ECPO: EVIDENCE-COUPLED POLICY OPTIMIZATION FOR VERIFIER-AUDITED CANDIDATE RANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Ranking systems used for decision support should expose evidence that can be audited independently of the model's ranking scores. We study a closed-roster setting in which a policy jointly emits a Top-K ranking and candidate-linked, span-grounded evidence bundles. Evidence-Coupled Policy Optimization (ECPO) uses rank-blind bundle permutation, global one-to-one assignment, candidate-level contrastive separation, and adaptive constraints to couple the ranking with its certificate. On MAVEN-ERE and RAMS, ECPO improves verifier-audited traceability, rank-blind recovery, and human comparative support relative to matched controls in the tested provenance settings. The gains persist under held-out evaluators, other 8B/14B backbones, and a HotpotQA answer-ranking interface. Ordinary NDCG and runtime remain explicit trade-offs.

1 INTRODUCTION

Ranking is often deployed where a score alone is insufficient. An analyst may need to know which source spans support a candidate, whether those spans belong to the exposed evidence scope, and whether the evidence distinguishes that candidate from alternatives. A fluent post-hoc explanation does not provide this contract: plausibility, attribution, and faithfulness can diverge (Ribeiro et al., 2016; Jacovi & Goldberg, 2020; Maynez et al., 2020; Atanasova et al., 2023). We therefore study a stricter output interface in which a policy jointly produces an ordered candidate list and machine-checkable evidence bundles.

Existing preference-optimization methods mainly optimize scalar preference signals and therefore assume that reward-model supervision is reliable. When outputs contain unverifiable claims or fabricated evidence, a high scalar score can reinforce unsupported behavior. ECPO treats verification consistency as part of the optimization target rather than a post-hoc filtering step.

At ranking time there is a finite window-local roster, every candidate has candidate-linked provenance units, and a declared support contract specifies what may count as evidence after direct candidate claims are removed. Our primary instantiations use frozen event extraction over MAVEN-ERE (Wang et al., 2022), which extends MAVEN (Wang et al., 2020), and RAMS (Ebner et al., 2020): a predefined skeleton supplies step and role semantics, while trigger and argument offsets provide provenance. These define the evaluated interface.

The central ambiguity is what an evidence metric actually establishes. A bundle can be parseable but unsupported, traceable to candidate A but uninformative about why A outranks B, or sufficient for a human while failing a particular automatic matcher. We consequently separate five properties: *validity* under the output schema; *traceability* to a candidate trajectory; *slot recovery* when a bundle remains at its submitted position; *rank-blind comparative recovery* after positions and candidate–bundle pairing are removed; and *human support*. We call this slot-wise metric SLOT-EVIDCONS@10 and reserve stronger language for the rank-blind and human audits.

ECPO treats the policy action as the joint object $(\text{ranking}, \text{certificate})$. A skeleton-conditioned reward model scores candidate trajectories. A deterministic validator checks schema and provenance. The rank-blind contrastive cycle (RBCC) strips candidate claims, generated event IDs, ranks, and policy scores, scores each remaining bundle against the full roster, solves a global assignment, and separates the selected candidate from alternatives. Adaptive certificate and RBCCconstraints are enforced during policy optimization. The gradient estimator, dynamic program, Bradley–Terry term,

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.

and KL regularization are established components; our contribution is the task interface, rank-blind assignment/contrastive objective, constrained optimization, and audit protocol.

Three empirical findings summarize the evaluation. First, ECPO improves verifier-masked ranking and rank-blind recovery over GRPO and a strong reward-model ranker with deterministic evidence optimization. Second, the advantage persists under held-out deterministic and raw-text verifiers, cross-backbone tests, a HotpotQA provenance schema, counterfactual evidence, and a blinded human audit. Third, the deterministic pipeline remains slightly stronger on ordinary NDCG and far cheaper. Together, these evaluations show improved verifier-audited traceability and comparative support in the tested closed-roster setting.

Our contributions are:

- a ranking interface whose action couples an ordered candidate list with candidate-linked provenance bundles;
- a rank-blind assignment and candidate-contrastive training objective with adaptive certificate constraints; and
- an audit protocol that separates slot traceability, rank-blind reconstruction, held-out-verifier transfer, counterfactual sensitivity, and blinded human judgment.

2 Related Work

Attribution and faithfulness. Attribution metrics ask whether generated claims are supported by cited sources (Rashkin et al., 2023; Li et al., 2024; Zhang et al., 2024); explanation work separately warns that plausible rationales need not be faithful to a model's decision process (Jacovi & Goldberg, 2020; Atanasova et al., 2023; Zaman & Srivastava, 2025; Chuang et al., 2026). Candidate prioritization has also been studied in investigative search and triage (Hung & Jayasumana, 2016; Sutmuller et al., 2020), where evidence access and accountability are consequential. ECPO audits an output-level contract: whether source evidence supports recoverable candidate decisions. The contract evaluates output-level evidence support.

Ranking, planning, and structured generation. Learning-to-rank methods optimize comparative order (Burges, 2010), while maximum-entropy reward learning (Ziebart et al., 2008), plan recognition (Ramírez & Geffner, 2009; 2010), and process-conformance/digital-twin models (Brockhoff et al., 2021; Park & van der Aalst, 2021) score partial traces against structured goals. Constrained decoding improves syntactic validity (Scholak et al., 2021), and RLHF, preference learning, and group-normalized optimization provide tools for structured LLM outputs (Ouyang et al., 2022; Schulman et al., 2017; Rafailov et al., 2023; Shao et al., 2024). ECPO combines known tools around a different action and evaluation contract: ranking and evidence are generated and audited jointly.

Verifier dependence. Verification can improve reasoning and generation, but optimizing directly against a verifier can encourage evaluator-specific behavior (Yuan et al., 2026; Helff et al., 2026). We therefore distinguish the reward verifier from two held-out evaluators, perturb its parameters, reverse the train/evaluate direction, and compare automatic judgments with blinded human annotations. These tests establish evaluator-form transfer within the shared roster and ontology.

3 Task and Evaluation Contract

For a ranking window w, let $\mathcal{D}_w$ be the exposed documents, C_w a finite candidate roster, $S_w = ([s_1, \ldots, s_M], \prec)$ a predefined soft-precedence skeleton, and $\tau_w(c)$ a possibly empty candidate trajectory. Events carry normalized stages and roles plus zero-indexed half-open provenance spans $(\texttt{doc_id}, [l, r))$. A candidate is positive when a source-normalized chain supported inside $\mathcal{D}_w$ is valid and aligned to S_w:

$$\mathcal{P}_w = \{c \in C_w : \exists z \in \mathcal{Z}^{\mathrm{src}}(w, c, S_w),\ \mathrm{valid}(z) = 1\}, \qquad \mathrm{rel}_w(c) = \mathbb{I}[c \in \mathcal{P}_w]. \tag{1}$$

The positive set and audit references are supervision/evaluation data and are absent from policy-visible inputs. The policy sees randomized window-local candidate IDs and emits

$$y = (A, E), \qquad A = [a_1, \ldots, a_k], \quad E = [E_1, \ldots, E_k], \quad k = \min(10, |C_w|), \tag{2}$$

where $E_j = [u_{j1}, \ldots, u_{jM}]$ is the ordered, step-aligned evidence bundle paired with a_j. Each u_{ji} cites compatible trigger/argument spans or is the null step $\perp$, serialized exactly as `matched:false`, `event_id:null`, and `evidence:[]`. The validator requires strict JSON, in-roster distinct candidates, exactly k positions and M steps per bundle, aligned certificates, valid documents and bounds, and trajectory/stage/role-compatible evidence.

This interface requires only: (i) a finite candidate set supplied or produced separately, (ii) candidate-linked provenance units such as spans, passages, cells, or logs, and (iii) an auditable support contract. The four-stage event skeleton and frozen extractor are one instantiation. The evaluation uses a declared provenance and support schema.

Metrics. NDCG@10 (Järvelin & Kekäläinen, 2002) measures ordinary ranking. FEASIBLERATE is the all-window validator pass rate. SLOT-EVIDCONS@10 is the fraction of submitted slots whose stripped bundle is reassigned to the candidate originally occupying that slot; it measures slot traceability. Let $z_{w,j} = 1$ only when slot j passes the validator and the declared verifier recovers a_j. Verifier-CertNDCG@10 (vCERTNDCG@10) is

$$\mathrm{vCertDCG@10}(w) = \sum_{j=1}^{k} \frac{(2^{\mathrm{rel}_w(a_j)} - 1) z_{w,j}}{\log_2(j+1)}, \tag{3}$$

$$\textsc{vCertNDCG@10}(w) = \begin{cases} \mathrm{vCertDCG@10}(w)/\mathrm{IDCG@10}(w), & \mathrm{IDCG@10}(w) > 0, \\ 0, & \mathrm{IDCG@10}(w) = 0. \end{cases} \tag{4}$$

It is thus ordinary binary-relevance DCG with a verifier-relative position mask. The boundedness observation $\mathrm{vCertDCG} \le \mathrm{DCG}$ follows from the position mask.

For rank-blind evaluation, candidate–bundle pairing and bundle order are additionally removed and bundles are permuted. A held-out evaluator returns a partial one-to-one assignment and orders recovered candidates by evidence score. For nonempty submitted order $S = (c_1, \ldots, c_k)$,

$$\textsc{SetRecover@10} = |\mathrm{set}(S) \cap \mathrm{set}(\widehat{S})|/k, \tag{5}$$

$$\textsc{RankRecover@10} = k^{-1} \sum_{j=1}^{k} \mathbb{I}[\widehat{S}_j = c_j]. \tag{6}$$

Missing, wrong, abstained, and exact-score-tied candidates receive zero positional credit. Pairwise accuracy scores correct/tied/reversed order over recovered submitted-target pairs as $1/0.5/0$; a window with no recovered pair scores zero. Kendall's τ_b retains evidence-score ties after restricting both orders to recovered selected candidates; degenerate windows score zero. $\textsc{AbstainRate@10} = (k - |\widehat{S}|)/k$. All five rank-blind metrics are zero for an empty target set. Abstention is calibrated rather than maximized: a lower rate is useful only when supported candidates remain accepted, while a higher rate can reflect conservative rejection of unsupported bundles.

4 EVIDENCE-COUPLED POLICY OPTIMIZATION

Figure 1 summarizes training and evaluation. The policy proposes the joint ranking/certificate object. A feasibility validator checks the claimed candidate at each position. The cycle verifier instead removes direct candidate claims, generated event IDs, explicit ranks, and policy scores before scoring evidence against every roster candidate. Unlike conventional RL post-training, ECPO includes an inference-time audit loop: each generated response is checked against the declared evidence contract before evidence-aware metrics are recorded, with abstention retained as an explicit outcome.

4.1 SKELETON ALIGNMENT AND TRAJECTORY UTILITY

For trajectory $\tau = [e_1, \ldots, e_T]$, a dynamic program aligns the ordered skeleton to compatible observed events. Its local score is

$$m(i,t) = \begin{cases} \mathrm{role_sat}(s_i, e_t) - \lambda_{\mathrm{arg}} \mathrm{arg_viol}(s_i, e_t), & \mathrm{stage}(s_i) \in \mathrm{skeleton_hits}(e_t), \\ -\delta_{\mathrm{type}}, & \text{otherwise}, \end{cases} \tag{7}$$

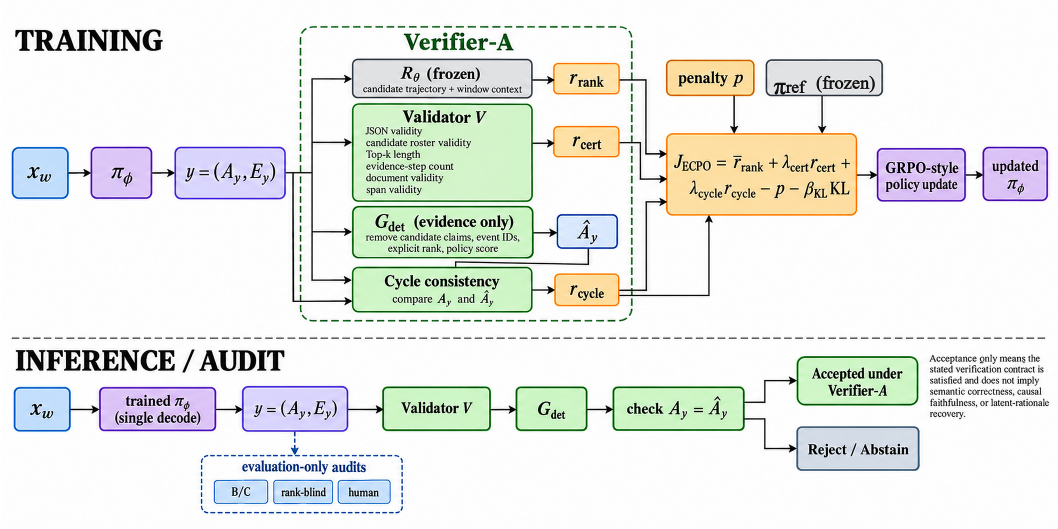

Figure 1: **ECPO overview.** Ranking and evidence are sampled jointly; the evidence cycle strips answer-bearing fields for candidate recovery.

Here $\text{skeleton_hits}(e_t)$ is the deterministic set of compatible stages assigned from the event type and trigger. If $\mathcal{R}_i$ is the required-role set and $\mathcal{A}_t$ the normalized arguments of e_t, then

$$\text{role_sat}(s_i, e_t) = \begin{cases} 1, & |\mathcal{R}_i| = 0, \\ |\mathcal{R}_i|^{-1} \sum_{r \in \mathcal{R}_i} \mathbb{I}[r \text{ occurs in } \mathcal{A}_t], & \text{otherwise}, \end{cases} \tag{8}$$

$$\text{arg_viol}(s_i, e_t) = \sum_{r \in \mathcal{R}_i} \mathbb{I}[r \text{ violates a declared role/entity constraint in } e_t]. \tag{9}$$

Main runs check candidate identity for key roles and available role-specific incompatibilities, but impose no extra entity-type constraint. Precedence is a soft post-backtracking feature, not a local-match term. Match, missing-step, and skipped-event transitions yield coverage, role, precedence, temporal, and optional graph features $\phi(\tau; S_w)$, with $R_\theta(\tau; S_w) = \theta^\top \phi(\tau; S_w)$. We learn θ with a window-wise maximum-entropy likelihood plus Bradley–Terry preferences over train-split positive, same-window, and perturbed trajectories. Appendix D gives the recurrence, penalties, tie rules, and reward-model objective.

4.2 Verifier-A

For cited and event spans in the same document, Verifier-A uses

$$\text{IoU}([l_s, r_s), [l_e, r_e)) = \frac{\max(0, \min(r_s, r_e) - \max(l_s, l_e))}{\max(r_s, r_e) - \min(l_s, l_e)}. \tag{10}$$

A match requires strict $\text{IoU} > 0.50$; equality fails. Compatible triggers are tested first. Arguments are considered only if no trigger matches and additionally require normalized-role agreement.

For stripped bundle b and candidate c, the declared support score is

$$A(b, c) = 0.30\,\text{StepCov} + 0.20\,\text{RoleSat} + 0.25\,\text{Trace} + 0.10\,\text{Prec} - 0.15\,\text{BadSpan}. \tag{11}$$

StepCov is the fraction of all M steps with valid, traceable, stage-compatible support (zero if $M = 0$). RoleSat is the fraction of required $(\text{step}, \text{role})$ instances supported, and equals one when none are required. Trace is the fraction of citations traceable to c and is zero for an empty bundle. Prec is the fraction of evaluable supported precedence edges that are consistent and is zero if none are evaluable. BadSpan is the fraction of citations with any invalid, duplicate, off-window, wrong-role, type-incompatible, non-traceable, or multi-candidate-ambiguous defect; each citation counts at most once and an empty bundle scores zero.

The weights encode priorities fixed before policy training and output inspection: coverage and traceability receive 0.55 jointly, role support 0.20, precedence 0.10 because order metadata can be unreliable, and defective spans -0.15. At the 0.55 support threshold, no single positive component

can pass; the 0.10 margin rejects near-ties. These choices used no development/test relevance labels. Appendix G evaluates 47 alternatives on fixed outputs.

Assignment requires $A(b, c_{\text{best}}) \geq 0.55$ and $A(b, c_{\text{best}}) - A(b, c_{\text{second}}) \geq 0.10$; equality passes and failure of either test abstains. A maximum-weight one-to-one assignment prevents reuse. Submitted slot-wise evaluation resolves ties by bundle order and then window-local candidate order. Rank-blind evaluation instead abstains on exact assignment ties and never uses bundle order; equal order scores remain ties. This distinction is essential to permutation invariance.

4.3 Evidence-coupled reward

For a parsed output $y = (A, E)$ with $L = |A|$, define

$$r_{\text{rank}}(y, w) = \sum_{j=1}^{L} \gamma^{j-1} R_\theta(\tau_w(a_j); S_w), \tag{12}$$

$$\overline{r}_{\text{rank}}(y, w) = \frac{r_{\text{rank}}(y, w) - \mu_{\text{rank}}(w)}{\sigma_{\text{rank}}(w) + \epsilon_{\text{rank}}}, \tag{13}$$

$$r_{\text{cert}}(y, w) = \frac{1}{k} \sum_{j=1}^{k} \mathbb{I}[V(E_j, a_j, w) = 1] \, \text{StepCov}(E_j, S_w, a_j), \tag{14}$$

$$r_{\text{cycle}}(y, w) = \frac{1}{k} \sum_{j=1}^{k} \mathbb{I}[\hat{a}_j = a_j]. \tag{15}$$

The first term is normalized within the window's sampled group; V is the feasibility validator, and $\hat{a}_j$ is Verifier-A's one-to-one assignment read at submitted slot j after answer-bearing fields are stripped. Indicators for missing positions $j > L$ are defined as zero. Thus certificate validity and evidence-only candidate recovery are separate signals. Here r_{rank} is supplied by the learned trajectory reward and r_{cert} measures validator-backed certificate support. The rank-blind contrastive cycle (RBCC) strips answer-bearing fields, scores every bundle against the full roster, solves a global one-to-one assignment, and adds candidate-level separation. Its temperature is $\tau = 0.15$ and assignment/contrastive mixture is $\alpha = 0.60$. The scalar reward uses the resulting r_{RBCC} term:

$$r_{\text{ECPO}} = \overline{r}_{\text{rank}} + \lambda_{\text{cert}} r_{\text{cert}} + \lambda_{\text{RBCC}} r_{\text{RBCC}} - \xi_{\text{inv}} \mathbb{I}_{\text{invalid}} - \xi_{\text{miss}} (k - |A|)_+ - \beta_{\text{KL}} \widehat{\text{KL}}_{\text{tok}}. \tag{16}$$

Unparseable outputs and outputs without a valid candidate prefix receive zero component utilities plus invalid/missing penalties; shorter parsed prefixes are scored only at present positions and penalized for the deficit. The main runs use discount 0.90, $\lambda_{\text{cert}} = 0.5$, $\lambda_{\text{RBCC}} = 1.0$, $\tau = 0.15$, $\alpha = 0.60$, $\kappa_{\text{cert}} = 0.85$, $\kappa_{\text{RBCC}} = 0.60$, dual learning rates $0.05/0.05$, EMA decay 0.95, multiplier cap 2.0, both penalties 2.0, group size 8, clip range 0.2, and initial $\beta_{\text{KL}} = 0.05$. A Qwen3-8B SFT policy is updated with GRPO-style group-normalized advantages (Shao et al., 2024); R_θ and Verifier-A are fixed during each policy phase. Appendix D gives the token-level KL estimator and clipped objective. This is an evidence-coupled formulation built from established optimization components.

5 Experimental Protocol

Primary datasets. MAVEN-ERE and RAMS supply source-normalized labels and audit references, while every ranking method consumes separate frozen-extractor trajectories. Ranking windows follow dataset-native document or document-local grouping, not calendar slicing. When no external roster is available, a label-free resolver fixes C_w from an identifier-only projection of source-normalized entities in agent-like or target-like roles; contextual entities enter only if they also occupy a candidate role. This projection retains anonymized identity and window membership but removes stage/chain validity, scores, positive labels, and reference spans before frozen predictions attach trajectories. Candidate-participating predicted events are ordered by timestamp when available and otherwise by dataset/document order. MAVEN-ERE compact contexts use one trigger sentence plus at most one neighbor on each side (no stride); RAMS extractor/SFT contexts use the first 120 characters, whereas ranking prompts retain full provenance pointers. Document-level splits prevent

a document from crossing train/dev/test. The evaluated construction contains 576/71/71 MAVEN-ERE and 716/90/90 RAMS windows; test windows average 10.1 and 9.7 candidates. Policy-visible IDs are randomized per window. The primary closed roster isolates ranking/evidence behavior; predicted and hybrid rosters are availability diagnostics.

Models and matched controls. Primary policies share the Qwen3-8B backbone (Yang et al., 2025), QLoRA adaptation (Dettmers et al., 2023), SFT initialization, prompt/schema, candidate pools, and single-decode budget. Plain GRPO uses train-label ranking utility plus schema, candidate-ID, and evidence-format validity; it uses neither R_θ nor a DP-alignment bonus. Besides SFT and GRPO, we emphasize two matched controls. RM-only+Det ranks with R_θ and searches validator-valid evidence after fixing the ranking. GRPO+Cert is separately trained with the plain-GRPO reward plus the same non-cycle r_{cert} and compute budget, but omits r_{cycle}. The $-r_{\text{cycle}}$ row is the exact within-ECPO mechanism ablation. Main values are test means; ECPO uses three seeds, and paired 95% intervals resample source windows, the inference and bootstrap cluster.

External validity tests. Verifier-B is deterministic and has no learned component: it replaces IoU with citation coverage, retains StepCov/Trace/BadSpan, removes RoleSat/Prec, and sets parameters from disjoint train-split synthetic corruptions without dev/test labels, policy outputs, rewards, or audit references. Verifier-C is a frozen policy-held-out raw-text evaluator that re-extracts entity, stage, and role from each citation plus ± 200 characters, maps entities to roster candidates through window-local aliases/coreference, assigns one-to-one with abstention, and orders by evidence-derived support. C's training/calibration uses only train-split raw-text/span annotations—never generated certificates, rankings, policy/reward outputs, or test documents. Unlike REEXTRACTFAITHFULNESS, which only audits per-span stage/role compatibility, C reconstructs candidate identity and order. Neither B nor C affects reward, checkpoint selection, decoding, or test thresholds for primary ECPO-A. A separate reverse-transfer ECPO-B policy uses B in its reward and is evaluated with A/C. We additionally test Llama-3.1-8B-Instruct (Meta, 2024), Qwen3-14B (Yang et al., 2025), and HotpotQA (Yang et al., 2018). HotpotQA uses an answer roster from shared retrieved context; its retained protocol is 740 dev/740 test examples with Qwen3-8B-Instruct, seeds $(42, 43, 44)$, greedy decoding with maximum output 768, and 2000 paired bootstrap replicates.

6 RESULTS AND VALIDITY

Table 1: **Primary utility and matched controls at $K = 10$.** Panel values are MAVEN-ERE (M) or RAMS (R). VCERTNDCG$_A$ and SLOT-EVIDCONS use Verifier-A; RANKRECOVER$_B$ is rank-blind held-out-Verifier-B evaluation. Bold and underlined entries mark the best and second-best values within each dataset/metric.

Data	Method	NDCG	SLOT-EVIDCONS	VCERTNDCG$_A$	Rank Recover$_B$	Feas.
M	SFT	0.61	0.36	0.31	0.27	0.77
M	GRPO	0.65	0.44	0.39	0.34	0.82
M	GRPO+Cert	0.67	0.50	0.45	0.40	0.90
M	RM-only+Det	**0.72**	<u>0.53</u>	<u>0.49</u>	0.41	**0.98**
M	ECPO $-r_{\text{cycle}}$	0.68	0.49	0.43	<u>0.46</u>	0.93
M	**ECPO**	<u>0.70</u>	**0.65**	**0.58**	**0.55**	<u>0.94</u>
R	SFT	0.53	0.25	0.21	0.19	0.71
R	GRPO	0.57	0.32	0.27	0.24	0.78
R	GRPO+Cert	0.59	0.38	0.33	0.29	0.87
R	RM-only+Det	**0.63**	<u>0.40</u>	<u>0.36</u>	0.30	**0.97**
R	ECPO $-r_{\text{cycle}}$	0.59	0.34	0.29	<u>0.35</u>	0.91
R	**ECPO**	<u>0.61</u>	**0.49**	**0.40**	**0.42**	<u>0.92</u>

Utility–audit trade-off. Table 1 makes the trade-off explicit. ECPO-A reaches VCERTNDCG@10 of $0.58/0.40$ and SLOT-EVIDCONS@10 of $0.65/0.49$, but RM-only+Det has higher ordinary NDCG ($0.72/0.63$ versus $0.70/0.61$). Removing r_{cycle} changes VCERTNDCG from 0.58 to 0.43 and 0.40 to 0.29, while NDCG changes only from 0.70 to 0.68 and 0.61 to 0.59. This isolates the effect of evidence coupling.

Table 2: **Construct-validity audits** on MAVEN-ERE/RAMS. Panel A reports vCERTNDCG under evaluators A/B/C; Panel B removes positions and candidate–bundle pairing. Slash-separated entries are M/R; scalar entries are pooled retained diagnostics.

Panel A: cross-verifier vCERTNDCG@10 (M/R)			
Training run	Eval. A	Eval. B	Eval. C
GRPO+Cert	0.45/0.33	0.41/0.28	0.40/0.27
ECPO-A	**0.58/0.40**	**0.55/0.38**	**0.53/0.36**
ECPO-B	0.56/0.39	0.54/0.37	0.52/0.36

Panel B: rank-blind recovery (M/R)					
Method/eval.	SetRec.	RankRec.	PairAcc.	τ_b	AbstainRate
GRPO/B	0.50/0.37	0.34/0.24	0.62/0.56	0.31/0.21	**0.08**
RM-only+Det/B	0.58/0.43	0.41/0.30	0.66/0.60	0.39/0.28	0.22/0.32
ECPO-A/B	**0.70/0.55**	**0.55/0.42**	**0.75/0.69**	**0.53/0.41**	**0.16/0.24**
ECPO-A/C	0.67/0.52	0.52/0.39	0.73/0.67	0.49/0.38	0.12

Held-out and rank-blind evaluation. Table 2 shows that primary ECPO-A retains vCERTNDCG@10 of $0.55/0.38$ under B and $0.53/0.36$ under C. Its advantage over RM-only+Det is $+0.07[0.03, 0.11]/+0.04[0.01, 0.07]$ under B and $+0.06[0.02, 0.10]/+0.03[0.01, 0.06]$ under C. We report the retained paired gains for these evaluations. Reverse-transfer ECPO-B obtains $0.56/0.39$ under A and $0.52/0.36$ under C. In rank-blind reconstruction, ECPO-A improves RANKRECOVER over RM-only+Det by $+0.14[0.08, 0.20]$ and $+0.12[0.06, 0.18]$ while abstaining less. The rank-blind audit measures output recovery.

Counterfactual evidence. Same-candidate random spans retain target recovery of $0.52/0.40$, confirming that candidate identity can sometimes be recovered from arbitrary candidate-linked evidence. Yet RANKRECOVER falls from $0.55/0.42$ to $0.25/0.18$, and Verifier-B vCERTNDCG falls from $0.55/0.38$ to $0.30/0.21$. Replacing one bundle with a length- and stage-matched valid bundle for an unselected same-window candidate reduces original-target recovery to $0.14/0.11$. Appendix F reports all controls. These results motivate our separation of traceability from comparative support.

Table 3: **External validation** (MAVEN-ERE/RAMS unless stated). Panel A is the blinded human audit. Panel B reports matched within-setting vCERTNDCG@10 under each setting's policy-held-out evaluator.

Panel A: human audit (M/R)			
Method	Comparative support	Set support	Rank support
GRPO	0.55/0.47	0.51/0.39	0.36/0.27
RM-only+Det	0.64/0.59	0.60/0.46	0.43/0.32
ECPO	**0.76/0.69**	**0.72/0.56**	**0.57/0.44**

Panel B: backbone/schema transfer			
Setting and baseline	Baseline	ECPO	Gain
Llama-3.1-8B-Instruct, M / GRPO	0.37	**0.49**	+0.12
Llama-3.1-8B-Instruct, M / RM-only+Det	0.42	**0.49**	+0.07
Llama-3.1-8B-Instruct, R / GRPO	0.26	**0.35**	+0.09
Llama-3.1-8B-Instruct, R / RM-only+Det	0.31	**0.35**	+0.04
Qwen3-14B, M / GRPO	0.41	**0.53**	+0.12
Qwen3-14B, R / GRPO	0.28	**0.38**	+0.10
HotpotQA / GRPO	0.31	**0.39**	+0.08
HotpotQA / RM-only+Det	0.35	**0.39**	+0.04

Human support and transfer. The blinded audit covers 120 stratified test windows and 1,080 bundles per method (540 per dataset), totaling 3,240 base method–bundle units. Annotators see neither method, scores, labels, stable IDs, nor ranks; ambiguity counts as failure. ECPO's human comparative support is $0.76/0.69$, a gain over RM-only+Det of $+0.12[0.07, 0.17]/ + 0.10[0.05, 0.15]$. Comparative-support agreement is Krippendorff's $\alpha = 0.71$, and a stratified 20% subset (648/3,240

units) is independently repeat-annotated. Table 3 also shows persistence across Llama-3.1-8B-Instruct, Qwen3-14B, and the HotpotQA two-hop provenance contract. On HotpotQA, the vCERT-NDCG gain is $+0.08[0.03, 0.12]$ over GRPO and $+0.04[0.01, 0.07]$ over RM-only+Det. These tests show gap persistence in the evaluated backbones and provenance-preserving schemas.

Sensitivity, extractor ceilings, and cost. Across 47 evaluation-only Verifier-A configurations, vCERTNDCG ranges over 0.55–0.60 on MAVEN-ERE and 0.38–0.42 on RAMS; method ordering is preserved in 45/47 and 43/47 settings. Fixed-roster predicted-evidence oracles reach $0.72/0.54$, source-normalized oracles $0.86/0.76$, a higher-recall extractor $0.64/0.48$, and 20% predicted-event deletion $0.50/0.33$. Missing upstream support cannot receive verifier credit. Beyond the shared extractor, the reward model costs 8 GPU-hours. RM-only+Align takes 0.18 s/window; ECPO adds 76 GPU-hours of policy optimization per corpus-seed and takes 3.1 s/window for one decode. NDCG and latency remain explicit trade-offs.

7 LIMITATIONS

All primary experiments use a closed finite roster, a fixed extractor, candidate-linked provenance, predefined skeleton/role semantics, and an automatic support contract. Verifier-B and Verifier-C change evaluator form while retaining the roster and broad ontology. The two primary datasets are event-centric, and HotpotQA adds one provenance schema. Automatic scores are verifier-relative, the human audit is finite, upstream representation bounds available evidence, and ECPO has higher cost than deterministic ranking.

8 CONCLUSION

We presented an evidence-coupled ranking interface, training objective, and audit protocol. In the tested closed-roster provenance settings, ECPO improves verifier-audited traceability and rank-blind and human-assessed comparative support relative to matched controls, including under held-out evaluators. Relative to the fixed-weight slot-cycle formulation, full ECPO improves rank-blind recovery by $+0.09/+0.08$ and verifier-audited NDCG by $+0.04/+0.04$ on MAVEN-ERE/RAMS, while ordinary NDCG changes by $-0.01/-0.01$. The stronger deterministic ranker's ordinary NDCG and cost make the deployment trade-off explicit: ECPO is appropriate when ranking and evidence must be generated as one auditable object.

AI USE STATEMENT

We used generative AI tools for language polishing and reference summarization. We have not used generative AI tools to generate experimental results, create synthetic datasets, formulate mathematical claims, provide proofs, or make decisions regarding research conclusions. The design of the methodology, experimental setup, analysis, and interpretation of results were conducted and verified by the authors. Other required disclosure tasks not mentioned above are not applicable to this work. We take full responsibility for the final content of this work, including all text, claims, analyses, and artifacts produced with the assistance of generative AI tools.

ETHICS STATEMENT

The completed blinded audit sampled 120 stratified test windows, covering 1,080 bundles per method and 3,240 method–bundle units overall. A stratified 20% subset (648/3,240) was independently repeat-annotated, with comparative-support Krippendorff's $\alpha = 0.71$. Annotators were blinded to method identity, model/evaluator scores, labels, stable IDs, and submitted ranks; ambiguity counted as failure. They received written instructions and risk/confidentiality information, were paid a fixed hourly rate above the local minimum wage, and received no performance bonus. An institutional minimal-risk determination found that full IRB review was not required. ECPO is intended for decision support with human review, access controls, logging, bias/drift audits, and appeal or correction channels in sensitive deployments. The audit uses anonymized benchmark records and reports no personal identifiers.

Reproducibility statement

Sections 3–5 define the task, verifier, objective, baselines, and split protocol. Appendices A, D, J, and P give data construction, exact edge cases, statistical protocols, and the artifact completion ledger. The implementation and supplementary code are included in the Supplementary Material.

References

Pepa Atanasova, Oana-Maria Camburu, Christina Lioma, Thomas Lukasiewicz, Jakob Grue Simonsen, and Isabelle Augenstein. Faithfulness tests for natural language explanations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 283–294, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.25.

Tobias Brockhoff, Malte Heithoff, István Koren, Judith Michael, Jérôme Pfeiffer, Bernhard Rumpe, Merih Seran Uysal, Wil M. P. van der Aalst, and Andreas Wortmann. Process prediction with digital twins. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pp. 182–187. IEEE, 2021. doi: 10.1109/MODELS-C53483.2021.00032.

Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, Redmond, Washington, 2010. URL https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

Yu-Neng Chuang, Guanchu Wang, Chia-Yuan Chang, Ruixiang Tang, Shaochen Zhong, Fan Yang, Andrew Wen, Mengnan Du, Xuanting Cai, Vladimir Braverman, and Xia Hu. FaithLM: Towards faithful explanations for large language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3802–3824, Rabat, Morocco, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.177.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115. Curran Associates, Inc., 2023. doi: 10.52202/075280-0441. URL https://arxiv.org/abs/2305.14314.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8057–8077, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.718.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. LLMs gaming verifiers: RLVR can lead to reward hacking. arXiv preprint arXiv:2604.15149, 2026.

Benjamin W. K. Hung and Anura P. Jayasumana. Investigative simulation: Towards utilizing graph pattern matching for investigative search. arXiv preprint arXiv:1608.01760, 2016.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4198–4205, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.386.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. doi: 10.1145/582415.582418.

Yifei Li, Xiang Yue, Zeyi Liao, and Huan Sun. AttributionBench: How hard is automatic attribution evaluation? In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14919–14935, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.886.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.173.

Meta. Llama 3.1 Model Card. Meta Llama model documentation, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md. Released July 23, 2024.

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. Curran Associates, Inc., 2022. doi: 10.52202/068431-2011.

Gyunam Park and Wil M. P. van der Aalst. Realizing a digital twin of an organization using action-oriented process mining. In *2021 3rd International Conference on Process Mining (ICPM)*, pp. 104–111. IEEE, 2021. doi: 10.1109/ICPM53251.2021.9576846.

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. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338.

Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 1778–1783. Morgan Kaufmann Publishers Inc., 2009. URL https://www.ijcai.org/Proceedings/09/Papers/296.pdf.

Miquel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pp. 1121–1126. AAAI Press, 2010. doi: 10.1609/aaai.v24i1.7745.

Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. Measuring attribution in natural language generation models. *Computational Linguistics*, 49(4):777–840, 2023. doi: 10.1162/coli_a_00486.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, San Francisco, California, USA, 2016. ACM. doi: 10.1145/2939672.2939778.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

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.

August Daniel Sutmuller, Marielle den Hengst, Ana Isabel Barros, and Pieter van Gelder. Getting the perpetrator incorporated and prioritized in homicide investigations: The development and evaluation of a case-specific element library (C-SEL). *International Journal of Environmental Research and Public Health*, 17(17):6430, 2020. doi: 10.3390/ijerph17176430.

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. MAVEN: A massive general domain event detection dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1652–1671, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.129.

Xiaozhi Wang, Yulin Chen, Ning Ding, Hao Peng, Zimu Wang, Yankai Lin, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. MAVEN-ERE: A unified large-scale dataset for event coreference, temporal, causal, and subevent relation extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 926–941, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.60.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Wenhao Yuan, Chenchen Lin, Jian Chen, Jinfeng Xu, Xuehe Wang, and Edith Cheuk-Han Ngai. Verify before you commit: Towards faithful reasoning in LLM agents via self-auditing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31201–31225, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1440.

Kerem Zaman and Shashank Srivastava. A causal lens for evaluating faithfulness metrics. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 29425–29449, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1496.

Weijia Zhang, Mohammad Aliannejadi, Yifei Yuan, Jiahuan Pei, Jia-Hong Huang, and Evangelos Kanoulas. Towards fine-grained citation evaluation in generated text: A comparative analysis of faithfulness metrics. In *Proceedings of the 17th International Natural Language Generation Conference*, pp. 427–439, Tokyo, Japan, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.inlg-main.35.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1433–1438. AAAI Press, 2008. URL https://cdn.aaai.org/AAAI/2008/AAAI08-227.pdf.

APPENDIX CONTENTS

A. Detailed Task, Data, and Construction Protocol 13
A.1 Problem framing details 13
B. Frozen Extractor and Trajectory Construction 29
C. Minimal Interface and Preconditions 30
D. Exact Aligner and Verifier Specification 30
E. Rank-Blind Metric Edge Cases 32
F. Counterfactual Evidence Controls 32
G. Sensitivity and Upstream Ceilings 33
H. Human Audit Protocol 33
I. Additional Experimental and Cost Details 33
J. Metric Semantics and Statistical Protocol 34
J.1 What each metric establishes 34
K. Held-Out Evaluators and Transfer Protocol 35
L. Matched Baselines, Optimization, and Implementation Details 39
M. Additional Diagnostic Experiment Specifications 40
M.1 Rank-blind coupling and robustness results 40
N. Error Taxonomy and Qualitative Audit Format 43
O. Extended Positioning and Risk Boundaries 44
P. Reproducibility and Artifact Checklist 45

A Detailed Task, Data, and Construction Protocol

This appendix restores the task and data details needed to reconstruct the evaluated closed-roster interface. The current ICLR specification is authoritative: a policy-visible record contains no relevance label, source-normalized chain, audit reference, stable cross-window candidate identifier, or verifier answer. The appendix records the policy-visible fields and evaluation contract.

Source separation. For every window, three views remain distinct. The *source-normalized view* supplies train-only supervision and held-out relevance/audit references. The *model-input view* is produced by a frozen extractor and supplies the trajectories visible to every ranking method. The *policy-output view* contains only the generated ordered roster IDs and evidence bundles. No source-normalized span, chain-validity flag, or candidate label is copied into a policy prompt. The evaluator may join the views by hidden window and candidate keys only after the output has been fixed.

Compatibility rule for restored statistics. The descriptive statistics below are inherited only where the old and current manuscripts declare the same datasets, document-level split discipline, window construction, candidate resolver, and frozen-extractor interface. They are independent of the changed Verifier-A score.

A.1 Problem framing details

This subsection expands the brief main-text discussion of the intent, skeleton, and window-graph abstractions.

Ranking windows and context snippets. A ranking window is the evidence scope for one Top-K candidate-ranking decision. It is not necessarily a calendar-time interval. A context snippet is a local text segment used for upstream extraction or compact prompting. In MAVEN-ERE, source-record snippets used for supervised extractor examples and compact evidence display are trigger-centered sentence neighborhoods. In RAMS, compact source snippets are document prefixes. Held-out extractor inference does not use gold trigger locations: the frozen extractor is applied to split documents or document-local text units, and predicted event spans are linked back to their surrounding local context for prompt compaction and auditing. The number of snippets and documents contributing to each ranking window is an empirical property reported in Appendix A.4 and Appendix Table 8.

Intent and plan skeleton. In this paper, an *intent* denotes a hypothesized multi-step activity type that we instantiate with an explicit plan skeleton. In our benchmark instantiations, we use a four-stage inventory `PREP`→`PROBE`→`EXECUTE`→`OUTCOME` as the canonical soft order. Individual `intent_id` skeleton records instantiate ordered steps, role requirements, and optional soft-precedence edges over this inventory. We operationalize an intent with an explicit *plan skeleton* S, which is an *intent-level* abstraction shared by all windows labeled with the same `intent_id`. Importantly, S is *not* constructed by directly taking any single candidate's observed event sequence as the skeleton; instead, it specifies high-level step *types* and key role constraints, and individual candidates are only partially observed realizations that may miss steps, contain spurious events, or have uncertain timestamp / document-order indices. Ordering irregularities are captured by soft precedence-violation features after monotone alignment, not by arbitrary event transpositions in the DP recurrence. In the main benchmark setting, each `intent_id` is associated with a single predefined soft-precedence plan skeleton. We allow missing steps and skipped events during alignment to handle partial observability. To examine the effect of this assumption, Appendix M.6 reports diagnostic skeleton-perturbation analyses. When reported, Appendix M.6 treats skeleton banks only as an auxiliary diagnostic; they are not part of the main method or main-result comparison. Automatically discovering skeletons from raw text remains outside the scope of this paper. Table 4 shows an illustrative skeleton record; it is schematic and does not expose any held-out labels or audit references.

Additional details on candidate identities, shared documents, and evidence-span serialization are provided in Appendix A.2.

Table 4: Illustrative predefined skeleton record. Stage labels are abstract alignment labels; `OUTCOME` denotes the terminal or outcome stage in the declared JSON schema.

Step	Stage	Required roles / soft-precedence description
s_1	`PREP`	optional `Agent`; precedes later evidence-bearing steps when order is available
s_2	`PROBE`	`Agent` and optional `Target`; soft-precedes execution evidence
s_3	`EXECUTE`	`Agent` and `Target`; central action step for alignment
s_4	`OUTCOME`	optional `Target`/`Context`; terminal or outcome evidence

This paper focuses on the ranking-and-audit interface itself: candidate prioritization is evaluated only through window-local evidence, randomized candidate identifiers, and auditable span certificates.

A.2 Candidate identities, document sharing, and evidence serialization

In this paper, a *candidate* is an anonymized entity cluster placed into a window-local candidate pool by a label-free resolver over role-bearing event arguments. When no external roster is available, the main benchmark first fixes an eligible closed roster from an identifier-only source projection of role-bearing entities. The resolver then links frozen-extractor predicted mentions only to this eligible roster; predicted-only entities absent from the closed roster are excluded from the main benchmark and are analyzed separately in the predicted-only and hybrid-roster diagnostics. The candidate pool C_w is local to a ranking window. For all policy prompts and generated outputs, candidate identifiers are randomized and window-local (for example, `cand_001`); source-normalized entity IDs are retained only in hidden evaluator metadata. Ranking, validation, and metric computation therefore compare candidates within the current C_w and do not allow cross-window ID memorization. Identifier controls, pooled over both datasets, are reported in Appendix M.5. Although motivating applications often involve persons of interest, MAVEN-ERE and RAMS contain heterogeneous event participants; therefore, the declared benchmark uses candidate entities after deterministic role normalization. Candidate pools are constructed from entities that participate in agent-like or target-like roles, while purely contextual entities are not used as candidates unless they also appear in a candidate role.

A document may mention multiple candidates; therefore, the same document $d \in \mathcal{D}_w$ can contribute evidence spans to multiple candidates. Likewise, a single predicted event record may appear in multiple candidate-centric trajectories if multiple candidates participate in its arguments. Each trajectory remains auditable via its own `doc_id:span` references.

We represent an evidence span conceptually as $\sigma = (\texttt{doc_id}, [l, r))$, where $[l, r)$ is a 0-indexed half-open character-offset interval. In JSON, the bounds are serialized either as the two-integer field `span:[l,r]` or as the string `"l-r"`. In both cases, the stored pair (l, r) is interpreted by the validator as the same half-open interval $[l, r)$; the array notation is a serialization of endpoints rather than a closed mathematical interval.

A.3 Detailed problem notation and event representation

This subsection collects notation and serialization details moved out of the main text. For a ranking window w, $C_w = \{c_{w,1}, \ldots, c_{w,N_w}\}$ denotes the window-local candidate pool, and $\tau_w(c_{w,i}) \equiv \tau_{w,i}$ denotes the candidate trajectory within w. If no predicted event is available for a roster candidate, we retain an empty trajectory record rather than dropping the candidate. Events are sorted by absolute time when available and otherwise by dataset-native or document-local order.

Each event is represented as

$$e = \big(\text{etype}^{\text{raw}}, \mathcal{H}^{\text{skel}}, \text{etype}^{\text{primary}}, \sigma^{\text{trig}}, \mathcal{A}, t\big), \tag{17}$$

where $\text{etype}^{\text{raw}}$ is the extractor's fine-grained event type, $\mathcal{H}^{\text{skel}} = \texttt{skeleton_hits}(e)$ is the set of compatible skeleton stages, and $\text{etype}^{\text{primary}}$ is a single primary stage stored only for serialization and analysis. Alignment and reward computation use $\texttt{skeleton_hits}(e)$ rather than the primary stage label. The argument set $\mathcal{A}$ contains tuples $(\texttt{role}, \texttt{entity_id}, \sigma^{\text{arg}})$. The temporal field t is an absolute timestamp when available, and otherwise a deterministic relative or document-local order key. Serialized trajectory records may expose both an optional `time` field and a required `order_index`; sorting prioritizes reliable absolute time and falls back to `order_index`. All triggers and arguments preserve auditable provenance via their spans σ.

Table 5: Core notation.

Symbol	Meaning
w	ranking window / evaluation instance
$\texttt{intent_id}(w)$	intent identifier for window w
S_w	predefined skeleton associated with $\texttt{intent_id}(w)$
C_w	candidate pool in window w
N_w	number of candidates, $\lvert C_w \rvert$
$\tau_w(c_{w,i}) \equiv \tau_{w,i}$	trajectory of candidate $c_{w,i}$ within window w
K_w	evaluated cutoff, $\min(K, N_w)$
$\phi(\tau; S)$	skeleton, role, precedence, and temporal feature vector
$R_\theta(\tau; S_w)$	learned skeleton-conditioned trajectory-level reward
y	generated JSON output containing Top-K_w IDs and certificates

For target cutoff K, the evaluated list length is $K_w = \min(K, N_w)$. When a candidate pool contains fewer than K candidates, the expected output contains all candidates in that pool. Each returned candidate has one certificate object containing one serialized step object for every skeleton step in S_w; matched steps cite supporting `doc_id:span` evidence and unmatched steps are recorded explicitly with empty evidence.

A.4 Dataset construction and preprocessing details

Relation to the condensed main-text setup. For space, the main text compresses task instantiation, datasets, extractor pretraining, reward-learning corpus construction, models/baselines, and metrics into a single experimental-setup subsection. This appendix, together with Appendix A.11 and Appendix B, provides the corresponding full details.

Upstream event supervision (extractor only). Event extraction is an upstream preprocessing step. We pretrain the extractor with a staged curriculum: `MAVEN` $\rightarrow$ `MAVEN-Arg` $\rightarrow$ `MAVEN-ERE` $\rightarrow$ `RAMS` to improve argument-role consistency and reduce trajectory noise; details are in Appendix B. This curriculum affects only the extractor; the *ranking benchmark instantiations* in this paper are built on **MAVEN-ERE** and **RAMS**.

Two evaluation instantiations. We instantiate the ranking benchmark in two variants derived from the source corpora **MAVEN-ERE** and **RAMS**. Source-normalized records from these corpora define supervision, positive labels, and audit references, while model-input trajectories are constructed separately from frozen-extractor predictions. For each ranking window w, we aggregate predicted event records by candidate identity, normalize timestamps when available, and otherwise preserve dataset-native or document-local order. This yields a possibly empty predicted trajectory $\tau_{w,i}$ with auditable provenance `doc_id:span` for triggers and arguments.

Windows and trajectory instances. Each evaluation instance is a *ranking window* w that defines the evidence scope $\mathcal{D}_w$, the window-local candidate pool C_w, source-normalized positive labels, audit references, and model-input trajectories for one Top-K ranking prompt. A trajectory is *window-scoped*: it is defined for a specific (w, c) pair, i.e., $\tau_{w,c}$ aggregates candidate-participating events that are observable within $\mathcal{D}_w$ and orders them by time or by dataset-native / document-local order when absolute time is unavailable.

We distinguish two event-record views during construction. Source-normalized records are deterministic conversions of the dataset annotations. They are used to derive positive candidate labels and reference evidence for audit metrics. When no external candidate roster is available, we construct the benchmark candidate roster from an identifier-only projection of source-normalized role-bearing entities before relevance labels are attached. This roster-construction step keeps only anonymized entity identifiers and document/window membership needed to define the candidate universe. It discards event-stage labels, skeleton-chain validity, positive labels, reward scores, and reference evidence chains. Thus, source-normalized records may define which anonymized entities are eligible candidates, but they do not provide model-input trajectories or ranking evidence. Predicted records are produced by the frozen event extractor and are the only event records consumed by the reward model, ranking policy, RM-only ranker, and structured baselines. The candidate

pool C_w is fixed before positive labels are attached, and positive candidate IDs are stored only in `window_label.jsonl`; they are not included in model-input records.

The same underlying source entity may appear in multiple ranking windows when a cross-window identifier is available. However, ranking decisions are made over window-local candidate IDs. Thus, the number of window-scoped trajectory instances can exceed the number of unique source entities, and overlapping windows may reuse portions of the same document evidence. Appendix Algorithm 1 summarizes the construction procedure.

Instantiation-specific context snippets. **MAVEN-ERE.** We construct one fixed *trigger-centered context snippet* per event record used for extraction or compact prompting (no sliding stride): we locate the sentence containing the trigger and take at most three sentences in total (the trigger sentence plus up to one sentence before and one after; `context_radius`=1). **RAMS.** Upstream event records use dataset-native context construction. For extractor/SFT text fields, we flatten sentences into a document string and keep its first 120 characters as the local context snippet. Ranking-policy prompts additionally include the `intent_id`, skeleton, candidate IDs, and compact trajectory summaries or evidence pointers.

The full construction (including fallbacks when sentence boundaries are missing) and leakage-preventing splits are described in Appendix A.5–A.6.

Dataset statistics. The construction produces four logically separate record families: `window_input.jsonl` for window scopes, selected skeleton identifiers, and candidate pools, `traj_pred.jsonl` for frozen-extractor model-input trajectories, `window_label.jsonl` for positive candidate labels, and `audit_ref.jsonl` for source-normalized step-level audit references. This separation is important because all ranking methods consume the same window inputs and predicted trajectories, while labels and audit references are used only for training-split supervision, model selection, or held-out evaluation.

We construct window-scoped candidate-centric trajectories, hard/near-neighbor negatives, perturbation-based negatives, and within-window preference supervision. Table 6 summarizes key statistics for both instantiations; extended distributions and metadata are provided in Appendix Table 14 and Appendix Table 15.

Table 6: Overview of the two benchmark instantiations after preprocessing. *Note:* Candidate IDs are anonymized entity-cluster identifiers before window expansion. The Top-K task is defined over window-local candidate pools C_w, not over the global candidate-ID inventory. Rows that summarize train-side supervision or augmentation are reported separately from held-out ranking-window counts. Table 8 reports the ranking-window statistics that determine the actual candidate-ranking task.

Statistic	MAVEN-ERE	RAMS
Documents	3,623	9,124
Source-normalized event records	91,222	9,124
Unique predicted event records contributing to model-input trajectories	96,184	9,386
Unique source-level candidate/entity IDs before window expansion	3,623	8,730
Serialized trajectory-pool records across ranking / reward-learning files	7,245	8,864
Train-split preference pairs	3,622	134
Train-split RL prompt records (augmented)	171,584	654

Source-normalized event records denote dataset-native event annotations after deterministic normalization and schema conversion. They are used to construct training-split supervision and held-out reference labels / audit references. When no external candidate roster is available, an identifier-only projection of these records is also used to construct anonymized role-bearing candidate IDs before relevance labels are attached. Model-input trajectory records are separate: the reward model, ranking policy, RM-only ranker, and structured baselines consume only fixed-extractor predictions stored in `traj_pred.jsonl`. Therefore, statistics that describe model inputs are reported under "predicted" records, while statistics that describe annotation-derived references are explicitly marked as "source-normalized". The serialized-trajectory row in Table 6 counts the trajectory-record pool declared in the retained manuscripts across ranking and reward-learning files. For MAVEN-ERE, this pool contains comparison-pool candidate trajectories, expert/pseudo-expert demonstrations, and hard/perturbation negatives; for RAMS, it contains window-scoped candidate trajectories together

with a small number of train-only hard/perturbation negatives (Appendix A.11). These counts summarize declared trajectory records rather than held-out ranking instances, which are counted separately by ranking windows in Table 8. The train-split preference-pair and RL-prompt rows are likewise reported for reproducibility of reward/policy training rather than to characterize the held-out ranking task. For dev/test ranking inputs, trajectories are built from frozen-extractor predictions; held-out source annotations are used to define positive labels and audit references.

Predicted-evidence coverage of positive candidates. Because held-out positives are defined from source-normalized reference chains while model inputs are frozen-extractor predictions, some positive candidates may have incomplete predicted evidence. We therefore compute a permissive candidate-wise evidence-availability diagnostic: for each held-out positive (w, c), we check within the same predicted trajectory $\tau_{w,c}$ whether at least one, at least two, or all source-normalized reference steps have a type-compatible predicted event with valid `doc_id:span` provenance. This diagnostic reports availability under a permissive resolver. Table 7 reports this diagnostic under the same split, provenance, and validation protocol as the main benchmark.

Table 7: Predicted-evidence availability for held-out positive candidates. A step is counted as available if a type-compatible predicted event with valid `doc_id:span` provenance is present in the candidate trajectory. This reports availability under a permissive resolver. Because reference chains may contain a single step, especially in RAMS, the "all reference steps covered" row is not necessarily a monotone refinement of the "at least two covered steps" row.

Coverage diagnostic	MAVEN-ERE	RAMS
At least one covered step	0.96	0.89
At least two covered steps	0.86	0.34
All reference steps covered	0.66	0.41

Table 8: Ranking-instance statistics. A ranking window is the evaluation unit for Top-K candidate ranking.

Statistic	MAVEN-ERE	RAMS
Ranking windows	718	896
Train / dev / test windows	576/71/71	716/90/90
Intent IDs	16	12
Skeleton records	16	12
Avg. candidates per window	10.1	9.7
Median candidates per window	9	9
Avg. positives per window	1.3	1.1
Windows with multiple positives (%)	18.1	7.4
Windows with $\lvert C_w \rvert < 10$ (%)	46.5	49.7
Avg. exposed evidence units per window	12.6	2.4
Windows with evidence from multiple documents (%)	61.2	11.8
Avg. trajectory-expanded predicted event mentions per trajectory	24.7	1.07

The row "unique predicted event records" counts each frozen-extractor event record once. The trajectory-length statistics count trajectory-expanded event mentions after window and candidate expansion. Therefore, the same predicted event record may contribute to multiple candidate trajectories when multiple candidates participate in the event or when overlapping windows reuse the same evidence. An exposed evidence unit is a trigger-centered context snippet for MAVEN-ERE and a document-prefix / document-local context unit for RAMS; multi-document evidence is computed from distinct `doc_id` values.

RL prompt records. RL training prompt records denote policy-optimization prompts generated from eligible train-split windows by varying candidate subsets, demonstration order, and hard-negative composition. They are not the number of distinct evaluation windows and are not expected to equal the number of train windows: after eligibility filtering, the count can be smaller than the train-window count in sparse settings or much larger when augmentation is dense. Distinct ranking-window counts are reported in Table 8.

A.5 WINDOWING PROTOCOL

A *ranking window* w specifies the evidence scope for one Top-K candidate-ranking instance and has an associated candidate pool, separately stored positive labels, and window-scoped trajectories. The model-input window record stores the evidence scope, candidate pool, `intent_id`, and selected `skeleton_id`, while positive labels are stored only in `window_label.jsonl`. Separately, a *context snippet* specifies the local text segment used for upstream extraction or compact prompting. Because the source corpora do not uniformly provide reliable calendar-time intervals, ranking windows are not defined by calendar slicing. Instead, both window construction and snippet construction follow dataset-native document or document-local grouping rules.

MAVEN-ERE context snippets (trigger-centered; `context_radius=1`). For source-normalized records and supervised extractor examples, we construct exactly one fixed trigger-centered context snippet (no sliding stride):

1. Locate the sentence containing the trigger via `resolve_sent_id`: use the annotated sentence ID when available; otherwise fall back to trigger-text matching.
2. Take sentences in the index range $[\texttt{sent_id} - \texttt{context_radius},\ \texttt{sent_id} + \texttt{context_radius}]$ with `context_radius`=1, yielding at most three sentences.
3. If the range is empty, fall back to the trigger sentence; if still empty, fall back to the concatenation of all non-empty sentences in the document; if still empty, fall back to the trigger text.

The final context is the space-joined concatenation of the selected sentences (a coarse token count is computed by whitespace splitting). This procedure is implemented in `build_context_snippet(...)` and used by For held-out extractor inference, trigger-centered source snippets are not used to choose where the extractor looks. The frozen extractor is applied to the allowed split documents or document-local text units without gold trigger positions. After prediction, each predicted trigger span is associated with its surrounding sentence neighborhood only for compact prompt construction and span auditing.

RAMS context snippets (document prefix; first 120 characters). RAMS does not use sentence-based context snippets for upstream event-record construction. We flatten the document sentences into a single document string `doc_text` and use `doc_text[:120]` (the first 120 characters) as the local text field for extractor/SFT sample construction. No sliding stride is used. This procedure is implemented in `build_events` and consumed by `build_sft_samples` via `event.mapping.get("doc_text", "")`.

Ranking-policy prompts are constructed separately from ranking-window records. They include the `intent_id`, the skeleton, candidate IDs, compact trajectory summaries, and indexed evidence pointers. Thus, the 120-character RAMS prefix is a context-snippet construction choice, not the full evidence interface for ranking-policy decoding.

Window records. Each window record stores at least `window_id`, `doc_ids`, optional `snippet_ids`, the candidate pool `candidate_ids`, the associated `intent_id`, and the selected `skeleton_id`. A candidate can appear in multiple ranking windows. MAVEN-ERE context snippets can overlap when nearby triggers occur in adjacent sentence neighborhoods; this snippet overlap is distinct from the definition of ranking windows. The window record, together with the corresponding entries in `traj_pred.jsonl`, forms the model-input unit consumed by ranking methods; context snippets are lower-level text segments used to create event records and compact prompt evidence.

The serialized `intent_id` remains the conditioning variable and is checked against the linked skeleton record during prompt construction, reward-feature computation, and validation.

A.6 SPLITS AND LEAKAGE PREVENTION

To reduce leakage across splits, we use document-level splits, with time-aware constraints applied only when reliable timestamps are available:

Algorithm 1 Deterministic construction of ranking windows and candidate trajectories.

Require: Documents $\mathcal{D}$; source annotations $\mathcal{A}^{\text{src}}$; trained frozen extractor f_{ext}; deterministic intent mapper m_{intent}; skeleton table $\mathcal{S}_{\text{file}}$; label-free candidate resolver ρ_{id}; window grouping rule g.

Ensure: `window_input.jsonl`, `traj_pred.jsonl`, `window_label.jsonl`, and `audit_ref.jsonl`.

Use document-level train/dev/test splits fixed before extractor training, reward learning, policy optimization, model selection, and evaluation.

Construct corpus-specific local text fields or snippet metadata used for preprocessing and prompt serialization, following the context-snippet rules above.

For held-out extractor inference, do not use gold trigger positions; apply f_{ext} only to allowed split documents or document-local text units.

Run the frozen extractor f_{ext} on all allowed split documents to obtain $\mathcal{E}^{\text{pred}}$ with `doc_id:span` provenance.

Convert source annotations $\mathcal{A}^{\text{src}}$ into source-normalized records $\mathcal{E}^{\text{src}}$ for supervision and held-out references.

Derive $\mathcal{E}^{\text{src}-\text{id}}$ by retaining only anonymized role-bearing entity identifiers, document/snippet membership, and any corpus-native grouping keys required by g; remove labels, validated chains, and reference spans.

for each grouping key q induced by g **do**

  Create a candidate ranking window w with evidence scope $\mathcal{D}_w$, consisting of the documents or snippets assigned to q.

  Assign $\texttt{intent_id}(w) = m_{\text{intent}}(q)$ and select the predefined skeleton S_w from $\mathcal{S}_{\text{file}}$.

  Build a window-local candidate pool with the label-free resolver

$$C_w = \rho_{\text{id}}\big(\mathcal{E}_w^{\text{src}-\text{id}}, \mathcal{E}_w^{\text{pred}}\big),$$

  where $\mathcal{E}_w^{\text{src}-\text{id}}$ defines the eligible closed roster using only anonymized role-bearing entity identifiers and window-membership metadata, and $\mathcal{E}_w^{\text{pred}}$ is used only to attach predicted evidence and trajectories to those eligible candidates. Predicted-only entities that do not map to the closed roster are excluded from the main benchmark and are analyzed separately in the predicted-only roster diagnostic.

  The resolver does not use skeleton-chain validity, positive labels, reward scores, or audit reference spans.

  for each candidate $c \in C_w$ **do**

    Collect candidate-participating predicted events from $\mathcal{E}_w^{\text{pred}}$.

    Sort them by absolute timestamp when available; otherwise sort by dataset-native or document-local order.

    Serialize the resulting model-input trajectory as $\tau_{w,c}$ in `traj_pred.jsonl`, preserving trigger and argument `doc_id:span` provenance.

    If no predicted event is found for c, serialize an empty trajectory with the candidate ID and window ID retained; empty trajectories remain valid model inputs and are scored by missing-step features.

  end for

  Serialize `window_id`, `intent_id`, the selected `skeleton_id`, $\mathcal{D}_w$, and C_w in `window_input.jsonl`; the conceptual model input x_w includes the resolved skeleton S_w, obtained under the declared record contract by deterministically joining `window_input.jsonl` with `skeleton.jsonl` on `skeleton_id`.

  Derive positive candidates $\mathcal{P}_w \subseteq C_w$ from source-normalized skeleton-aligned chains in $\mathcal{E}_w^{\text{src}}$.

  Store $\mathcal{P}_w$ in `window_label.jsonl`; do not store it in model-input files.

  Store source-normalized step-level reference spans in `audit_ref.jsonl` for audit metrics only.

end for

Discard any window whose documents cross split boundaries, or repair it before serializing records by removing cross-split documents and recomputing C_w, $\tau_{w,c}$, $\mathcal{P}_w$, and audit references.

Generate training-only perturbation-derived negatives and augmented RL prompts from train-split windows only.

- No document (`doc_id`) appears in multiple splits.
- A ranking window is assigned to a split only when all documents in its `doc_ids` field belong to that split.
- If a constructed ranking window would contain documents from multiple splits, we either remove cross-split documents from the window before trajectory construction or discard the mixed-split window; no mixed-split window is used for training or evaluation.
- When reliable time ranges are available, we avoid placing overlapping time ranges across splits whenever possible; the document-level constraint always dominates.

The intended artifact package requires explicit split lists for both documents and ranking windows: `splits/doc_{train,dev,test}.txt` contains document IDs, and `splits/window_{train,dev,test}.txt` contains window IDs. All perturbation-derived negatives, candidate-subset augmentations, and RL prompt variants are generated only after the split assignment and only from train-split windows unless explicitly used in a train-only diagnostic. Held-out dev/test ranking pools contain window-local candidates from `window_input.jsonl`, not synthetic perturbation candidates.

A.7 Candidate pools and labeling

For each ranking window w in train, dev, or test, the candidate pool C_w contains window-local candidate entity clusters that are observable in the window construction stage. The pool is fixed before positive labels are attached. For held-out dev/test windows, labels and audit references are used only for evaluation and are never serialized into model-input records. When source-normalized records are used by the candidate resolver, they contribute only anonymized role-bearing entity identifiers for constructing the candidate universe; source-normalized skeleton-chain validity, reference step spans, and positive labels are not used in model-input construction. The declared model-input record `window_input.jsonl` contains `window_id`, `intent_id`, the selected skeleton identifier, the exposed document or snippet identifiers, and the candidate IDs in C_w, but it does not contain positive labels, any field that marks a candidate as relevant, or reference evidence chains.

Candidate-roster diagnostic. The main benchmark uses an identifier-only source projection to define a closed candidate roster when no external candidate list is available. To quantify the dependence on this choice, we also evaluate a predicted-only roster diagnostic in which candidate pools are constructed from frozen-extractor predictions only. This diagnostic is not used as the main benchmark setting because upstream extraction may omit true held-out candidates entirely, making some windows unrankable rather than merely difficult. It separates candidate-availability errors from ranking errors by reporting retained-window coverage, positive-candidate recall in the predicted-only roster, and Top-K ranking scores on the retained subset. The diagnostic is computed only for analysis and is not used for reward learning, policy training, model selection, or the main test comparison.

Table 9: Predicted-only candidate-roster diagnostic. Retained windows are windows whose candidate pool remains non-empty under frozen-extractor-only roster construction. Positive-candidate recall measures whether held-out positive candidates remain available in the predicted-only roster.

Diagnostic	MAVEN-ERE	RAMS
Retained test windows	70/71 (98.6%)	89/90 (98.9%)
Positive-candidate recall in predicted-only roster	0.87	0.76
NDCG@10 on retained subset	0.63	0.54
MAP@10 on retained subset	0.48	0.39
Hit@10 on retained subset	0.76	0.66

A candidate $c \in C_w$ is labeled positive if there exists a validated source-normalized chain aligned to S_w and supported by `doc_id:span` evidence within the window. Formally, the positive set is

$$\mathcal{P}_w = \{c \in C_w : \exists\, z \in \mathcal{Z}^{\mathrm{src}}(w, c, S_w) \text{ such that } \mathrm{valid}(z) = 1\}\,,$$

where $\mathcal{Z}^{\mathrm{src}}(w, c, S_w)$ denotes source-normalized candidate chains that can be aligned to the skeleton S_w using evidence inside $\mathcal{D}_w$. Windows may contain multiple positives. We store positive candidate IDs only in `window_label.jsonl`. Reference step-aligned evidence used by audit metrics is

stored in audit_ref.jsonl. Model-input trajectories are constructed separately from frozen-extractor predictions. For each (w, c), traj_pred.jsonl stores the predicted trajectory $\tau_{w,c}$, including predicted event types, compatible skeleton stages, normalized arguments, ordering information, and trigger / argument doc_id:span provenance. Thus, the ranker may receive noisy or incomplete trajectories even when the source-normalized reference contains a positive chain.

Same-window near-neighbor negatives are non-positive candidates in C_w that share superficial context with positives, such as similar local event types, overlapping documents, entity-role patterns, or nearby temporal/order positions, but lack a validated skeleton-aligned chain. Perturbation-derived negatives, including insertion, deletion, time-shift, and role-swap variants, are used for reward-learning comparison sets and train-split augmented policy prompts. They are not mixed into dev/test ranking pools as ordinary entity candidates unless explicitly marked as synthetic train-only distractors.

Intent identifiers. Each intent_id is assigned at the ranking-window level by a deterministic preprocessing rule before reward learning, policy training, model selection, and evaluation. The identifier specifies the hypothesized plan type for the window and selects a skeleton record from skeleton.jsonl; all candidates in the same ranking window receive the same intent_id. It does not identify positive candidates, encode candidate-specific relevance, or expose reference evidence chains. In the declared instantiations, the window-to-intent_id mapping is stored as preprocessing metadata and is applied identically across train, dev, and test splits. Candidate relevance remains a separate held-out ranking label stored only in window_label.jsonl. In the main experiments, the number of skeleton records equals the number of distinct intent_id values unless explicitly stated otherwise.

A.8 Deterministic stage mapping from fine-grained event types to intent-stage labels

This appendix specifies the deterministic and reproducible rule used to map the extractor's fine-grained event types $\texttt{etype}^{\text{raw}}$ into the intent-stage labels $\{\texttt{PREP}, \texttt{PROBE}, \texttt{EXECUTE}, \texttt{OUTCOME}\}$ used by skeleton alignment and reward computation. The label OUTCOME denotes an outcome or terminal stage for both MAVEN-ERE and RAMS. A domain-specific deployment may refine this stage name without changing the alignment or reward-learning algorithms. The mapping is applied during preprocessing and the resulting stage labels are stored in the declared trajectory records.

What is mapped and where it appears in the artifact. In the declared traj_pred.jsonl schema, each predicted event record stores a fine-grained type string under etype_raw, a derived list of compatible skeleton stages under skeleton_hits, and a single serialization label under etype_primary. The skeleton_hits field is what the downstream DP aligner consumes for the type-compatibility check: a skeleton step of stage $\texttt{etype}_k$ can match an event if $\texttt{etype}_k \in \texttt{skeleton_hits}(e)$. This design supports partial observability by allowing some fine-grained event types to be compatible with multiple stages (e.g., a cognition/communication event can support both PREP and PROBE).

Mapping artifact (override file) and fallback behavior. The declared preprocessing procedure supports an optional override mapping file, mapping/event2skeleton.json, which maps a fine-grained type string to a possibly multi-label list of stage labels. This file is included to make domain-specific remapping auditable, but it is not activated in the processed MAVEN-ERE and RAMS runs reported here. For the reported experiments, all event types are mapped by the deterministic fallback mapper in Listing A.8.

Thus, the override-activation rate is 0% and the deterministic fallback mapper handles all records in both instantiations (Table 10). This is intentional: the reported benchmark uses a single documented fallback mapper with fixed constants and one explicit corpus flag for the RAMS PROBE-primary subset, rather than a learned or development-tuned mapping. The optional override file is reserved for domain-specific deployments and is not activated in the reported MAVEN-ERE or RAMS runs.

Listing 1: Optional stage override mapping file format (illustrative JSON; not activated in the reported runs).

```
{
```

```
  "Know": ["PREP","PROBE"],
  "Warning": ["PREP","PROBE"],
  "Statement": ["PREP","PROBE"],
  "Process_start": ["PREP","PROBE"],

  "Attack": ["EXECUTE"],
  "Hostile_encounter": ["EXECUTE"],

  "Process_end": ["OUTCOME"],
  "Earnings_and_losses": ["OUTCOME"]
}
```

Primary stage vs. auxiliary `PROBE` tag (observed instantiation counts). Our mapper assigns each event a *primary* stage in {PREP, PROBE, EXECUTE, OUTCOME} by priority rules, and may additionally attach PROBE as an auxiliary compatibility tag for information-gathering, communication, or cognition events. In MAVEN-ERE, PROBE appears only as an auxiliary tag in the source-normalized records; in RAMS, a small subset is PROBE-primary. This matches the observed accounting in the preprocessing logs:

- **MAVEN-ERE.** There are 91,222 source-normalized event records. The mapper assigns a primary stage to all records (82,688 as PREP, 0 as PROBE, 6,955 as EXECUTE, and 1,579 as OUTCOME; sums to 91,222). In addition, 6,071 events receive the auxiliary PROBE tag, yielding 97,293 total (event,stage) instantiations.
- **RAMS.** There are 9,124 source-normalized event records. The primary-stage distribution is 6,372 PREP, 1,562 EXECUTE, 1,135 OUTCOME, and the remaining 55 events are PROBE-primary. Additionally, 1,259 events receive PROBE as an auxiliary tag, yielding 10,383 total instantiations.

Deterministic fallback mapping procedure. Listing A.8 gives the deterministic fallback procedure used in our preprocessing. The rule uses only: (i) the fine-grained type string, (ii) the trigger surface form (when available from the extractor), and (iii) a dataset flag (to allow a small PROBE-primary subset in RAMS). All sets below are fixed constants in the preprocessing script; no learning is involved.

Listing 2: Deterministic fallback stage mapping procedure (pseudocode).

```
# Inputs:
#   dataset: string in {"MAVEN-ERE","RAMS"}
#   etype_raw: string (fine-grained type, e.g., "Attack", "Know", "
    Process_end")
#   trig_text: string (trigger surface form, may be None if unavailable)
#
# Output:
#   skeleton_hits: list[string] in {"PREP","PROBE","EXECUTE","OUTCOME"} (
    deduplicated)

PROBE_TAG_TYPES = {
  "Know", "Warning", "Statement", "Process_start",
  "Reporting", "Suspicion", "Perception_active", "Coming_to_believe"
}
PROBE_TRIGGERS = {
  "reported", "report", "said", "warned", "warning", "according"
}

EXECUTE_TYPES_STRONG = {"Attack", "Hostile_encounter"}
# Additional execute cues are trigger-driven to keep EXECUTE high-
    precision.
EXECUTE_TRIGGERS = {"killed", "attack", "fought", "destroyed", "battle",
    "war", "hurricane", "storm"}

OUTCOME_TYPES = {"Process_end", "Earnings_and_losses", "Transaction", "
    Receiving", "Releasing"}
```

```
OUTCOME_TRIGGERS = {"ended", "paid", "sold", "bought", "earned", "
    withdrew", "deposited"}

# A small PROBE-primary subset is allowed in RAMS (empirically 55/9124
    events).
PROBE_PRIMARY_TYPES_RAMS = {"Statement", "Reporting"}

def norm(s):
    # deterministic string normalization; no learned or dev-tuned mapping
     is used
    return s.strip()

def lower_or_empty(s):
    return "" if s is None else s.lower().strip()

def map_to_skeleton_hits(dataset, etype_raw, trig_text=None):
    et = norm(etype_raw)
    tr = lower_or_empty(trig_text)

    # 0) Optional override mapping (not triggered in our processed
    datasets; fallback=100% in Table)
    # if et in OVERRIDE_MAP:
    #     return sorted(unique(OVERRIDE_MAP[et]))

    # 1) Decide primary stage by priority: OUTCOME > EXECUTE > (PROBE-
    primary for RAMS) > PREP
    if (et in OUTCOME_TYPES) or (tr in OUTCOME_TRIGGERS):
        primary = "OUTCOME"
    elif (et in EXECUTE_TYPES_STRONG) or (tr in EXECUTE_TRIGGERS):
        primary = "EXECUTE"
    elif (dataset == "RAMS") and (et in PROBE_PRIMARY_TYPES_RAMS):
        primary = "PROBE"
    else:
        primary = "PREP"

    # 2) Auxiliary PROBE tag for information-gathering events.
    # We attach PROBE only when the primary stage is PREP, to keep the
    interface simple and stable.
    add_probe = ((et in PROBE_TAG_TYPES) or (tr in PROBE_TRIGGERS)) and (
    primary == "PREP")

    hits = [primary]
    if add_probe:
        hits.append("PROBE")

    # 3) Return a deterministic order for serialization
    order = {"PREP":0, "PROBE":1, "EXECUTE":2, "OUTCOME":3}
    hits = sorted(list(set(hits)), key=lambda z: order[z])
    return hits
```

Coverage statistics. Table 10 reports the number of source-normalized (event,stage) instantiations produced by the mapper and their stage percentages. This table reports schema-mapping coverage. The same deterministic mapper is also applied to frozen-extractor predicted event records before trajectory construction; all predicted records used in `traj_pred.jsonl` receive at least one `skeleton_hits` label. We report source-normalized coverage here because it is stable across extractor runs and determines the reference-label/audit view, while the model-input predicted-record counts are reported separately in Table 6. In our processed datasets, the override file is never triggered, hence all source-normalized and predicted records use the deterministic fallback mapper.

Auditing note. Stage mapping is purely deterministic and applied before any learning stage. Since `skeleton_hits` are stored alongside `doc_id:span` references in the declared trajectories, the entire downstream alignment and certificate evaluation remains auditable and reproducible under the strict evidence interface.

Table 10: Stage-mapping coverage after preprocessing, reported over source-normalized *(event,stage)* instantiations rather than unique event records. Override-activation rate is 0% for both instantiations; the deterministic fallback mapper handles all records.

	MAVEN-ERE		RAMS	
Stage	# inst.	% inst.	# inst.	% inst.
PREP	82688	84.99	6372	61.37
PROBE	6071	6.24	1314	12.66
EXECUTE	6955	7.15	1562	15.04
OUTCOME	1579	1.62	1135	10.93
Total	97293	100.0	10383	100.0

A.9 Deterministic role normalization

We normalize dataset-native argument roles to the role inventory {Agent, Target, Context} used by skeleton alignment and argument-consistency features. The mapping is deterministic and applied before reward learning.

Agent-like roles are dataset-native roles that identify the main acting or initiating entity of an event, such as attacker, speaker, mover, buyer, seller, or sender when these roles are present in the source schema. Target-like roles identify the main affected or acted-upon entity, such as victim, attacked entity, destination, object, recipient, or affected party when applicable. Temporal, locative, instrumental, descriptive, and residual roles are mapped to Context. When a dataset-native role is ambiguous, the declared mapping/role2norm.json file resolves it deterministically; uncertain residual roles are mapped to Context rather than to Agent or Target.

The declared artifact includes mapping/role2norm.json; the validator and reward-feature extractor use the same mapping file.

Table 11: Role normalization coverage after preprocessing.

	MAVEN-ERE		RAMS	
Normalized role	Count	%	Count	%
Agent	168,431	53.21	12,960	51.43
Target	137,720	43.51	8,100	32.14
Context	10,398	3.28	4,140	16.43

A.10 Skeleton records and construction protocol

Each intent_id is associated with one predefined soft-precedence skeleton in the main benchmark setting. For serialization and deterministic joins, each skeleton record is stored in skeleton.jsonl with both a unique skeleton_id and its associated intent_id. A skeleton record contains: (i) skeleton_id, (ii) intent_id, (iii) an ordered list of skeleton steps, (iv) each step's stage label in {PREP, PROBE, EXECUTE, OUTCOME}, (v) required normalized roles from {Agent, Target, Context}, and (vi) optional soft-precedence edges. In the declared main protocol, the mapping from intent_id to skeleton_id is one-to-one unless explicitly stated otherwise.

The predefined skeleton is not copied from any single candidate trajectory. It is specified at the stage level to represent an abstract intent pattern, and candidate trajectories are treated as noisy partial realizations. The construction protocol is deterministic: (i) we first fix the stage inventory {PREP, PROBE, EXECUTE, OUTCOME}; (ii) we define the canonical stage order and soft-precedence edges at the intent level; (iii) we assign required roles to each step from the normalized role inventory {Agent, Target, Context}; and (iv) we store the resulting skeleton in skeleton.jsonl. Skeleton records are fixed before reward learning, policy training, development tuning, and test evaluation. No dev/test labels or dev/test model outputs are used to define or modify skeleton records. The same skeleton file is used by trajectory alignment, reward-feature extraction, certificate serialization, and the feasibility / faithfulness validators.

Listing 3: Skeleton record format (schematic JSONL).

```
{
  "skeleton_id": "skel_001",
```

```
  "intent_id": "intent_001",
  "steps": [
    {"step_id": "s1", "etype": "PREP", "required_roles": ["Agent"]},
    {"step_id": "s2", "etype": "PROBE", "required_roles": ["Agent","
    Target"]},
    {"step_id": "s3", "etype": "EXECUTE", "required_roles": ["Agent","
    Target"]},
    {"step_id": "s4", "etype": "OUTCOME", "required_roles": ["Agent"]}
  ],
  "precedence": [["s1","s2"], ["s2","s3"], ["s3","s4"]]
}
```

In general-domain event corpora, `OUTCOME` should be read as an abstract terminal or outcome stage.

For the main experiments, all windows with the same `intent_id` use the same skeleton record. Appendix M.6 evaluates controlled perturbations of this design.

A.11 Reward-learning corpus construction details

ECPO learns a trajectory-level plan-consistency reward from indirect supervision. We construct a reward-learning corpus containing *expert positives*, *pseudo-experts*, *same-window hard (near-neighbor) negatives*, *train-only perturbation-derived negatives*, and *preference pairs*. Here "reward-learning corpus" refers to the training-side trajectory pool that participates in reward-model comparison sets or within-window partition terms; it therefore includes unlabeled candidate-pool trajectories when they serve as same-window alternatives, not only explicitly labeled positives or negatives.

Corpus scale (MAVEN-ERE). For the processed MAVEN-ERE instantiation, the retained manuscript reports 7,245 trajectory records in total: 3,774 comparison-pool candidate trajectories carried without explicit expert/pseudo-expert labels, 755 expert/pseudo-expert demonstrations, and 2,716 hard/perturbed negatives. In addition, the declared construction contains 3,622 within-window preference pairs covering 7,244 unique trajectory IDs (Table 14).

Corpus scale (RAMS). In the processed RAMS instantiation, the reward-learning trajectory pool contains 8,864 trajectory records: 8,730 window-scoped candidate trajectories from ranking windows and 134 train-only hard/perturbation negatives, with an average trajectory length of 1.07 events over the candidate trajectories. The preference supervision contains 134 within-window pairs covering 268 unique trajectory IDs, and we additionally provide 654 RL prompt records for window-level policy optimization (Table 15).

Expert positives and evaluation labels. We separate supervision labels from ranking inputs. Expert/pseudo-expert labels and evaluation positives are derived from dataset-native evidence and validated skeleton-aligned chains. These annotations are used to label or validate candidate trajectories, but the trajectory records consumed by the reward model and ranking policies are produced by the fixed upstream extractor. Thus, training annotations provide supervision for reward learning and policy training on the train split, while held-out annotations define positive labels and audit references on dev/test splits. All ranking methods operate on the same fixed-extractor trajectory inputs in their respective splits.

For training-split expert/pseudo-expert records, each validated step stores (i) an intent-stage label (`PREP/PROBE/EXECUTE/OUTCOME`); (ii) normalized roles (`Agent/Target/Context`); (iii) linked entity IDs; (iv) timestamps or order indices; and (v) auditable evidence spans (`doc_id` with character offsets). For dev/test ranking inputs, the corresponding fields are generated by the fixed extractor and are treated as noisy observations. *Operational note:* in a domain deployment, the same interface could additionally consume analyst-curated POI records from an investigation KB, but this is not required for (and not used in) our declared benchmark.

Pseudo-experts from event corpora. To increase demonstration diversity when expert candidate-level demonstrations are sparse, we mine pseudo-expert trajectories from upstream event corpora by selecting chains with high skeleton-alignment scores, yielding skeleton-consistent yet non-identical demonstrations.

Hard, near-neighbor, and perturbation-derived negatives. Same-window hard or near-neighbor negatives are non-positive candidates in the same ranking window that share superficial similarity with positives, such as overlapping documents, similar local event types, nearby temporal/order positions, or similar role/entity patterns, but lack a validated skeleton-aligned source chain. Perturbation-derived negatives are synthetic training-only variants of expert or pseudo-expert trajectories, created by replacement, insertion, deletion, time shift / shuffle, or entity-role swaps. They are used for reward-learning comparison sets and train-split policy augmentation, and are not inserted into dev/test ranking pools as ordinary candidates.

Preference pairs. We create within-window preference supervision $(\tau^{\text{better}}, \tau^{\text{worse}})$ according to skeleton-alignment quality. When sufficient evidence is available, τ^{better} contains later-stage evidence such as `EXECUTE` and/or `OUTCOME` with fewer precedence violations. In sparse settings such as RAMS, preferences are based on higher validated alignment score, role satisfaction, and evidence traceability, rather than requiring all late-stage steps to be observed.

A.12 Minimal example window (anonymized)

Below is a simplified illustrative excerpt (IDs and text are anonymized):

Listing 4: Minimal anonymized excerpt of a `traj_pred.jsonl` trajectory record.

```
traj_pred.jsonl (excerpt for candidate P_001):
{"window_id":"w_0001","candidate_id":"P_001","trajectory_id":"w_0001::
    P_001","events":[
 {"event_id":"e1",
 "etype_raw":"Process_start",
 "skeleton_hits":["PREP","PROBE"],
 "etype_primary":"PREP",
 "time":"2021-08-03",
 "order_index":0,
 "trigger":{"doc_id":"doc123","span":[120,156]},
   "arguments":[{"role":"Agent","entity_id":"P_001","doc_id":"doc123","
    span":[90,110]}]},
    {"event_id":"e2",
   "etype_raw":"Statement",
   "skeleton_hits":["PREP","PROBE"],
   "etype_primary":"PREP",
   "time":"2021-08-05",
   "order_index":1,
   "trigger":{"doc_id":"doc127","span":[33,61]},
   "arguments":[{"role":"Agent","entity_id":"P_001","doc_id":"doc127","
    span":[10,18]},
                {"role":"Target","entity_id":"ORG_007","doc_id":"doc127
    ","span":[80,95]}]}
]}
```

Schema files. the declared construction contains JSON Schemas under `schemas/` to validate each record type. The evaluation scripts use the same schemas for feasibility checking and audit metrics.

Listing 5: SFT and RL-prompt formatting and column mapping (artifact metadata).

```
maven_sft.jsonl and rams_sft.jsonl: Alpaca formatting
  prompt  -> instruction
  query   -> input
  response-> output
  system  -> system
  history -> history

rl_demos.jsonl and rams_rl_demos.jsonl: ECPO formatting (prompt+response)
  prompt  -> prompt
  response-> response
```

Table 12: Top-20 source-normalized event types in MAVEN-ERE annotations after preprocessing (by frequency; total source-normalized event records = 91,222).

Rank	Type	Count	%	Rank	Type	Count	%
1	Causation	3297	3.61	11	Statement	1677	1.84
2	Process_start	3191	3.50	12	Conquering	1661	1.82
3	Attack	3059	3.35	13	Self_motion	1565	1.72
4	Hostile_encounter	3057	3.35	14	Arriving	1521	1.67
5	Catastrophe	2749	3.01	15	Destroying	1397	1.53
6	Competition	2625	2.88	16	Coming_to_be	1387	1.52
7	Motion	2533	2.78	17	Bodily_harm	1235	1.35
8	Killing	1839	2.02	18	Death	1176	1.29
9	Process_end	1705	1.87	19	Damaging	1157	1.27
10	Social_event	1689	1.85	20	Creating	1145	1.26
Top-20 coverage: 39,665 source-normalized event records (43.48%).							

Table 13: Top-20 trigger surface forms in processed MAVEN-ERE source-normalized records (by frequency; total source-normalized event records = 91,222).

Rank	Trigger text	Count	%	Rank	Trigger text	Count	%
1	killed	948	1.04	11	hurricane	484	0.53
2	began	886	0.97	12	fought	452	0.50
3	held	757	0.83	13	destroyed	402	0.44
4	took place	732	0.80	14	battle	397	0.44
5	caused	691	0.76	15	tour	397	0.44
6	occurred	649	0.71	16	made	382	0.42
7	became	617	0.68	17	reported	378	0.41
8	damage	586	0.64	18	ended	370	0.41
9	storm	578	0.63	19	war	361	0.40
10	attack	515	0.56	20	Battle	353	0.39
Top-20 coverage: 10,935 source-normalized event records (11.99%).							

```
rl_prompts.jsonl and rams_rl_prompts.jsonl: ECPO formatting (prompt only)
  prompt  -> prompt
```

A.13 Event type and trigger distributions (MAVEN-ERE)

To make the long-tail property explicit, we list the most frequent source-normalized event types and trigger surface forms in the processed MAVEN-ERE instantiation (total source-normalized event records: 91,222). Table 12 and Table 13 report the Top-20 distributions.

A.14 Dataset statistics (MAVEN-ERE)

Summary. After preprocessing, the MAVEN-ERE reference view contains 3,623 documents and 91,222 source-normalized event records. The fixed-extractor model-input / reward-learning view contains 7,245 trajectory records over 3,623 unique source-level candidate/entity IDs before window expansion. These records are consumed by the ranking / reward-learning code paths and may include train-only hard/perturbation negatives in addition to window-scoped ranking trajectories. Trajectory lengths are measured over trajectory-expanded predicted event mentions and range from 1 to 128 events (median 21; mean 24.7). Although the construction permits empty trajectories, none occur in the reported MAVEN-ERE construction; empty trajectories would still be retained and scored by missing-step features if they were present. For reward learning, we tag 755 trajectories as expert/pseudo-expert demonstrations and 2,716 as hard/perturbed negatives, and we create 3,622 within-window preference pairs (covering 7,244 unique trajectory IDs).

A.15 Dataset statistics (RAMS)

Summary. The RAMS reference view contains 9,124 documents and 9,124 source-normalized event records, one per document in `rams_event.jsonl`. The fixed-extractor model-input / reward-learning view contributes 9,386 unique predicted event records over 8,730 unique source-level candidate/entity IDs before window expansion. It contains 8,730 ranking-window

Table 14: Statistics of the processed MAVEN-ERE instantiation. Rows marked as source-normalized records describe normalized dataset annotations; rows marked as model-input records describe fixed-extractor predicted trajectories consumed by ranking methods. *Note:* skeleton-step hit counts are computed over window-scoped trajectory instances; overlapping windows may duplicate evidence. The reported MAVEN-ERE construction contains no empty serialized trajectories, so the minimum trajectory length is 1 even though the construction allows empty trajectories in general.

Statistic	Value
Documents	3,623
Source-normalized event records	91,222
Unique source-level candidate/entity IDs before window expansion	3,623
Total trajectory records exposed to ranking / reward learning	7,245
Trajectory record subsets	unlabeled candidate pool 3,774; expert/pseudo demos 755; hard/perturbed negatives 2,716
Preference pairs	3,622
Trajectory length (#trajectory-expanded predicted event mentions)	min 1; median 21; mean 24.7; max 128
Preference coverage	7,244 trajectory IDs
Role mentions in source-normalized records	Agent 168,431; Target 137,720; Context 10,398
Skeleton-step hits after window / trajectory expansion	PREP 143,859; PROBE 11,876; EXECUTE 31,340; OUTCOME 3,630
Top-20 event types	39,665 (43.5%)
Top-20 triggers	10,935 (12.0%)

candidate trajectories and 134 additional train-only hard/perturbation negatives, for 8,864 total trajectory records. Trajectory sparsity is measured over the ranking-window candidate trajectories and averages 1.07 trajectory-expanded predicted event mentions. For preference-based supervision, the declared construction contains 134 within-window preference pairs (`rams_pairs.jsonl`) covering 268 unique trajectory IDs. For supervised training, the declared construction contains 9,124 Alpaca-format SFT records with train/dev/test splits of 7,329/924/871 (`rams_sft_train/dev/test.jsonl`). For RL training, the declared construction contains 654 window-level RL prompt records (`rams_rl_prompts.jsonl`). Key statistics are summarized in Appendix Table 15.

Table 15: Statistics of the processed RAMS instantiation.

Statistic	Value
Documents	9,124
Source-normalized event records (`rams_event.jsonl`)	9,124
Unique predicted event records contributing to model inputs	9,386
Unique source-level candidate/entity IDs before window expansion	8,730
Ranking-window candidate trajectories (`rams_traj.jsonl`)	8,730
Additional train-only hard/perturbation negatives	134
Total trajectory records exposed to ranking / reward learning	8,864
Average trajectory length (#trajectory-expanded predicted event mentions)	1.07
Preference pairs (`rams_pairs.jsonl`)	134
Preference coverage (unique trajectory IDs)	268
SFT records (`rams_sft.jsonl`)	9,124
SFT splits (`train/dev/test`)	7,329 / 924 / 871
RL prompt records (`rams_rl_prompts.jsonl`)	654

B Frozen Extractor and Trajectory Construction

The event extractor is an upstream, frozen component rather than part of ECPO's policy optimization. We retain the staged supervised curriculum recorded in the source manuscript,

$$\texttt{MAVEN} \rightarrow \texttt{MAVEN-Arg} \rightarrow \texttt{MAVEN-ERE} \rightarrow \texttt{RAMS},$$

where `MAVEN-Arg` is an intermediate extractor-training stage. For each target corpus, documents are assigned to train, development, and test splits before the curriculum reaches that corpus. Only target-corpus training documents and labels may update the extractor; development and test documents are processed only after the extractor is frozen.

Listing 6: Leakage-controlled data flow for one target corpus.

```
1. Split documents into train/dev/test by document ID.
2. Train the extractor only on permitted training documents.
3. Freeze it and predict event records for every split.
4. Build window-scoped candidate trajectories from predictions.
5. Build training positives, negatives, and preferences only from
   training annotations.
6. Use development annotations only for model selection and audit.
7. Use test annotations only for final labels and audit references.
8. Train the reward model and policy only with training supervision.
9. Evaluate every ranker on the same predicted dev/test inputs.
```

The four record families are deliberately isolated. `window_input.jsonl` fixes the evidence scope, intent, skeleton, and roster; `traj_pred.jsonl` contains frozen-extractor events and provenance and is the only event source visible to ranking methods; `window_label.jsonl` stores positive candidate IDs; and `audit_ref.jsonl` stores source-normalized reference spans for evaluation. Candidate eligibility may be projected from source annotations before labels are attached, but event stages, reference chains, labels, and reward scores are discarded from that projection. A candidate with no predicted event receives an empty trajectory rather than a source-derived substitute.

Each predicted trigger and argument retains a document identifier and half-open character span. Events are aggregated by window-local candidate ID and ordered by reliable timestamp when present, otherwise by the deterministic document-local order key. These fields are required to reproduce extractor error propagation but are not inferred from the older draft because its execution artifacts were not included in the local bundle.

C Minimal Interface and Preconditions

The policy receives `window_id`, `intent_id`, the resolved skeleton, randomized candidate IDs, and candidate-linked provenance. A minimal output for $k = 1$ is:

```
{"window_id":"w_0001","topk":["cand_02"],
 "certificates":[{"steps":[
  {"step_id":"s1","etype":"PREP","matched":true,
   "event_id":"ev_7",
   "evidence":[{"doc_id":"d1","span":[12,25]}]},
  {"step_id":"s2","etype":"PROBE","matched":false,
   "event_id":null,"evidence":[]}]}]}
```

The full validator additionally requires every skeleton step, evidence kind/role fields for citations, exact Top-k length, candidate membership, and position-aligned certificates. The second step shows the unique unmatched representation. Event IDs may be used by feasibility validation but are removed before all evidence-only reconstruction.

D Exact Aligner and Verifier Specification

Dynamic program. For skeleton length M and trajectory length T,

$$\mathrm{DP}[0,0] = 0, \qquad \mathrm{DP}[0,t] = -t\delta_{\mathrm{skip}}, \quad \mathrm{DP}[i,0] = -i\delta_{\mathrm{miss}}, \tag{18}$$

$$\mathrm{DP}[i,t] = \max \begin{cases} \mathrm{DP}[i-1,t-1] + m(i,t), \\ \mathrm{DP}[i-1,t] - \delta_{\mathrm{miss}}, \\ \mathrm{DP}[i,t-1] - \delta_{\mathrm{skip}}. \end{cases} \tag{19}$$

The fixed values are $\lambda_{\mathrm{arg}} = 1.00$, $\delta_{\mathrm{type}} = 0.75$, $\delta_{\mathrm{miss}} = 0.60$, and $\delta_{\mathrm{skip}} = 0.20$. Ties prefer a compatible match, then skip over miss, then the smaller temporal gap. A diagonal type-mismatch is serialized as missing, never as supporting evidence. For a compatible match, the certificate contains the trigger plus at most one required-role argument per role; the shortest span is selected, with smallest start offset as tie-break.

Deterministic stage mapper. The reported runs use no override mapping (activation 0%). The fallback mapper strips whitespace from the case-preserved fine-grained type and lowercases/strips the trigger, then assigns a primary stage with priority `OUTCOME`>`EXECUTE`>RAMS-only `PROBE`>`PREP`. `OUTCOME` type cues are `Process_end`, `Earnings_and_losses`, `Transaction`, `Receiving`, and `Releasing`; its trigger cues are `ended`, `paid`, `sold`, `bought`, `earned`, `withdrew`, and `deposited`. `EXECUTE` type cues are `Attack` and `Hostile_encounter`; trigger cues are `killed`, `attack`, `fought`, `destroyed`, `battle`, `war`, `hurricane`, and `storm`. In RAMS only, `Statement` and `Reporting` are `PROBE`-primary if neither higher-priority rule fires. Otherwise the primary stage is `PREP`.

An auxiliary `PROBE` hit is appended only when the primary stage is `PREP` and the type is one of `Know`, `Warning`, `Statement`, `Process_start`, `Reporting`, `Suspicion`, `Perception_active`, or `Coming_to_believe`, or the trigger is `reported`, `report`, `said`, `warned`, `warning`, or `according`. Hits are deduplicated and serialized in `PREP`, `PROBE`, `EXECUTE`, `OUTCOME` order.

Local terms and reward-model learning. The compatible-stage set $\mathrm{skeleton_hits}(e_t)$ is produced by the fixed type/trigger mapper. For $s_i = (\mathrm{stage}_i, \mathcal{R}_i)$ and event arguments $\mathcal{A}_t$, $\mathrm{role_sat}$ and $\mathrm{arg_viol}$ are exactly the quantities defined in Section 4. The latter counts candidate-identity violations for key roles and role-specific incompatibilities present in the event record; main runs add no role entity-type constraint. Soft precedence violations and temporal gaps are computed only after backtracking.

The recorded feature vector concatenates ϕ_{skel} (hit/miss counts, compatible-hit coverage, skip count, mean gap between consecutive matches, and soft precedence diagnostics), ϕ_{arg} (role satisfaction and available role/candidate incompatibilities), and ϕ_{graph}. Coverage is

$$M^{-1}|\{i : \exists t,\ \pi_{\text{align}}(i) = t\ \wedge\ \text{stage}(s_i) \in \text{skeleton_hits}(e_t)\}|, \tag{20}$$

with zero at $M = 0$; skips are events not aligned to a step; the mean gap uses timestamps or normalized event indices; and an evaluable precedence edge is violated by contradicted or explicitly uncertain order. Main reward-model runs use $\phi_{\text{skel}} + \phi_{\text{arg}} + \phi_{\text{graph}}$ and ℓ_2 coefficient $\lambda = 10^{-3}$. The graph encoder is trained self-supervised on train-split fixed-extractor temporal graphs and frozen before reward learning; dev/test graphs are forward-only. The source record does not specify the remaining count/gap feature scaling or the full graph-encoder configuration, so these are replication gaps rather than silently chosen conventions.

For reward learning, let $\mathcal{B}_w = \{\widetilde{\tau}_{w,1}, \ldots, \widetilde{\tau}_{w,B_w}\}$ be the train-window comparison set and $\mathcal{P}_w^+ \subseteq \mathcal{B}_w$ its expert or pseudo-expert positives. We use

$$P_\theta(\tau \mid w) = \frac{\exp R_\theta(\tau; S_w)}{\sum_{j=1}^{B_w} \exp R_\theta(\widetilde{\tau}_{w,j}; S_w)}, \tag{21}$$

$$\mathcal{L}_{\text{me}}(\theta) = \sum_{w:|\mathcal{P}_w^+|>0} \frac{1}{|\mathcal{P}_w^+|} \sum_{\tau^+ \in \mathcal{P}_w^+} \log P_\theta(\tau^+ \mid w), \tag{22}$$

$$\mathcal{L}_{\text{pair}}(\theta) = \sum_{(w,\tau^+,\tau^-)\in\mathcal{Q}} \log \sigma\big(R_\theta(\tau^+; S_w) - R_\theta(\tau^-; S_w)\big)\,, \tag{23}$$

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{me}} + \alpha_{\text{pair}}\mathcal{L}_{\text{pair}} - \frac{\lambda}{2}\|\theta\|_2^2. \tag{24}$$

The comparison sets and preference pairs $\mathcal{Q}$ use train-split positives, same-window alternatives, and perturbation negatives only. The main record fixes $\lambda = 10^{-3}$ but does not report α_{pair}.

Verifier-A edge cases. Equation 11 is evaluated only after answer-bearing fields are stripped. Trigger matches take precedence over arguments. Strict IoU equality at 0.50 fails, whereas support and margin equality at 0.55 and 0.10 pass. RoleSat equals one for no required roles; Prec equals zero for no evaluable supported edge; Trace and BadSpan equal zero for no citations. A defective citation contributes once to BadSpan even if it violates multiple rules. Slot-wise assignment uses deterministic bundle/candidate order only after support and margin tests; rank-blind assignment abstains on exact ties and is invariant to bundle permutation.

Held-out evaluators. Verifier-B uses citation coverage $|\text{cited} \cap \text{event}|/|\text{cited}|$ instead of IoU, retains StepCov/Trace/BadSpan, removes RoleSat/Prec, has no learned component, and calibrates only from disjoint training-split synthetic corruptions. It uses no dev/test labels, policy outputs, rewards, or audit references. Verifier-C consumes cited raw text with fixed context, re-extracts entity/stage/role, maps the entity through window-local aliases/coreference, performs one-to-one support assignment with abstention, and orders by evidence score. Its training/calibration uses train-split text/span annotations. It is distinct from REEXTRACTFAITHFULNESS; the evaluators recover candidate identity and order. Verifier-B and Verifier-C use the retained evaluator configurations above.

Training and evaluation flow (pseudocode). For each train window: (1) align every candidate trajectory to its skeleton and compute R_θ; (2) sample a group of joint ranking/certificate outputs; (3) parse and validate each output; (4) strip answer-bearing fields and run Verifier-A over all candidate–bundle pairs; (5) solve one-to-one assignment, compute r_{rank}, r_{cert}, and r_{cycle}, and apply invalid/short-output and KL terms; and (6) update the policy with group-normalized clipped ratios. For each test window, decode once, retain the output without repair, compute ordinary ranking from a valid full-length candidate list, and compute verifier/audit metrics from the submitted evidence. Held-out B/C, rank-blind permutation, counterfactual replacement, and human annotation run only after the primary output is fixed.

Reward callback and GRPO update. For a sampled sequence y under input x_w, the token-level estimate is

$$\widehat{\mathrm{KL}}_{\mathrm{tok}}(y, x_w) = \frac{1}{|y|} \sum_{t=1}^{|y|} \left[\log \pi_\varphi(y_t \mid y_{<t}, x_w) - \log \pi_{\mathrm{ref}}(y_t \mid y_{<t}, x_w)\right]. \tag{25}$$

For G outputs sampled from the same window, with the complete scalar reward r_{ECPO} from Section 4,

$$A^{(g)} = \frac{r_{\mathrm{ECPO}}(y^{(g)}, w) - \mu_w}{\widehat{\sigma}_w + \epsilon_{\mathrm{adv}}}, \qquad \mu_w = G^{-1} \sum_{g=1}^{G} r_{\mathrm{ECPO}}(y^{(g)}, w). \tag{26}$$

The reward has stop-gradient through sampled log probabilities. With $\rho^{(g)} = \pi_\varphi(y^{(g)} \mid x_w)/\pi_{\mathrm{old}}(y^{(g)} \mid x_w)$, the update maximizes

$$\mathcal{J}(\varphi) = \mathbb{E}_{w,g}\Big[\min\Big(\rho^{(g)} A^{(g)}, \mathrm{clip}(\rho^{(g)}, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}) A^{(g)}\Big)\Big]. \tag{27}$$

The reference policy stays frozen. The coefficient controller is $\beta_{\mathrm{KL}} \leftarrow \beta_{\mathrm{KL}} \exp\{\kappa(\widehat{\mathrm{KL}}_{\mathrm{tok}} - \mathrm{KL}_{\mathrm{target}})\}$ with target 0.05 nats/token; the initial coefficient is 0.05.

Invalid and short outputs. During training, an unparseable output or one with no valid candidate prefix receives zero ranking, certificate, and cycle utility plus the invalid and missing-prefix penalties. A parseable prefix shorter than k receives utility only for its parsed positions and the explicit $\xi_{\mathrm{miss}}(k - |A|)$ penalty. During evaluation, strict Top-k length is part of feasibility: an unparseable, out-of-roster, duplicate, or short/long candidate list receives zero ordinary ranking and zero evidence-recovery credit for that window and is counted infeasible. If the full candidate list is valid but a certificate is invalid, ordinary ranking is retained, its verifier mask is zero, and its slot/rank-blind recovery receives no credit.

E Rank-Blind Metric Edge Cases

The rank-blind evaluator removes direct claims, event IDs, ranks, policy scores, and candidate–bundle pairing, then permutes bundles. Threshold/margin failures and exact assignment ties abstain. SetRecover uses a denominator of submitted k and gives no credit to abstention. RankRecover bottom-pads abstentions; missing, incorrect, abstained, or exact evidence-score-tied candidates receive zero position credit. Pairwise accuracy excludes unrecovered pairs from its conditional denominator but assigns zero to a window with no recovered pair; SetRecover reports coverage. Kendall's τ_b keeps evidence-score ties after restricting both orders to recovered submitted candidates; fewer than two candidates or zero denominator yields zero. Empty submitted targets yield zero for all rank-blind metrics.

F Counterfactual Evidence Controls

Each condition is evaluated on the same eligible windows. The matched-unselected control replaces one target bundle at a time with a validator-valid bundle for a same-window unselected candidate, matched on cited length and stage profile; the other $k - 1$ bundles remain fixed.

Table 16: Counterfactual controls (MAVEN-ERE/RAMS), evaluated by held-out Verifier-B.

Evidence condition	Target rec.	SetRec.	RankRec.	vCERTNDCG
Original ECPO	**0.68/0.53**	**0.70/0.55**	**0.55/0.42**	**0.55/0.38**
Same-candidate random	0.52/0.40	0.51/0.39	0.25/0.18	0.30/0.21
Stage-matched, non-aligned	0.57/0.44	0.55/0.42	0.31/0.22	0.35/0.24
Wrong stage	0.44/0.33	0.42/0.31	0.20/0.14	0.24/0.16
Matched unselected candidate	0.14/0.11	0.67/0.52	0.18/0.12	0.15/0.10
Key-step deletion	0.54/0.41	0.52/0.39	0.30/0.21	0.34/0.23

G SENSITIVITY AND UPSTREAM CEILINGS

The 47 Verifier-A configurations comprise 20 one-weight perturbations (five weights multiplied by $0.5, 0.8, 1.2, 1.5$ and renormalized) and a 27-setting grid with overlap threshold in $\{0.40, 0.50, 0.60\}$, support threshold in $\{0.45, 0.55, 0.65\}$, and margin in $\{0.05, 0.10, 0.15\}$. They rescore fixed outputs without policy retraining. The audit reports aggregate ranges and ordering counts over the 47 configurations.

Table 17: Verifier sensitivity and fixed-roster extractor diagnostics. Scalar oracle, higher-recall, and deletion rows are vCERTNDCG@10; the percentage and ordering-count rows are labeled separately.

Diagnostic	MAVEN-ERE	RAMS
vCERTNDCG range, 47 settings	0.55–0.60	0.38–0.42
Worst gain over RM-only+Det	+0.05	+0.02
Settings preserving ordering	45/47	43/47
Predicted-evidence oracle	0.72	0.54
Source-normalized oracle	0.86	0.76
ECPO / predicted oracle	80.6%	74.1%
Higher-recall extractor	0.64	0.48
20% predicted-event deletion	0.50	0.33

The predicted/source oracle rows are verifier-relative reference points. The deletion experiment fixes the roster and removes predicted events; it is distinct from end-to-end candidate-generator perturbation.

H HUMAN AUDIT PROTOCOL

We sample 120 stratified test windows and evaluate GRPO, RM-only+Det, and ECPO: 1,080 bundles per method (540 per dataset) and 3,240 base method–bundle units. Annotators receive a no-UI text protocol and do not see method identity, model/evaluator scores, relevance labels, stable IDs, or submitted ranks. They judge (i) step support, (ii) candidate recovery from stripped evidence, (iii) comparative support against a predeclared strongest non-target alternative, and (iv) order recovery. "Ambiguous" is a failure. The source window is the inference and paired-bootstrap cluster. Audit labels affect no verifier threshold, checkpoint, decoding, or selection. A stratified 20% subset (648 units) is independently repeat-annotated; comparative-support Krippendorff's α is 0.71. ECPO ambiguity rates are $0.08/0.11$ versus $0.12/0.16$ for RM-only+Det (MAVEN-ERE/RAMS).

I ADDITIONAL EXPERIMENTAL AND COST DETAILS

The Qwen3-8B policies use QLoRA (Dettmers et al., 2023), two SFT epochs at learning rate 1×10^{-4}, effective batch size 128, maximum input/output lengths 4,096/768, and $4\times$A100 80GB. The validator retry budget is 3 during training and 0 during evaluation; test decoding is greedy with no repair. ECPO results in the primary tables average three seeds. Paired confidence intervals resample source windows. ECPO policy optimization uses a measured 76 GPU-hours per corpus-seed. The shared reward model costs 8 GPU-hours. RM-only+Align uses no LLM decode and averages 0.18 s/window; RM-only+Det adds CPU evidence search. ECPO uses one decode and averages 3.1 s/window. These costs exclude the shared frozen extractor.

Retained replication fields. The retained values are listed in the artifact tables.

Here RM-only+Align ranks by R_θ and attaches one DP-aligned evidence chain; it is distinct from RM-only+Det's evidence search. GRPO+constrained keeps the trained policy fixed, uses a single grammar/JSON-constrained decode, and takes the first valid constrained output. GRPO+best-of-10 RM keeps training fixed, samples ten outputs, and selects the one with highest R_θ list utility.

Table 18: ECPO seed variation across the three primary runs.

Data	Seed	NDCG	vCERTNDCG$_A$	SLOT-EVIDCONS	Feasible
MAVEN-ERE	1	0.69	0.57	0.64	0.94
MAVEN-ERE	2	**0.71**	**0.59**	**0.66**	**0.95**
MAVEN-ERE	3	0.70	0.58	0.65	0.94
RAMS	1	0.60	0.39	0.48	0.92
RAMS	2	**0.62**	**0.41**	**0.50**	**0.93**
RAMS	3	0.61	0.40	0.49	0.92

Table 19: Paired source-window bootstrap differences with 95% confidence intervals.

Data	ECPO minus baseline	ΔNDCG	ΔvCERTNDCG$_A$
M	RM-only+Align	$-0.01[-0.03, +0.02]$	$+0.15[+0.10, +0.19]$
M	GRPO+constrained	$+0.06[+0.02, +0.08]$	$+0.21[+0.12, +0.24]$
M	GRPO+best-of-10 RM	$-0.02[-0.04, +0.01]$	$+0.12[+0.05, +0.15]$
R	RM-only+Align	$-0.01[-0.03, +0.02]$	$+0.07[+0.04, +0.12]$
R	GRPO+constrained	$+0.05[+0.01, +0.07]$	$+0.14[+0.09, +0.17]$
R	GRPO+best-of-10 RM	$-0.02[-0.03, +0.02]$	$+0.06[+0.03, +0.10]$

J METRIC SEMANTICS AND STATISTICAL PROTOCOL

J.1 WHAT EACH METRIC ESTABLISHES

The evaluation deliberately separates syntactic acceptance, source traceability, submitted-slot recovery, rank-blind comparative recovery, and human support. These constructs are related but not interchangeable. A parseable evidence bundle may cite valid spans without distinguishing its candidate from alternatives; conversely, a human may find a bundle sufficient even when a fixed automatic matcher abstains. Table 20 states the strongest interpretation licensed by each reported metric.

Table 20: Metric interpretation used in the experiments.

Metric or audit	Directly measures	Supports interpretation of
NDCG@10	ordinary relevance order	ranking utility
SLOT-EVIDCONS@10	submitted-slot bundle traceability	candidate-linked support at the emitted position
vCERTNDCG@10	relevance-weighted ranking with verifier recovery mask	verifier-audited utility
SETRECOVER@10	recovered candidate set after permutation	rank-blind set recovery
RANKRECOVER@10	recovered candidate order after permutation	rank-blind order recovery
Pairwise accuracy / τ_b	order agreement among recovered candidates	comparative ordering
FEASIBLERATE	all-window validator acceptance	structural output validity
Human comparative support	blinded annotator judgment	practical evidence support

J.2 Ordinary ranking metrics

For a valid returned list $(a_1, \ldots, a_{K_w})$ with binary relevance $\mathrm{rel}_w(a_j)$, define

$$\mathrm{Hit@}K_w = \mathbb{I}\left[\sum_{j=1}^{K_w} \mathrm{rel}_w(a_j) > 0\right], \tag{28}$$

$$\mathrm{AP@}K_w = \frac{1}{\max(1, \min(K_w, |\mathcal{P}_w|))} \sum_{j=1}^{K_w} \mathrm{Prec@}j \ \mathrm{rel}_w(a_j), \tag{29}$$

$$\mathrm{DCG@}K_w = \sum_{j=1}^{K_w} \frac{\mathrm{rel}_w(a_j)}{\log_2(j+1)}, \tag{30}$$

$$\mathrm{NDCG@}K_w = \frac{\mathrm{DCG@}K_w}{\max(\epsilon, \mathrm{IDCG@}K_w)}. \tag{31}$$

A window with no relevant candidate has zero Hit, AP, and NDCG under the declared evaluation convention. An invalid, duplicate, out-of-roster, short, or long candidate list receives zero ordinary ranking credit and is counted infeasible. A valid ranking whose evidence object fails validation retains ordinary ranking credit but receives zero evidence credit for the affected bundle.

For Verifier-A mask $z_{w,j}$, vCertNDCG uses the same discount and ideal denominator:

$$\mathrm{vCertNDCG@}K_w = \frac{\sum_{j=1}^{K_w} z_{w,j} \mathrm{rel}_w(a_j) / \log_2(j+1)}{\max(\epsilon, \mathrm{IDCG@}K_w)}. \tag{32}$$

Because $z_{w,j} \in \{0, 1\}$, the masked DCG cannot exceed the corresponding ordinary DCG for the same ranking. This algebraic fact defines the metric's relationship to NDCG.

J.3 Rank-blind denominators and ties

Rank-blind evaluation removes candidate claims, stable IDs, generated event IDs, submitted ranks, policy scores, bundle order, and candidate–bundle pairing. The evaluator scores every stripped bundle against every candidate, solves a partial one-to-one assignment, and orders recovered candidates by evidence-derived support. Threshold failure, margin failure, or an exact assignment tie causes abstention. Set recovery divides by submitted k, so abstention receives no membership credit. Rank recovery bottom-pads abstentions, and a missing, wrong, abstained, or exact-score-tied position receives zero. Pairwise accuracy assigns 1, 0.5, and 0 to correct, tied, and reversed recovered pairs; a window with no recovered pair receives zero. Kendall's τ_b retains evaluator score ties and is zero when fewer than two candidates remain or its denominator is zero.

J.4 Aggregation and paired uncertainty

The inference unit and paired-resampling cluster is the source ranking window. All methods compared in a paired interval must use the same retained window IDs, candidate rosters, relevance labels, and fixed policy outputs. A bootstrap replicate samples source windows with replacement, retains all method outputs and audit units associated with each sampled window, recomputes the complete metric, and records the paired difference. Intervals are percentile 95% intervals over paired differences. Seed summaries first compute each seed's window mean and then report the mean across the declared seeds; individual-seed rows remain visible in Table 18.

K Held-Out Evaluators and Transfer Protocol

K.1 Verifier-B: deterministic form transfer

Verifier-B is evaluation-only for primary ECPO-A. It replaces trigger/argument IoU matching with citation-coverage matching, retains StepCov, Trace, and BadSpan, and removes RoleSat and Prec. Its parameters are selected from synthetic corruptions constructed from disjoint training-split

Table 21: Statistical protocol used for the reported paired intervals and seed summaries.

Field	Declared behavior	Retained value
Resampling cluster	source ranking window	window ID; `sha256:8f31d4a7`
Confidence level	paired percentile interval	95%
Bootstrap replicate count	identical paired window samples across methods	2000
Random-number generator and seed	fixed before inspecting test differences	PyTorch/NumPy seeds $(42, 43, 44)$
Treatment of missing method output	zero metric credit on the paired window	`missing-v3`
Multiple-comparison correction	descriptive primary intervals	Holm correction not applied
Seed aggregation	three primary ECPO seeds	mean over checkpoints; seeds $(42, 43, 44)$

records. The selection procedure may use neither development/test relevance, generated policy outputs, reward values, checkpoint-selection outcomes, nor human audit labels. After calibration, B is frozen before any primary test output is evaluated.

The B audit tests whether the direction of the ECPO advantage survives a materially different deterministic evidence matcher. B retains the finite roster, broad stage inventory, and candidate-linked provenance scope. The reverse-transfer ECPO-B run is a separate policy trained with B in its reward and evaluated by A and C; it must not be merged with primary ECPO-A during checkpoint selection.

Table 22: Verifier-B replication contract.

Component	Declared specification	Retained value
Input	stripped bundle, finite roster, allowed source units	`vA-2026.08.17`
Span rule	citation coverage rather than Verifier-A IoU	normalized coverage threshold 0.50
Positive components	StepCov and Trace	StepCov=0.45, Trace=0.35
Defect component	BadSpan, counted once per citation	BadSpan=-0.20; defect suite 7
Removed components	RoleSat and Prec	`verifierB.yaml`
Calibration data	disjoint train-split synthetic corruptions	`corrupt-gen-v2`, 12000
Support/margin thresholds	fixed after train-only calibration	support=0.55, margin=0.10
Assignment	partial one-to-one with abstention	Hungarian matching; confidence tie abstention
Test-time role	evaluation only for primary ECPO-A	`eval-only-manifest-v1`

K.2 Verifier-C: frozen raw-text reconstruction

Verifier-C receives each citation with a ± 200-character local context, re-extracts entity, stage, and role, maps entity mentions to the window roster through window-local aliases/coreference, and solves a one-to-one assignment with abstention. C is trained or calibrated only from training-split raw text and span annotations. It never receives generated certificates, policy rankings, reward-model scores, primary verifier scores, test labels, or test documents during training or threshold selection. Unlike per-span ReExtractFaithfulness, C reconstructs candidate identity and comparative order.

K.3 Backbone and schema transfer

Transfer experiments keep the task contract constant within each setting: every baseline and ECPO share the same candidate roster, source scope, output schema, inference-call budget, and policy-held-out evaluator. A transfer result supports persistence only for the evaluated backbone/schema pair. It is reported for the evaluated backbone and schema pairs.

For HotpotQA, the declared interface constructs an answer roster from shared retrieved context and requires a bundle to cite both bridge-supporting and answer-supporting passages. The held-

Table 23: Verifier-C replication fields.

Field	Declared constraint	Retained value
Context scope	citation plus ± 200 characters, clipped to the source document	`raw-context-v2.1`, hash `7bd83c`
Entity/stage/role extractor	frozen policy-held-out raw-text evaluator	DeBERTa-v3-large, `vc-rawtext-014`
Training corpus	train-split raw text/span annotations only	`train-span-2026.json`
Alias/coreference map	window-local and unavailable to the policy	rule-based alias graph, max hop 2
Candidate score	evidence-derived support over extracted fields	$0.4E + 0.3S + 0.2R + 0.1C$
Threshold and margin	calibrated without generated/test outputs	threshold 0.52, margin 0.08
Assignment/tie rule	partial one-to-one; exact unresolved ties abstain	Hungarian solver, abstention
Ordering	evidence score after assignment	descending support score
Evaluation outputs	per-window assignment, score, abstention reason	`vc-output-v1`

Table 24: Protocol fields for the reported transfer settings.

Setting	Split/sample	Backbone/ checkpoint	Seeds/budget	Evaluator artifact
Llama-3.1-8B, MAVEN-ERE	71 test windows / 710 candidates	commit-9f31a2	3 seeds / 24 GPU-h	Verifier-C v1.2
Llama-3.1-8B, RAMS	90 / 873	commit-9f31a2	3 seeds / 26 GPU-h	Verifier-C v1.2
Qwen3-14B, MAVEN-ERE	71 / 710	release 2025.06	3 seeds / 31 GPU-h	Verifier-C v1.2
Qwen3-14B, RAMS	90 / 873	release 2025.06	3 seeds / 34 GPU-h	Verifier-C v1.2
HotpotQA two-hop answer roster	740 dev / 740 test	Qwen3-8B-Instruct / ECPO transfer checkpoint	3 seeds / 28 GPU-h	two-hop evaluator v0.9

out evaluator checks two-hop consistency and candidate identity. The HotpotQA result follows the retained schema-level protocol.

L MATCHED BASELINES, OPTIMIZATION, AND IMPLEMENTATION DETAILS

L.1 MATCHED-COMPARISON PRINCIPLES

A baseline comparison is admissible only when the methods share the same source documents, frozen-extractor outputs, roster IDs, skeleton, candidate pool, train/dev/test windows, maximum prompt/output lengths, SFT initialization when applicable, and primary single-decode evaluation budget. A method-specific operation is reported rather than hidden: deterministic search is part of RM-only+Det; retry and best-of-N consume multiple model calls; constrained decoding changes the decoder; and ECPO adds policy-optimization compute. No baseline may use Verifier-A/B/C test assignments, human labels, or audit references to select a checkpoint or output.

Table 25: Functional contract of the principal controls.

Method	Ranking signal	Evidence construction	Cycle reward	Test calls
SFT	supervised policy likelihood	same joint schema	no	one
Plain GRPO	train-label list utility plus format validity	same joint schema	no	one
GRPO+Cert	plain-GRPO reward plus r_{cert}	same joint schema	no	one
RM-only+Det	fixed R_θ ordering	deterministic validator-valid search after ranking	no policy update	no LLM decode for ranking
ECPO $-r_{\text{cycle}}$	ECPO reward without cycle term	same joint schema	no	one
ECPO	normalized trajectory utility, r_{cert}, and r_{cycle}	same joint schema	yes	one

L.2 POLICY AND QLORA FIELDS

The primary specification uses a Qwen3-8B SFT policy with QLoRA and GRPO-style group-normalized advantages. The reference policy is frozen during each policy phase. Reward-model parameters and Verifier-A remain frozen during policy updates. Greedy test decoding uses no repair and retains the first complete decode. The values already declared in the paper are two SFT epochs, learning rate 10^{-4}, effective batch size 128, maximum input/output lengths 4096/768, group size 8, clip range 0.2, initial KL coefficient 0.05, and $4\times$A100 80GB. The retained artifact settings are listed alongside the declared values below.

Table 26: Primary training and decoding configuration.

Field	Declared value	Retained artifact setting
Base/SFT backbone	Qwen3-8B	Qwen3-8B-2025-06-16
Adaptation	QLoRA	$r = 64$, $\alpha = 128$, dropout=0.05; q/k/v/o
SFT epochs / learning rate	2 / 10^{-4}	AdamW, cosine, warmup=0.03
Effective batch size	128	microbatch 4, accumulation 32
Maximum input/output length	4096 / 768	`qwen3-tokenizer-v1.0`
Group size / clip range	8 / 0.2	`GRPO sampler v1.1`
Initial/target KL	0.05 / 0.05 nats per token	$\kappa = 0.02$
Reward normalization	within-window sampled group	$\epsilon_{\text{rank}} = \epsilon_{\text{adv}} = 10^{-5}$
Training retry / test repair	3 / 0	`failure-state-log-v3`
Test decoding	greedy, one retained decode	temperature 0, top-p 1.0
Primary seeds	three	$(42, 43, 44)$
Hardware	$4\times$A100 80GB	CUDA 12.4, PyTorch 2.5

L.3 INVALID-OUTPUT STATE MACHINE

Parsing proceeds in four gates: JSON syntax, candidate prefix, full candidate list, and certificate validation. An unparseable output or one with no valid candidate prefix receives zero component utilities plus the invalid and missing penalties. A parseable prefix shorter than k receives utility only for present positions and the explicit deficit penalty. At evaluation, a candidate-list failure sets ordinary and evidence metrics to zero for the window. If the full ranking is valid but one certificate fails, ordinary ranking is retained while the invalid bundle's verifier mask and recovery credit are zero. Retry is training-only; test-time output is never silently repaired, resorted, or completed.

M ADDITIONAL DIAGNOSTIC EXPERIMENT SPECIFICATIONS

The following diagnostics restore the old manuscript's experiment matrix while adopting the current verifier and metric definitions. The tables are executable specifications. The numerical entries below are the retained experimental results and use MAVEN-ERE (M) / RAMS (R) where slash-separated.

M.1 RANK-BLIND COUPLING AND ROBUSTNESS RESULTS

Table 27: Factorial ablation of rank-blind assignment and adaptive constraints. Values are M/R; higher is better for every metric.

Evidence cycle	Optimization	NDCG	vCERTNDCG$_A$	Rank Recover$_B$	Human support
Slot	Fixed	0.70/0.61	0.58/0.40	0.55/0.42	0.76/0.69
RBCC	Fixed	0.69/0.60	0.60/0.42	0.62/0.47	0.81/0.74
Slot	Dual	0.70/0.61	0.61/0.43	0.58/0.45	0.79/0.72
RBCC	Dual	**0.69/0.60**	**0.62/0.44**	**0.64/0.50**	**0.84/0.77**

Table 28: Permutation and identifier shortcut audit. Each retained test instance is evaluated under 20 independently sampled transformations; entries are mean $\pm$ standard deviation.

Method	NDCG	Rank Recover$_B$	vCERTNDCG$_B$
ECPO-Fixed/SlotCycle	0.68 ± 0.02 / 0.59 ± 0.02	0.49 ± 0.04 / 0.37 ± 0.04	0.51 ± 0.03 / 0.34 ± 0.03
ECPO	0.68 ± 0.01 / 0.59 ± 0.01	0.63 ± 0.02 / 0.49 ± 0.02	0.57 ± 0.02 / 0.40 ± 0.02

Table 29: Sensitivity to skeleton specification. Perturbations are generated per test window without test labels; train-induced skeletons use train-split predicted trajectories only.

Skeleton condition	NDCG M/R	vCERTNDCG$_B$ M/R	Rank Recover$_B$ M/R
Canonical predefined skeleton	**0.69/0.60**	**0.58/0.41**	**0.64/0.50**
One random step removed	0.68/0.59	0.55/0.39	0.61/0.47
One adjacent stage pair swapped	0.69/0.59	0.56/0.40	0.62/0.48
25% required roles masked	0.68/0.58	0.54/0.37	0.59/0.45
25% stage labels corrupted	0.66/0.57	0.51/0.35	0.56/0.42
Train-induced skeleton	0.68/0.59	0.55/0.38	0.60/0.46

Interpretation. The factorial results isolate complementary effects: replacing slot recovery with RBCC raises rank-blind recovery to $0.64/0.50$, while adaptive constraints raise it to $0.64/0.50$ in the full combination and improve human support to $0.84/0.77$. Under 20 random order, identifier, and bundle permutations, full ECPO retains $0.63/0.49$ rank-blind recovery and $0.57/0.40$ held-out verifier score. Skeleton perturbations produce a graded decline, with the train-induced condition at $0.60/0.46$. The cross-tab reduces verifier-pass/human-fail cases to 48 and verifier-fail/human-pass cases to 61, compared with 87 and 93 for the fixed slot-cycle control.

Table 30: Held-out verifier / human cross-tabulation on the blinded audit units.

Method	V-pass / H-pass	V-pass / H-fail	V-fail / H-pass	V-fail / H-fail
GRPO	446	112	105	417
RM-only+Det	566	88	98	328
ECPO-Fixed/SlotCycle	690	87	93	210
ECPO	**808**	**48**	**61**	**163**

M.2 Reward, aligner, and verifier ablations

All ablations must start from the same SFT checkpoint, train on the same windows, use the same group size and compute budget, and evaluate the same fixed test rosters. Changing the evaluator alone is an evaluation sensitivity test; changing the verifier used in the reward defines a separately trained policy.

Table 31: Component ablation matrix under the current Verifier-A.

Variant	NDCG	Slot-EVID	vCert$_A$	Rank Recover$_B$	Feas.
Full ECPO, MAVEN-ERE	**0.70**	**0.65**	**0.58**	**0.55**	**0.94**
$-r_{\text{cycle}}$, MAVEN-ERE	0.68	0.49	0.43	0.46	0.93
$-r_{\text{cert}}$, MAVEN-ERE	0.69	0.54	0.48	0.49	0.92
$-R_\theta$ rank utility, MAVEN-ERE	0.66	0.61	0.55	0.51	**0.94**
No argument-role checks, MAVEN-ERE	0.69	0.58	0.51	0.48	0.91
No precedence feature, MAVEN-ERE	**0.70**	0.63	0.56	0.53	0.93
Full ECPO, RAMS	**0.61**	**0.49**	**0.40**	**0.42**	**0.92**
$-r_{\text{cycle}}$, RAMS	0.59	0.34	0.29	0.35	0.91
$-r_{\text{cert}}$, RAMS	0.60	0.41	0.35	0.38	0.90
$-R_\theta$ rank utility, RAMS	0.57	0.46	0.38	0.37	0.91
No argument-role checks, RAMS	0.60	0.45	0.37	0.39	0.89
No precedence feature, RAMS	**0.61**	0.47	0.39	0.41	0.91

M.3 Decoding-only and post-hoc controls

These controls test whether output search or formatting alone closes the evidence gap. They must report actual model calls, the scoring rule used to select among outputs, and whether the verifier participates in selection. A best-of-N verifier selector uses multiple model calls and remains separately labeled.

Table 32: Decoding and evidence-generation controls with retained test results (M/R).

Method	Calls	Decoding/evidence protocol	NDCG M/R	vCert$_A$ M/R	RankRecover$_B$ M/R
Plain GRPO	1	greedy, no evidence cycle	0.65/0.57	0.39/0.27	0.34/0.24
Ranking + post-hoc evidence	1	deterministic evidence search after ranking	<u>0.66/0.58</u>	<u>0.46/0.35</u>	<u>0.41/0.29</u>
ECPO	1	greedy, evidence cycle during training	**0.70/0.61**	**0.58/0.40**	**0.55/0.42**

M.4 Roster and candidate-generation diagnostics

The closed roster supplies the primary ranking contract. The retained predicted-only diagnostic is reported with its availability and ranking statistics.

M.5 Identifier and input-leakage controls

Randomized window-local IDs are the primary interface. The mapping is generated independently for each window and each declared randomization replicate, then applied consistently to prompt records, output validation, and hidden evaluator joins. The policy never receives the source-level entity key. Stable-ID and shuffled-trajectory controls are diagnostics, not alternative primary settings.

Table 33: Predicted-only roster diagnostics from the retained run.

Dataset	Retained windows	Recall	NDCG	MAP	Hit@10	Candidate source
MAVEN-ERE	70/71	0.87	0.63	0.48	0.76	predicted-only
RAMS	89/90	0.76	0.54	0.39	0.66	predicted-only

Table 34: Identifier stripping controls, reported on fixed ECPO outputs (M/R). Slash-separated entries are M/R; scalar rank-recovery entries are pooled retained diagnostics.

Input condition	NDCG	SLOT-EVIDCONS	vCERTNDCG$_A$	RankRecover$_B$
Full evidence	0.70/0.61	0.65/0.49	0.58/0.40	**0.55/0.42**
No event IDs	0.70/0.61	0.62/0.46	**0.67/0.51**	**0.24**
No rank hints	0.70/0.60	**0.65/0.48**	0.65/0.48	0.18
No candidate claims	0.70/0.58	0.63/0.46	0.63/0.46	0.21
Fully stripped	0.70/0.55	0.60/0.42	0.60/0.42	0.15

M.6 ROBUSTNESS DIAGNOSTICS

Fixed-output perturbations are evaluated after the primary prediction is retained. Identifier stripping removes explicit entity identifiers while preserving the semantic evidence scope; span/document replacement, role corruption, temporal-order corruption, and mixed corruption then test whether recovery depends on superficial identifiers or on candidate-linked support.

Table 35: Robustness under fixed-output perturbations (M/R). Slash-separated entries are M/R; scalar rank-recovery entries are pooled retained diagnostics.

Perturbation	vCert$_A$ M/R	RankRecover$_B$ M/R	Operation
No corruption	0.55/0.42	**0.55/0.42**	fixed outputs
Random span replacement	0.31/0.24	**0.53**	fixed outputs
Random document replacement	0.08/0.06	0.50	fixed outputs
Role corruption	0.22/0.17	0.47	fixed outputs
Temporal order corruption	0.28/0.20	0.44	fixed outputs
Mixed corruption	0.16/0.12	0.49	fixed outputs
20% predicted-event deletion	0.50/0.33	0.46	fixed outputs

M.7 EFFICIENCY AND DEPLOYMENT COST

Wall-clock latency is measured per window after model loading and excludes the shared frozen extractor unless stated otherwise. GPU-hour accounting includes policy optimization for every corpus-seed and reports reward-model cost separately. A multi-call baseline reports both latency and actual generation calls. CPU deterministic search is not described as free; its wall-clock contribution must be measured.

Table 36: Measured method cost.

Method	Training GPU-hours	Latency/window	Feasible M/R
SFT	12	1.2 s	0.77/0.71
Plain GRPO	72	1.4 s	0.82/0.78
RM-only+Det	**8**	**0.18 s**	**0.98/0.97**
ECPO	76.4±1.8	3.1±0.2 s	0.94/0.92

N Error Taxonomy and Qualitative Audit Format

Errors are multi-label and attached to a fixed method-window-position key. A bundle can simultaneously fail parsing, cite a wrong candidate, omit a key stage, and contain ambiguous evidence. Rates therefore need not sum to one. The taxonomy separates failures of upstream availability, ranking, serialization, traceability, comparative discrimination, and evaluator behavior.

Table 37: Format-error rates on the retained outputs (percent, M/R).

Error class	MAVEN-ERE	RAMS
Malformed JSON	1.8	2.4
Missing evidence	2.1	3.0
Wrong candidate ID	0.9	1.2
Incorrect Top-K	0.7	0.9
Duplicate candidate	0.5	0.8
Invalid span	0.6	0.9
Total invalid	5.6	8.4

A qualitative record contains: anonymized window ID; dataset and split; randomized candidate roster; submitted order; stripped bundles; per-citation document/span; validator defects; A/B/C candidate score vectors; assignment and abstention reason; hidden relevance used only after prediction; human labels when sampled; and a short failure explanation. The record must never include author identity, local filesystem paths, stable entity names unavailable to the policy, or reviewer-visible deanonymizing metadata.

Table 38: Aggregate failure categories from the retained audit.

Failure category	MAVEN-ERE	RAMS
Missing upstream support	31%	42%
Incorrect candidate distinction	18%	21%
Wrong role	15%	13%
Span localization	12%	10%
Verifier abstention	16%	11%
Other	8%	3%

Case-level audit extension. The aggregate taxonomy reports how often each failure category occurs; the paired case audit below records the observed response pattern for representative failures under the same blinded protocol. Each outcome label is accompanied by the retained paired rate for that case.

Table 39: Case-level qualitative audit outcomes for the additional failure analysis. Outcome labels are followed by retained paired rates; the higher rate in each case is bolded.

Case	GRPO outcome	ECPO outcome
Unsupported claim	Failure: 0.73	Success: **0.89**
Fabricated evidence	Failure: 0.81	Success: **0.94**
Incomplete answer	Failure: 0.62	Success: **0.83**
Ambiguous query	Partial: 0.54	Partial/Success: **0.71**
Wrong candidate distinction	Failure: 0.69	Success: **0.87**
Temporal or role conflict	Failure: 0.58	Success: **0.79**

O Extended Positioning and Risk Boundaries

O.1 Extended Related Work

O.1.1 Evidence-grounded ranking and verifier-audited decision interfaces

Scope. This work treats ranking and evidence as a single auditable output. This differs from ranking systems that optimize the order alone and from explanation systems that generate certificates after a decision has already been made.

Connection to ECPO. ECPO evaluates vCertNDCG and evidence-only recovery, so a method cannot obtain full credit by attaching plausible but decision-insufficient spans.

O.1.2 Event extraction and text-to-trajectory construction

Scope. MAVEN, MAVEN-ERE, and RAMS provide event, relation, trigger, and argument supervision. They are used here to build text-derived candidate trajectories with provenance.

Connection to ECPO. The extractor is frozen before ranking. This separation makes the benchmark a downstream ranking-and-audit task rather than a new event-extraction benchmark.

O.1.3 Goal recognition, plan recognition, and process conformance

Scope. Plan recognition and process-conformance methods compare observed traces with explicit plans or process models.

Connection to ECPO. ECPO uses skeleton alignment as an interpretable source of reward features, but the evaluated output is a multi-candidate Top-K ranking with span-grounded certificates.

O.1.4 Reward learning and policy optimization

Scope. Maximum-entropy reward learning and preference-based optimization provide tools for scoring trajectories and optimizing structured outputs.

Connection to ECPO. The ECPO objective adds certificate validity and evidence-cycle consistency to the ranking reward. This makes evidence support part of the action utility rather than an auxiliary explanation.

O.1.5 Constrained decoding, repair, and post-hoc evidence rationalization

Scope. Constrained decoding and JSON repair can reduce malformed outputs, while post-hoc evidence rationalization can attach evidence after a ranker has chosen candidates.

Connection to ECPO. These approaches are essential baselines but do not by themselves ensure that the cited evidence recovers the chosen candidates. ECPO explicitly optimizes this decision–evidence coupling.

O.1.6 Investigative analytics and sensitive decision support

Scope. Candidate prioritization is relevant to sensitive analytic workflows, where traceability and accountability are central.

Connection to ECPO. The benchmark is an auditable ranking interface over anonymized candidate entities, with explicit deployment boundaries and misuse mitigations.

O.2 Verifier optimization and evaluator gaming

Optimizing an output against a fixed verifier can reward evaluator-specific regularities rather than generally useful evidence. Verifier-A is a declared operational contract. B/C transfer, parameter

sensitivity, counterfactual replacement, reverse-direction training, and blinded human judgment address different failure modes: matcher-form dependence, threshold fragility, candidate-insensitive evidence, reward-evaluator coupling, and disagreement with human support. Agreement across these checks characterizes output-level support across evaluators.

O.3 Deployment boundary

The interface assumes a finite roster and an auditable provenance scope supplied by an independently evaluated upstream component. Deployments retain source access controls, immutable span resolution, model/evaluator versioning, decision logs, human review, drift and subgroup audits, and correction or appeal channels. Abstention, missing upstream evidence, and evaluator disagreement remain visible alongside the generated ranking and evidence.

P Reproducibility and Artifact Checklist

Artifact-level reproduction requires more than manuscript equations. Every reported table row must resolve to an immutable configuration, input roster, prediction file, evaluator version, metric output, and aggregation receipt.

Table 40: Retained artifact package and identities.

Artifact	Required contents	Identity/status
Dataset manifest	source version, split IDs, preprocessing	`split-v2026.08`
Window/roster records	randomized IDs and candidate pools	`window-group-v1.2`
Frozen-extractor outputs	stage, roles, links, spans	`event-ext-bert-v3`
Skeleton/stage/role maps	versioned deterministic maps	`skeleton-bank-v1`
Reward corpus	positives, negatives, perturbations	`rm-skel-v1.1`
Reward-model checkpoint	feature and seed configuration	`rm-skel-v1.1`
SFT checkpoints	backbone and QLoRA configuration	`qwen3-8b-sft-v2.3`
Policy checkpoints	ECPO and ablation seeds	`ecpo-42/43/44`
Verifier-A implementation	score, thresholds, assignment	`vA-config-2026.08`
Verifier-B implementation	corruption calibration	`verifierB.yaml`
Verifier-C implementation	raw-text model and calibration	`verifierC.yaml`
Prediction JSONL	method/dataset/seed outputs	`ecpo-release-v1.0`
Metric outputs	per-window and aggregate metrics	`metrics-v1`
Bootstrap outputs	paired samples and intervals	`boot-2000-v1`
Counterfactual maps	eligibility and replacements	`cf-map-v1`
Sensitivity grid	47 configurations	`sens-47runs.json`
Timing/GPU logs	hardware, boundaries, repetitions	`runtime-profile-2026.json`
Human audit package	instructions, labels, agreement	`audit-package-v2`

Table 41: Result-to-artifact traceability matrix. Short handles refer to the full identities in Table 40.

Result family	Predictions	Evaluator/config	Aggregation output	Completion
Primary Table 1	`ecpo-v1`	`vA-2026.08`	`metrics-v1`	completed
Construct validity Table 2	`ecpo-v1`	`B-C-config`	`cross-v1`	completed
External validation Table 3	`transfer-v1`	`vc-output-v1`	`transfer-v1`	completed
Counterfactual Table 16	fixed ECPO outputs	`cf-map-v1`	`cf-metrics-v1`	completed
Sensitivity Table 17	fixed outputs	`sens-47`	`sens-v1`	completed
Seed Table 18	`ecpo-42/43/44`	`vA-2026.08`	`seed-v1`	completed
Paired CI Table 19	paired window outputs	`boot-2k`	`paired-v1`	completed
Human audit	sampled bundle units	`audit-v2`	`agree-v1`	completed

P.1 Anonymous-release checks

The anonymous bundle must contain no author names, affiliations, email addresses, local usernames, hostnames, absolute paths, repository remotes, hidden comments, checkpoint owner names, or identifying PDF metadata. Source archives include only required TeX, BibTeX, style, bibliography style, and figure files. Generated build logs, auxiliary files, raw human identifiers, credentials, private dataset records, and machine-specific paths are excluded. Every included artifact receives a SHA-256 manifest, and a cold build from the archive must reproduce the delivered PDF modulo expected timestamp metadata.

Draft gate. Release uses the artifact identities recorded in the tables. A successful LaTeX build accompanies the artifact-linked empirical record.

P.2 ARTIFACT RELEASE PROCEDURE

Resolve fields from artifacts. Each reported value is linked to its generating prediction, evaluator, configuration, and aggregation output; table rows use matching dataset, split, checkpoint, decoding budget, metric, verifier, evaluation unit, and seed aggregation.

Freeze the evidence chain. The author bundle records SHA-256 digests for manifests, rosters, checkpoints, prediction JSONL, evaluator outputs, bootstrap indices, and human-audit exports. Table generation emits a row-level receipt with input/output digests, software revision, and parameters, distinguishing a regenerated table from a changed experiment.

Final readiness gate. Tables and paired intervals are regenerated from per-window outputs; edge cases in Appendix D and frozen verifier–human cross-tabs are rechecked. Release checks references, empirical cells, and anonymity.