = -\mathbb{E}_{(x,y^+,y^-)} \log \sigma\left(\beta \Delta r_\theta^M\right). \tag{3}$$

All claim text, search, evidence, memo, and formatting tokens therefore have $m_t = 0$ and are excluded from the preference loss; conventional DPO is recovered by setting $m_t = 1$ for every completion token. Spanning both correction directions prevents the training signal from reducing FSR simply by pushing every claim toward UNSUPPORTED. This optional stage directly supervises only the verdict tokens and does not reward changes to the preceding search trajectory. Section 6 tests whether this update lowers false support while preserving balanced factuality performance.

5 Experimental Setup

Backbone. We conduct our main evaluation of **EAVer** on Qwen2.5-7B-Instruct (Qwen Team, 2025) and Qwen3-8B (Yang et al., 2025), providing comprehensive validation across two strong open-weight models. Additional ablations span 4B–32B models from multiple families, including Llama 3.1 (Grattafiori et al., 2024), to assess the generalizability of our synthesized trajectories.

Search setting. Following Search-R1 (Jin et al., 2025), we implement the search actions used during training with a local dense-search backend, enabling stable and reproducible interaction trajectories. Queries are encoded with E5-base-v2 (Wang et al., 2022), and a FAISS index (Johnson et al., 2017) returns the top three search results. This fixed local search backend avoids fluctuations in the latency, availability, and outputs of external search APIs. For evaluation, we keep the model checkpoint, <search> action protocol, and decoding configuration fixed, but replace the local search backend with the Google Search API without retraining the model.

Data. We evaluate on two benchmarks. For in-domain evaluation, we construct a clean test set from the official 5,900-example test split of VeriFastScore (Rajendhran et al., 2025). We remove persona and creative-writing sources, abstentions and invalid responses, examples without valid binary claims, and any sample whose question–response fingerprint matches data used for training, trajectory synthesis, rollout generation, or development. The resulting set contains 22,532 claims, with zero sample-level fingerprint overlap with all touched pools. For out-of-distribution evaluation, we use the independently released, human-annotated FaStFact-Bench (Wan et al., 2025). We map its fine-grained labels to our binary scheme and discard non-factual housekeeping annotations, abstentions, and samples without verifiable claims, yielding 6,954 claims.

Baselines. We compare against two baseline families shown in Table 2. The search-free group, which uses no external search, includes LLM-only Qwen2.5-7B-Instruct (Qwen Team, 2025), LLM-only Qwen3-8B (Yang et al., 2025), Qwen3-Max, GPT-4.1, Claude Sonnet 4.6, and Claude Opus 4.8, each using one model invocation per sample. The search-augmented group comprises

Table 2: **Main results.** Results on **VeriFastScore** (in-domain) and **FaStFact-Bench** (out-of-domain, human-annotated). Searches/sample averages search-API requests over both datasets; LLM calls are excluded, and "–" denotes no external search. Evaluation metrics are percentages; Ext. F1 denotes claim-extraction F1. Best reported values in each column are **bold**.

Method	Searches/ sample	VeriFastScore (%)			FaStFact-Bench (%)		
		Macro F1	FSR ↓	Ext. F1	Macro F1	FSR ↓	Ext. F1
Zero-shot LLM verifiers							
Qwen2.5-7B-Instruct	–	43.61	96.71	64.69	48.58	96.66	71.30
Qwen3-8B	–	49.54	89.18	65.63	54.30	89.72	71.83
Qwen3-max	–	60.07	74.65	73.75	70.93	58.64	70.12
GPT-4.1	–	61.30	74.23	69.14	70.73	61.45	72.11
Claude Sonnet 4.6	–	62.66	72.53	74.17	70.75	41.18	62.95
Claude Opus 4.8	–	65.11	68.79	73.37	72.21	42.11	65.28
Search-augmented pipelines and task-trained verifiers							
FActScore	35.88	64.83	56.57	59.43	64.58	47.39	30.97
SAFE	106.90	63.40	64.01	58.92	62.10	65.60	56.82
VeriScore	19.18	63.38	55.47	77.53	60.89	50.25	46.58
FIRE	37.69	63.40	64.00	58.92	69.32	48.13	32.04
VeriFastScore	15.62	70.55	**38.28**	81.25	64.99	36.32	46.10
End-to-end Agentic Verifier							
EAVer (Qwen2.5-7B-Instruct)	4.21	71.77	38.51	80.52	66.01	41.20	71.82
EAVer (Llama-3.1-8B)	3.78	72.40	40.90	80.77	66.54	58.53	73.39
EAVer (Qwen3-8B)	**3.14**	**73.43**	41.57	**81.48**	**74.05**	**35.87**	**74.24**

FActScore (Min et al., 2023), SAFE (Wei et al., 2024), VeriScore (Song et al., 2024), FIRE (Xie et al., 2025), and VeriFastScore (Rajendhran et al., 2025). All experiments are run three times, and we report the mean over the three runs.

6 Main Results

Table 2 reports the main comparison on two held-out benchmarks under a unified evaluation protocol. We group methods by category: zero-shot LLM verifiers without search, search-augmented pipelines with GPT-4.1 backends, and **EAVer**.

Effectiveness. We select Qwen2.5-7B-Instruct, Qwen3-8B, and Llama-3.1-8B as three representative backbones from the Qwen and Llama families to evaluate the effectiveness of **EAVer** across model architectures. Among them, **EAVer** (Qwen3-8B) achieves the best overall Macro-F1 on both the in-domain VeriFastScore benchmark and the out-of-domain FaStFact-Bench, reaching 73.43 and 74.05, respectively. Compared with the search-free Qwen3-8B verifier, training on our agentic verification trajectories reduces FSR from 89.18 to 41.57 on VeriFastScore and from 89.72 to 35.87 on FaStFact-Bench. This pronounced reduction shows that the gain is not inherited from the backbone alone: the trained policy learns to use evidence to identify unsupported claims rather than defaulting to SUPPORTED. **EAVer** (Qwen3-8B) also obtains the highest claim extraction F1 on both benchmarks. By contrast, most prior search-augmented pipelines exhibit a substantial drop in claim extraction F1 out of domain. Our case inspection suggests that their static, LLM-driven decomposition stages tend to over-extract fine-grained or redundant claims, inflating $|\widehat{C}|$ without a commensurate increase in matched gold claims and thereby lowering extraction precision and F1. VeriFastScore attains the lowest in-domain FSR, but this result should be interpreted in light of its two-stage fine-tuning on roughly 9K synthetic prompt–response pairs and its collection of sentence-level evidence before claim decomposition (Rajendhran et al., 2025). Finally, Claude Opus 4.8 achieves a strong 72.21 Macro-F1 on FaStFact-Bench as a search-free verifier, providing empirical support for its use as our privileged teacher. Although neither trajectory synthesis nor post-training uses FaStFact-Bench examples or annotations, the resulting Qwen3-8B student reaches 74.05 Macro-F1, suggesting that the teacher-guided agentic supervision transfers beyond its source data.

Decision-focused preference post-training. Figure 4 evaluates decision-focused DPO applied to **EAVer** on VeriFastScore across three backbones under fixed search and decoding settings. Across all three backbones, applying DPO to **EAVer** lowers FSR while preserving or modestly improving Macro-F1. For Qwen2.5-7B, which exhibits the largest FSR reduction, FSR drops from 38.51 to 34.16, while Macro-F1 rises from 71.77 to 72.31. The consistent upper-left shifts show that preference post-training strengthens the ability of **EAVer** to reject unsupported claims rather than merely shifting the prediction prior.

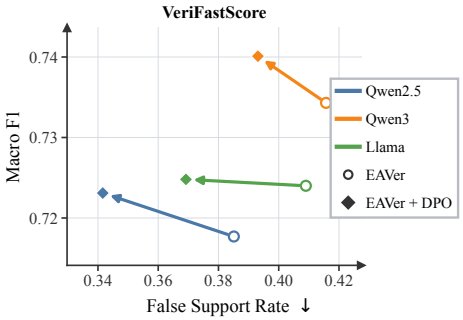
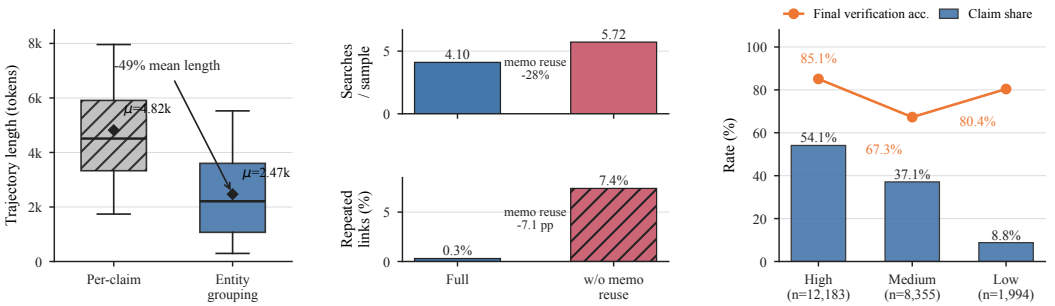


Figure 4: **Decision-focused DPO on VeriFastScore.** Circles/diamonds: before/after DPO. Upper-left shifts show lower FSR and higher Macro-F1. Colors mark backbones.

Cost-efficient verification. For a like-for-like comparison, Table 2 counts external search requests only. **EAVer** (Qwen3-8B) averages 3.14 searches per sample, about 80% fewer than the 15.62 used by VeriFastScore, the most search-efficient baseline. This measure does not include the LLM calls incurred by multi-stage orchestration. For example, SAFE can require up to $2+5\times(1+1)+1=13$ calls for a single relevant claim: 2 LLM calls for self-containment and relevance, 5 query-generation calls paired with 5 Google Search calls, and 1 final verdict call (Wei et al., 2024). In contrast, **EAVer** groups related claims, directly verifies high-confidence groups, searches only for uncertain groups, and reuses evidence returned by search through in-context memos, allowing one search to support decisions for several related claims.

7 Analysis and Extensions

We organize the remaining experiments around two questions: whether **EAVer**'s core components jointly enable efficient and reliable verification, and whether the learned agentic policy generalizes across model generations, parameter scales, and families alike.



(a) Grouping compresses trajectories (b) Memos reduce redundant search (c) Claim share and verification accuracy

Figure 5: **Mechanism-level component analysis for Qwen3-8B.** (a) On FaStFact-Bench, entity grouping reduces token-level trajectory length per sample. (b) On FaStFact-Bench, disabling memo-based evidence reuse increases searches per sample and repeated-link redundancy. (c) On VeriFastScore, bars show claim share and final verification accuracy by confidence level.

7.1 Component Ablations

Figure 5(a,b) and Table 3 report FaStFact-Bench ablations, while panel (c) diagnoses confidence behavior on VeriFastScore. In panel (a), grouping cuts trajectory length from 4,821 to 2,474 tokens (49%) through shared claim context. In panel (b), disabling memos increases searches per sample from 4.10 to 5.72 and repeated links from 0.3% to 7.4%, confirming that evidence reuse avoids redundant retrieval. Panel (c) shows a non-monotonic relation between confidence and accuracy: high-, medium-, and low-confidence claims reach 85.1%, 67.3%, and 80.4%, respectively. Medium-confidence claims can be deceptively familiar, prompting broad yet non-discriminative queries. For example, verifying "Einstein won the Nobel Prize in 1922" may search pages that mention both the 1921 prize and its 1922 presentation, making the award year easy to misread. Low-confidence claims often contain niche names or technical terms that support narrower queries and better-matched evidence.

Table 3: **Component ablations on FaStFact-Bench.** F1 is Macro-F1; trajectory tokens and search requests are averaged per sample.

Policy	F1 (%) ↑	FSR (%) ↓	Tokens ↓	Searches ↓
Full policy	**74.05**	**35.87**	**2,474**	**4.10**
w/o grouping	71.82	39.41	4,821	6.35
w/o conf. gate	72.19	36.07	4,058	7.65
w/o memo	72.41	40.29	4,260	5.72

7.2 Generalization of Model Families and Sizes

Using the same training and evaluation setup, we test **EAVer** on six Qwen and Llama models (4B–32B) on out-of-domain FaStFact-Bench.
Figure 6 shows that **EAVer** consistently improves long-form verification across both model families and at every tested scale, increasing Macro-F1 and reducing FSR for all six backbones. Qwen2.5-7B, the weakest base model in our evaluation, achieves a 17.4-point Macro-F1 improvement after policy training. This demonstrates policy transfer across model generations, families, and scales without backbone-specific synthesis.
Within Qwen3, the 4B and 32B models improve Macro-F1 from 52.7 to 67.9 and from 56.0 to 65.9, respectively. Their FSRs decrease from 91.4 to 52.1 and from 88.3 to 61.7. The gains vary with the backbone, and final performance is not monotonic in parameter count, indicating that model size alone does not determine verification quality.

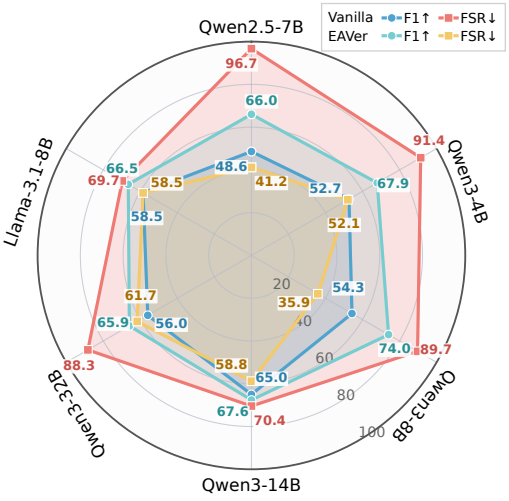


Figure 6: **Cross-backbone generalization on FaStFact-Bench. EAVer** improves Macro-F1 (↑) and FSR (↓) across all six backbones.

8 Conclusion

We introduced **EAVer**, a trainable response-level policy that combines claim grouping, selective search, in-context evidence reuse, and factuality decisions. Privileged-teacher trajectories provide supervision for the complete interaction, while controlled, bidirectional preferences support decision-focused refinement. On VeriFastScore and FaStFact-Bench, the Qwen3-8B policy achieves Macro-F1 scores of 73.43 and 74.05 with 3.14 searches per sample averaged across both benchmarks, about 80% fewer than the most search-efficient baseline.

Across six backbones, the same policy-learning framework improves verification across model families and scales. On VeriFastScore, decision-focused post-training further reduces false support while preserving or improving Macro-F1 across three backbones. Together, these results support learning evidence acquisition, reuse, and factuality judgment as a unified policy, and motivate extending agentic verification to broader long-form evaluation tasks.

AI USE STATEMENT

Language models are used as part of the research method: a privileged teacher synthesizes verification trajectories, and language-model verifiers and agentic policies are evaluated as described in Sections 4–7 and Appendix C. An AI coding assistant was also used to assist with the ICLR template migration, manuscript organization, and LaTeX formatting checks.

REPRODUCIBILITY STATEMENT

The task, matching protocol, evaluation data, and search settings are specified in Sections 3 and 5. Appendix B provides training objectives, hyperparameters, and backbone-specific settings. Appendix C details demonstration synthesis, filtering, the tool-interaction protocol, and preference-pair construction. It also gives teacher instructions and the checkpoint-selection criterion.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to reason with search for LLMs via reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2503.19470.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. Factool: Factuality detection in generative ai – a tool augmented framework for multi-task and multi-domain scenarios, 2023. URL https://arxiv.org/abs/2307.13528.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Jeff Johnson, Matthijs Douze, and Herve Jegou. Billion-scale similarity search with GPUs, 2017. URL https://arxiv.org/abs/1702.08734.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang 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, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.org/2023.emnlp-main.741/.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract.html.

Qwen Team. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Rishanth Rajendhran, Amir Zadeh, Matthew Sarte, Chuan Li, and Mohit Iyyer. VeriFastScore: Speeding up long-form factuality evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 9234–9259, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.491. URL https://aclanthology.org/2025.findings-emnlp.491/.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning, 2025a. URL https://arxiv.org/abs/2503.05592.

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. Smart-Searcher: Incentivizing the dynamic knowledge acquisition of LLMs via reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13572–13586, Suzhou, China, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.731. URL https://aclanthology.org/2025.findings-emnlp.731/.

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, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.552. URL https://aclanthology.org/2024.findings-emnlp.552/.

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, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.1295. URL https://aclanthology.org/2025.findings-emnlp.1295/.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2022. URL https://arxiv.org/abs/2212.03533.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

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, 2024. URL https://arxiv.org/abs/2403.18802.

Zhuohan Xie, Rui Xing, Yuxia Wang, Jiahui Geng, Hasan Iqbal, Dhruv Sahnan, Iryna Gurevych, and Preslav Nakov. FIRE: Fact-checking with iterative retrieval and verification. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2901–2914, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.158. URL https://aclanthology.org/2025.findings-naacl.158/.

An Yang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. 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 Evaluation and Reproducibility Details

Evaluation protocol. The agent generates a complete trajectory containing <claims>, <search>, <memo>, and <verify>blocks, and concludes with an <answer>JSON listing all predicted claims and labels. Predicted claims are matched to gold claims by string similarity (SequenceMatcher, threshold 0.5); matched claims contribute to confusion-matrix metrics. We report *claim coverage* (fraction of gold claims matched), *Macro F1* (mean of supported-F1 and unsupported-F1, our primary metric), *FSR* (the fraction of gold-UNSUPPORTED claims that are predicted as SUPPORTED), *searches per sample*, and *searches per claim* (SPC). Agentic evaluation uses max_turns= 20 and max_new_tokens= 8192.

B Training Design and Hyperparameters

This section provides the optimization, tokenization, and systems details for the two policy-training stages. The first stage teaches the complete verification behavior from executable demonstrations. The second stage starts from the resulting policy and applies a conservative preference update to factuality decisions. All backbone comparisons use the same 1,447 demonstrations, action schema, and optimization settings unless an architecture-specific memory constraint is stated explicitly.

B.1 Policy Demonstration Training

Training objective. Each training item is a complete multi-turn interaction consisting of a system instruction, a question and long-form model response, assistant actions, retrieval observations, evidence memos, claim-level decisions, and the final structured answer. We optimize causal language-model likelihood only on assistant outputs. Tokens belonging to the system, user, and retrieval-environment turns are assigned the ignore index, as is the assistant role prefix. The supervised target contains the assistant content and its end-of-turn marker. Consequently, the policy learns to generate <claims>, <search>, <memo>, <verify>, and <answer> outputs while conditioning on, but not imitating, the question, source response, or tool observation. This assistant-only objective also prevents long retrieved passages from dominating the loss.

Sequence construction. We serialize each sample with the native conversation markers of its backbone family and preserve all turns in chronological order. For Qwen2.5 and Qwen3, each message is enclosed by <|im_start|> and <|im_end|> markers. For Llama 3.1, we use the corresponding header and end-of-turn tokens and omit the auxiliary knowledge-date preamble so that training and evaluation use identical templates. We do not insert a separate reasoning marker and do not pack multiple samples into one sequence. Sequences are right-truncated to 8,192 tokens and padded to a multiple of eight; if a tokenizer has no dedicated padding token, its end-of-sequence token is used for padding.

Optimization and precision. Table 4 summarizes the shared configuration. Training uses the TRL supervised fine-tuning trainer, bfloat16 model and optimizer computation, FlashAttention-2, and gradient checkpointing with non-reentrant recomputation. We use AdamW with a peak learning rate of 2×10^{-5}, cosine decay, a 0.05 warmup ratio, weight decay 0.01, and gradient clipping at 1.0. The per-device batch size is one. On eight GPUs, two gradient-accumulation steps give an effective global batch of 16 for the models up to 14B parameters. Each configuration is trained independently with seeds 42, 123, and 2027. Logging is performed every five optimizer steps. Checkpoints are written at epoch boundaries and contain model weights only, which avoids retaining optimizer states solely for evaluation.

Distributed execution. All reported policies are trained on a single node with eight NVIDIA H20 GPUs, each with approximately 95–96 GB of memory. Models up to 14B parameters use DeepSpeed ZeRO-2 with optimizer-state CPU offload. The dense 32B model uses ZeRO-3, which shards parameters across all eight GPUs and offloads optimizer states to host memory. For the 32B setting, we additionally reduce gradient accumulation from two to one and disable data-loader worker processes to keep peak host memory below the node limit. These changes affect systems memory only; the demonstration corpus, loss mask, learning rate, schedule, and sequence length remain unchanged.

Table 4: **Shared policy demonstration training configuration.** The same semantic training corpus and objective are used for all backbones. Only the memory-sensitive settings listed in Table 5 vary.

Category	Setting	Value
Data	Demonstrations	1,447 multi-turn policy trajectories
Sequence	Maximum sequence length	8,192 tokens, no sample packing
Objective	Token supervision	Assistant content and assistant end-of-turn marker only
Optimization	Optimizer	AdamW
Optimization	Learning rate and schedule	2×10^{-5}, cosine decay
Optimization	Warmup and weight decay	0.05 warmup ratio, 0.01 weight decay
Optimization	Gradient clipping	1.0
Batching	Per-device batch size	1
Batching	Effective global batch	16 for models up to 14B; 8 for the 32B model
Numerics	Precision and attention	bfloat16, FlashAttention-2
Memory	Activation memory	Gradient checkpointing, non-reentrant implementation
Distributed	Data parallelism	One node with eight GPUs
Checkpointing	Save policy	Each epoch; weights only; at most three checkpoints
Reproducibility	Training seeds	42, 123, and 2027

Table 5: **Backbone-specific training settings.** Evaluations use the selected epoch. The 32B changes address host and device memory limits rather than backbone-specific tuning.

Backbone	Epochs run	Selected epoch	Grad. accum.	Global batch	ZeRO stage	Loader workers
Qwen2.5-7B-Instruct	3	2	2	16	2	2
Qwen3-4B	3	2	2	16	2	2
Qwen3-8B	3	2	2	16	2	2
Qwen3-14B	3	2	2	16	2	2
Qwen3-32B	2	2	1	8	3	0
Llama-3.1-8B-Instruct	3	2	2	16	2	2

B.2 Decision-Focused Preference Training

The input is the 794-pair bidirectional corpus described in the preference-pair construction below. Each pair has chosen and rejected completions with identical claim decomposition, tool interaction, retrieved evidence, memo content, and formatting. Only synchronized factuality labels differ.

The model attends to the full prompt and completion, including all search observations, but the DPO sequence score is accumulated only over the SUPPORTED and UNSUPPORTED decision tokens inside `<verify>` and `<answer>`. The decision-token mask therefore changes which positions contribute to the preference objective without deleting contextual evidence from the forward pass. The policy can use the complete trajectory to evaluate a label, while the preference gradient cannot reward incidental changes to search queries, evidence wording, or memo style.

Table 6 gives the complete preference-training configuration. We train for three epochs with a per-device batch of one and effective global batch of 32 across eight GPUs. The learning rate is 5×10^{-7} with cosine decay and a 0.10 warmup ratio. We use sigmoid DPO with $\beta = 0.1$, zero weight decay, no auxiliary negative-log-likelihood term, bfloat16 precision, gradient checkpointing, and a paged 8-bit AdamW optimizer. The maximum combined sequence length is 5,120 tokens and the maximum prompt length is 2,048 tokens. Reference-model log probabilities are precomputed with the same decision mask. Models are saved once per epoch; final checkpoint selection uses the disjoint development set and the constrained criterion described in the main paper.

C Policy Demonstration and Preference Data Construction

This section describes the two training corpora at the level of their statistical construction rather than implementation details. We refer to the first corpus as *policy demonstrations* because it supervises

Table 6: **Decision-focused preference training configuration.**

Setting	Value
Initialization	Selected policy checkpoint
Preference pairs	794
Epochs	3
Per-device batch size	1
Effective global batch	32
Learning rate	5×10^{-7}
Schedule	Cosine decay
Warmup ratio	0.10
Weight decay	0
DPO temperature β	0.1
Loss type	Sigmoid DPO
Maximum total length	5,120 tokens
Maximum prompt length	2,048 tokens
Precision	bfloat16
Optimizer	Paged 8-bit AdamW
Gradient checkpointing	Enabled, non-reentrant
Reference log probabilities	Precomputed
Auxiliary NLL coefficient	0
Random seed	42

the complete behavior of the verifier, not only its final factuality labels. The second corpus contains controlled *preference pairs* that refine the decision rule without changing the learned search policy.

C.1 Policy Demonstration Synthesis

Source pool and sampling. The source examples are drawn from the VeriFastScore training pool and contain a question, a long-form model response, and an ordered list of atomic claims with binary gold labels. We retain only examples with at least one valid binary claim and no more than 40 claims. Invalid responses, abstentions, and examples dominated by persona or creative writing are excluded. Sampling covers the eight source categories that occur in the corresponding development distribution: ELI5, AskHistorian, FreshQA, FActScore, new books, LongFacts, ShareGPT, and WritingPrompts. Source quotas follow the development distribution rather than oversampling examples with unsupported claims. This preserves a realistic mixture of supported, unsupported, and mixed-label samples.

Before synthesis, normalized question–response fingerprints are compared against every development and test pool. Any overlap is removed. The same fingerprint audit is repeated after merging synthesis batches and before model fitting. Because the claim annotations are used as privileged synthesis inputs, they are never used to select examples from the held-out evaluation benchmarks.

Controlled retrieval environment. Teacher trajectories interact with the same retrieval protocol later exposed to the student policy. The corpus is formed by deduplicating the title and content of retrieval snippets associated with VeriFastScore claims, yielding 1,098,517 unique passages. An E5-base-v2 encoder maps queries and passages to 768-dimensional vectors. A flat FAISS inner-product index returns the top three passages for every search. The retrieved passages are injected into the next environment turn inside an `<information>` block. Using a fixed local index keeps evidence stable across synthesis shards, training, and controlled evaluation, and ensures that the teacher cannot rely on capabilities unavailable to the student.

Privileged-teacher principle. A strong teacher receives the question, source response, exact ordered gold claims, and gold binary labels. This private key is used for three purposes only: (1) the `<claims>` array must reproduce the annotated atomic claims verbatim and in order; (2) confidence values are calibrated so that ambiguous details are routed to retrieval; and (3) `<verify>` and `<answer>` decisions must match the annotated labels. The teacher must nevertheless reach

those decisions through common knowledge or retrieved evidence. The generated trajectory must not mention the private key, labels, hints, supervision, or equivalent privileged information.

Listing 1 shows the central portion of the teacher prompt. The omitted portions give detailed confidence examples, entity-key naming rules, and the same structural constraints that are checked automatically after generation.

```
You are an expert long-form factuality verification agent producing a
high-quality reasoning demonstration for a student model to learn from.

[PRIVILEGED ANSWER KEY: never reveal or reference in output]
You are given:
  - the exact atomic claims to extract, and
  - the correct supported or unsupported label for each claim.

The key has only three uses:
  (1) reproduce the claims verbatim and in order in <claims>;
  (2) calibrate confidence for search routing;
  (3) reach the same final labels through genuine evidence reasoning.

Never refer to the key, gold labels, hints, ground truth, privileged
information, or provided supervision in any output token.

Evidence rule:
  supported   = evidence substantiates the specific claim.
  unsupported = evidence contradicts the claim or does not confirm its
                specific date, number, name, or causal link.

Hard constraints:
  - <claims> is parseable JSON with consecutive ids.
  - <search> is the last tag in an action turn.
  - Every search is followed by one <memo> before verification.
  - Claims sharing an entity_key are processed in one group.
  - Every claim receives exactly one <verify>.
  - <answer> contains all claims in order.
  - At most eight searches are issued.
```

Listing 1: Privileged-teacher instruction excerpt used during policy demonstration synthesis.

The corresponding teacher user message is instantiated separately for each sample:

```
Question:
{question}

Response to verify:
{response}

[PRIVATE SUPERVISION: never reveal or reference in any output token]
Decompose the response into EXACTLY these {N} atomic claims, verbatim
    and
in the given order, and reach EXACTLY these labels
in <verify> and <answer>:

1. [supported] {claim_1}
2. [unsupported] {claim_2}
...
N. [supported|unsupported] {claim_N}

All output must read as a genuine investigation. Start with <claims>.
```

Listing 2: Teacher user-message template. Braced fields are filled per sample.

Five-tag trajectory schema. Every retained demonstration follows one compact action language: <claims> specifies claim decomposition and routing, <search> invokes the only external tool,

<memo> stores a short evidence summary in the shared context, <verify> records one claim decision, and <answer> aggregates all claims and labels. We intentionally omit separate route, recall, and free-form reasoning tags. The reduced schema exposes the decisions that matter for control while keeping evidence reuse implicit in the conversation history.

The <claims> array contains four fields per claim. id is a consecutive integer, claim is the verbatim atomic claim, confidence is *high*, *medium*, or *low*, and entity_key is a normalized central entity or topic. Claims with the same entity key form a group. If all members of a group have high confidence, the teacher verifies them directly in a single batch. If any member has medium or low confidence, the group issues one focused search, writes one memo, and then verifies every member, including any high-confidence members, against the shared context. This makes search count depend on uncertain entity groups rather than raw claim count.

```
<claims>
[
  {"id":1,
   "claim":"Einstein won the Nobel Prize in Physics.",
   "confidence":"high",
   "entity_key":"einstein"},
  {"id":2,
   "claim":"Einstein won the Nobel Prize in 1922.",
   "confidence":"medium",
   "entity_key":"einstein"}
]
</claims>

<search>Einstein Nobel Prize in Physics award year</search>

<information>
Doc 1 states that the 1921 Nobel Prize in Physics was awarded to
Albert Einstein and presented in 1922.
</information>

<memo>Einstein received the 1921 Nobel Prize in Physics (Doc 1).</memo>
<verify id=1>supported</verify>
<verify id=2>unsupported</verify>

<answer>{"claims":[
  {"claim":"Einstein won the Nobel Prize in Physics.",
   "label":"supported"},
  {"claim":"Einstein won the Nobel Prize in 1922.",
   "label":"unsupported"}
]}</answer>
```

Listing 3: Example five-tag trajectory with environment-generated retrieval observations.

Confidence calibration. High confidence is reserved for stable, textbook-level facts that the teacher can recall specifically. Medium confidence denotes familiar subject matter with uncertainty about an exact date, quantity, name, or causal relation. Low confidence denotes niche, recent, or technical content that requires retrieval. The teacher is explicitly instructed not to mark every claim low, which would teach wasteful search, or every claim high, which would suppress tool use. Calibration is evaluated indirectly through schema consistency, search placement, and the empirical distribution of searches rather than by treating confidence as an independently supervised target.

Entity grouping and evidence reuse. Entity keys are lowercase, snake-case identifiers of the central subject. They are intentionally broader than an individual claim. For example, claims about Einstein's award, birthplace, and nationality share the key einstein; a key such as einstein_birth_year_1879 would be rejected as unnecessarily narrow. Groups are processed in first-appearance order. A memo is a one-sentence summary that retains the decisive names, dates, numbers, and a compact document reference. Because the memo remains in the dialogue history, later decisions can attend to it directly. There is no external memory store and no separate recall operation.

Interactive rollout. Generation is paused whenever the teacher completes a <search> action. The environment parses the query, returns three passages inside an <information> turn, and resumes generation with the complete history. The search closing tag is used as a generation stop condition, ensuring that the tool action is the final tag of its turn. At most eight searches and 30 assistant-generation calls are allowed for one sample. A generation reaching its length limit without a complete <answer> is discarded rather than truncated into a nominally valid demonstration.

Quality-control gates. Every synthesized sample passes all of the following checks before entering the training corpus:

1. **Schema validity.** The <claims> payload is parseable JSON; claim identifiers are consecutive; confidence values belong to the three-level vocabulary; and entity keys satisfy the normalized naming rule.
2. **Decomposition alignment.** The number, order, and stripped text of generated claims match the annotated claims exactly. This prevents teacher paraphrases from changing the unit scored by the evaluator.
3. **Tool protocol.** Search actions occur at turn boundaries, stay within the eight-search budget, and are followed by a memo before the corresponding decisions.
4. **Decision alignment.** Every claim identifier occurs in exactly one <verify>; its label equals the annotation; and the final <answer> repeats all claims and labels in order.
5. **Leakage rejection.** Every assistant turn is scanned case-insensitively for references to an answer key, gold labels, privileged information, supervision, hints, or equivalent phrasing. A match rejects the complete sample rather than editing the offending phrase.
6. **Student-view reconstruction.** The privileged system and user turns are removed and replaced with the deployment prompt. Assistant actions and retrieval observations are preserved verbatim. The reconstructed conversation is scanned again.
7. **Dataset-level audit.** Duplicate identifiers and normalized question–response fingerprints are removed, and overlap with all development and held-out sets is required to be zero.

After quality control and deduplication, the final corpus contains 1,447 trajectories. The multi-stage rejection procedure is deliberately conservative: a malformed, misaligned, truncated, or potentially leaked sample is dropped rather than repaired. Random human spot checks provide an additional semantic audit of tool use and evidence-label consistency.

Student view. The student receives no gold claims or labels. Its system prompt states the action schema, confidence-conditioned routing rule, evidence semantics, and search budget. Its user message contains only the question, long-form response, and an approximate claim-count cue. Listing 4 provides the complete policy portion of the prompt used to serialize the final demonstrations.

```
You are a long-form factuality verification agent.
You will receive a question and a model's response.
Verify the response by:
decomposing it into atomic claims with confidence calibration, grouping
    by
entity, and verifying each group with search when needed.

Process:
  1. Emit <claims> as a JSON array. Each object has:
       "id":          int 1..N
       "claim":       atomic claim text, verbatim from the response
       "confidence":  "high" | "medium" | "low"
       "entity_key":  lowercase snake_case central entity

  2. Process each unique entity_key in first-appearance order:
     - If all claims in the group are high confidence, output batched
       <verify> tags only.
     - If any claim is medium or low confidence, output one <search>
       covering the group, wait for <information>, write one <memo>,
       and then verify all claims in the group.
```

```
  3. Finish with:
       <answer>{"claims":[
         {"claim":"...", "label":"supported|unsupported"}, ...
       ]}</answer>

Rules:
  - Search only when a group contains a medium or low confidence claim.
  - After every useful search, write a memo before verification.
  - Process all claims sharing an entity_key in one group turn.
  - "unsupported" includes cases where evidence does not confirm the
    specific date, number, name, or causal link.
  - Use at most eight searches.
```

Listing 4: Student policy prompt stored in the demonstration corpus.

```
Question:
{question}

Model Response:
{response}

(This response contains approximately {num_claims} verifiable claims.
Verify them as described above.)

You have access to one tool:
  <search>query</search>
The retrieval engine replies inside <information>...</information>.

After every useful search, write:
  <memo>one-line distilled evidence (Doc X)</memo>

Output the final aggregate in:
  <answer>{"claims":[...]}</answer>
```

Listing 5: Student user-message and tool-description template.

Final corpus statistics. The 1,447 retained trajectories contain 19,334 claim decisions. Of these, 70.4% are SUPPORTED and 29.6% are UNSUPPORTED. The mean trajectory issues 2.24 searches. Mean generated trajectory length, including injected retrieval observations after the initial prompt, is 9.16 thousand characters. Character length is reported because it is independent of the tokenizer used for each backbone. Search count is much smaller than mean claim count, reflecting both direct verification of high-confidence groups and reuse of one retrieval result across related claims.

C.2 PREFERENCE-PAIR CONSTRUCTION

Motivation and candidate rollouts. Policy demonstrations teach decomposition, routing, search, memo formation, and factuality decisions jointly. Preference data are constructed only after this policy has converged, and are used to correct residual decision errors. We generate policy rollouts on a candidate pool that is disjoint from the development and held-out evaluation sets. Only structurally valid trajectories with at least one incorrect verdict are considered. We further require the existing context to contain sufficient evidence for the gold decision so that the selected case represents a decision failure, not a missing-search failure.

Same-trajectory counterfactuals. For each eligible trajectory, the original model completion becomes the rejected member of a preference pair. The chosen member is produced by copying the complete trajectory and correcting only erroneous labels in two synchronized locations: the corresponding `<verify>` content and the matching `label` value inside `<answer>`. Claim text, identifiers, confidence, entity keys, group order, search queries, retrieval observations, memos, turn boundaries, and all other tokens are kept fixed. This construction turns each pair into a controlled counterfactual in which the factuality decision is the only changed variable.

```
Shared trajectory prefix:
  <claims>...</claims>
  <search>...</search>
  <information>...</information>
  <memo>...</memo>

Rejected:
  <verify id=7>supported</verify>
  ...
  {"claim":"...", "label":"supported"}

Chosen:
  <verify id=7>unsupported</verify>
  ...
  {"claim":"...", "label":"unsupported"}
```

Listing 6: Schematic same-trajectory preference pair. All omitted context is byte-identical.

Conventional preference construction from independently sampled completions would allow the chosen and rejected members to differ in search count, retrieved documents, decomposition style, memo content, length, and factuality labels. Those differences make it difficult to identify which behavior the optimizer is rewarding. In a same-trajectory pair, non-decision text is held fixed by construction. Its token log probabilities need not cancel, because an earlier corrected label can change the autoregressive context. The decision-token mask explicitly excludes non-decision positions from the sequence score, retaining only factuality-label tokens inside `<verify>` and `<answer>`. The full evidence remains visible as context at the active label positions.

Bidirectional corrections. A corpus containing only false-support corrections would lower FSR but could teach a degenerate policy that predicts UNSUPPORTED for every uncertain claim. We therefore retain both correction directions. There are 405 false-support-to-unsupported pairs and 389 false-unsupported-to-supported pairs, giving 794 pairs in total. The near-balanced direction count allows the preference signal to target specific evidence-conditioned errors rather than shift the global class prior. The chosen completions contain 15,686 claim decisions, of which 61.7% are SUPPORTED and 38.3% are UNSUPPORTED. They average 3.69 searches and 13.80 thousand characters per trajectory.

Preference-pair validation. Each pair is admitted only after the following invariants are checked:

1. The rejected rollout is structurally valid and its erroneous claim identifiers are matched to gold annotations.
2. Each corrected `<verify>` label matches its corresponding `<answer>` label.
3. Chosen labels match the gold decisions, while the rejected completion retains at least one identified false-support or false-unsupported error.
4. After replacing decision values with placeholders, chosen and rejected trajectories are identical. This checks the same-trajectory property.
5. Both correction directions are retained during balancing; no unsupported-only label policy can satisfy all pairs.
6. Duplicate pairs and question–response fingerprint overlap with development or held-out evaluation pools are rejected.

Relationship to the training loss. The pair construction and decision-focused loss enforce the same locality at two different levels. Data construction controls what differs between y^+ and y^-; the loss mask controls which token log probabilities contribute to the optimization objective. The active positions are only label values inside `<verify>` and `<answer>`. Search and evidence tokens remain in the causal prefix, so a decision can still depend on retrieved information, but they receive no direct preference reward.