POLYPRM: MANDATE-CONDITIONED PROCESS EVALUATION FOR ROLE-SPECIFIC AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Process reward models (PRMs) typically evaluate intermediate reasoning and actions against a single objective. For role-specific agents, however, the same candidate process may have different values under different decision responsibilities. We introduce RIVA-Bench, a matched benchmark that holds decision contexts and candidate slates fixed while varying task-defined mandates and selector-independent, provenance-tracked utility targets. RIVA-Bench contains 111,805 labeled candidate processes, 12,872 decision states, and 50 roles across eight tasks. We also introduce PolyPRM, a 1.5B mandate-conditioned evaluator that learns across roles while preserving their distinct candidate orderings. PolyPRM achieves the highest direct-selection score on all eight tasks and outperforms frontier LLM judges supplied with the same complete mandate. It improves frozen ReAct and Debate agents over their base selection policies in all 16 evaluated settings and attains the best or tied-best non-oracle outcome in 15. Counterfactual ablations show that its selections follow mandate semantics rather than role names. Joint training further matches or exceeds routed role-specific experts across all six datasets while using one checkpoint and $1/r$ of their aggregate training data.

1 INTRODUCTION

Large language model agents act through sequences of reasoning, information gathering, tool use, and interaction with external environments.Yang et al. (2025); Yao et al. (2022); Liu et al. (2024) An error at one step can redirect the remaining trajectory, so reliability depends on intermediate decisions as well as final outcomes. Terminal evaluation can overlook unsupported reasoning or stale evidence that happens not to change the result, and a terminal reward cannot identify which intermediate decision caused an outcome, since later actions may compensate for an earlier mistake and the outcomes of unchosen alternatives remain unobserved.Fan et al. (2026); Zheng et al. (2025) Process reward models (PRMs) address this gap by scoring intermediate reasoning steps and actions for training, search, and candidate selection.Lightman et al. (2024); Xi et al. (2026); Wang et al. (2024c); Setlur et al. (2025); Sohn et al. (2026) Their effectiveness depends on a clear definition of process value and sufficient supervision to learn it, obtained either from human annotations and explicit rubrics, which require task-specific design,Lightman et al. (2024); Kim et al. (2024) or from sampled trajectories and terminal outcomes, which incur additional rollouts and must infer local quality from delayed signals.Wang et al. (2024c); Xi et al. (2026)

Beyond label acquisition, the organization of supervision determines what a PRM can reuse. Task-specific resources provide precise labels for one task format and verification rule, but their knowledge transfers poorly even to related settings.Zhu et al. (2025); Zeng et al. (2025); Zheng et al. (2025) Domain-level resources broaden coverage, yet combining their supervision under one rubric or scalar target can obscure differences in what constitutes a good decision for different roles.Wang et al. (2024a); Fu et al. (2026) Consider a climate negotiation in which the same state and candidate commitments are evaluated from each stakeholder's mandate: a market-oriented mandate ranks cross-border carbon trading highest, while a transition-oriented mandate prefers ending new fossil-fuel investment. Process value is therefore not a property of the candidate process alone. We use *decision role* to denote the task-specific responsibility under which an agent acts, and *decision mandate* for the corresponding objective, authority, relevant evidence, constraints, and consequences.Wang et al. (2024b); Lai et al. (2026) The resulting challenge is to reuse process knowledge across role-specific agents while preserving the candidate ordering required by each mandate.

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

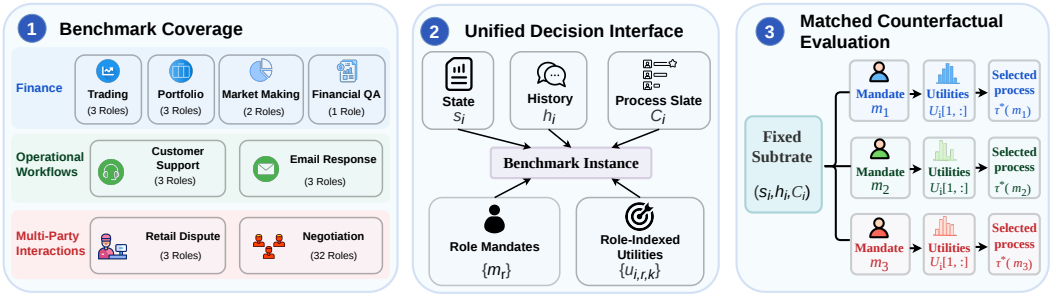

Figure 1: RIVA-Bench holds the decision context and candidate slate fixed while evaluating each candidate under role-indexed mandates and independently grounded outcomes.

This motivates *RIVA-Bench*, short for *Role-Indexed Value Assessment*, which holds the state, interaction history, and candidate slate fixed while varying the active mandate and evaluates each selected candidate against a role-specific outcome constructed independently of the selector. This matched construction makes mandate dependence identifiable: changes in state, evidence, or proposal quality cannot explain an observed ranking reversal. RIVA-Bench contains 111,805 labeled candidate processes grouped into 12,872 decision states and 50 task-specific decision roles, spanning eight tasks in financial decisions, customer-facing workflows, retail disputes, email response, and multi-party negotiation, and including a single-role compatibility setting and NegoBench's benchmark-native roles, with every task retaining its native action and utility semantics.Chen et al. (2021); Barres et al. (2025); Benac et al. (2026) The same construction extends to any domain with a shared decision space, multiple stakeholder responsibilities, and outcomes that can be grounded independently of the evaluated selector. We will release the seven redistributable task configurations with their construction and evaluation code; Email Response remains privacy-restricted and is reported only in aggregate. Figure 1 summarizes the evaluation interface.

The same matched interface supplies the supervision used to train *PolyPRM*, a shared process evaluator that estimates $V_\theta(s, h, \tau \mid m)$, where s and h denote the decision state and interaction history, τ is a candidate process, and m is the active mandate. PolyPRM learns jointly from the decision roles available within a task configuration while retaining their mandate-specific utility targets, and at deployment reranks proposals generated by a frozen role-specific agent under that agent's active mandate. Each agent remains assigned to one decision role; only the evaluator and its process knowledge are shared. The contribution is therefore not conditioning alone: PolyPRM targets intermediate candidate processes under task-defined responsibilities, and the matched construction makes the effect of that conditioning identifiable against outcomes defined independently of the model.

Matched counterfactuals show that mandate conditioning has a directional rather than merely regularizing effect. On Retail Dispute and Email Response, removing the mandate reduces mandate-counterfactual accuracy to zero; on Retail Dispute, selected utility falls from 0.984 with the full mandate to 0.648 without it, and further to 0.433 and 0.339 under wrong and shuffled mandates. PolyPRM therefore follows the ordering implied by the supplied mandate rather than simply becoming less certain when role information is removed or corrupted, while mandate ablations produce little change in low-conflict settings where the supported roles largely agree, so the effect tracks measured role disagreement. Using a Qwen2.5-1.5B backbone adapted with LoRA, PolyPRM achieves the best direct-selection result on all eight tasks, including against frontier LLM judges supplied with the same complete mandate, obtains the best or tied-best non-oracle outcome in 15 of 16 downstream agent settings, and matches or exceeds routed role-specific experts on all six evaluated datasets using a single checkpoint.

Our contributions are threefold.

- **RIVA-Bench.** A matched benchmark of 111,805 labeled candidate processes, 12,872 decision states, and 50 task-specific decision roles across eight tasks.
- **PolyPRM.** We formulate process value as mandate-conditioned and develop a shared evaluator that learns across role-specific agents while retaining their distinct utility targets.
- **Empirical evidence.** We establish the directional effect and boundary of mandate conditioning, improve direct selection and frozen-agent decisions, and show that joint training can replace a collection of routed role experts with one checkpoint.

2 Related Work

2.1 Process Reward Models and Agent Evaluation

Process reward models provide feedback on intermediate reasoning rather than relying only on terminal outcomes. Early studies compared process- and outcome-based supervision, while PRM800K demonstrated the value of large-scale human step annotations Uesato et al. (2022); Lightman et al. (2024). Math-Shepherd instead derives supervision from sampled continuations Wang et al. (2024c), and subsequent methods estimate future progress or generate verification rationales before assigning rewards Setlur et al. (2025); Khalifa et al. (2025); Zhao et al. (2026). ProcessBench evaluates whether these models can localize reasoning errors beyond their training distributions Zheng et al. (2025).

Interactive-agent benchmarks evaluate action sequences through environment transitions, task completion, and policy compliance Liu et al. (2024); Zhou et al. (2024a); Barres et al. (2025). Recent work extends process supervision to agent actions and tool-use trajectories Xi et al. (2026); Li et al. (2026); Fan et al. (2026). These methods provide increasingly granular evaluation, but generally define process quality relative to one task objective. Context-conditioned reward models relax this for final responses and are evaluated on paired examples whose preference reverses with the context Pitis et al. (2024); PolyPRM applies the same identification logic to intermediate candidate processes under fixed context and candidate slates, and scores the selected candidate against held-out task-native outcomes rather than preference labels.

2.2 Roles and Decision Mandates

Roles can describe social expectations or connect organizational functions to responsibilities, authority, and constraints Biddle (1986); Sandhu et al. (1996). In LLM systems, roles commonly specify character profiles Lai et al. (2026); Wang et al. (2024b), functional specializations in multi-agent collaboration Li et al. (2023); Hong et al. (2024), or identities and private goals in social simulations Zhou et al. (2024b). These formulations primarily determine how an agent behaves.

PolyPRM instead uses a decision role to determine whose objective and constraints govern process evaluation: each agent retains one active mandate, while the evaluator is shared across roles. The closest evaluator-side approach is MRRG, which elicits criteria from multiple evaluator roles and consolidates them into one rubric-based score, whereas PolyPRM preserves role-indexed utilities throughout learning and therefore does not collapse conflicting candidate orderings Fu et al. (2026).

3 RIVA-Bench: Role-Indexed Value Assessment for Agent Processes

Existing process-reward datasets typically associate each candidate process with a single notion of quality. Adding role descriptions to independently collected examples does not establish that an observed preference difference is caused by the role because the state, available evidence, candidate set, or labeling criterion may also have changed. We therefore introduce ***RIVA-Bench***, short for *Role-Indexed Value Assessment*, which constructs matched decision instances for evaluating process value under explicit decision mandates.

3.1 Benchmark Construction

RIVA-Bench converts task trajectories, operational records, benchmark environments, and structured cases into matched decision instances. For each decision point i, we preserve the available state s_i and history h_i and construct a fixed Process Slate $\mathcal{C}_i = \{\tau_{i1}, \ldots, \tau_{iK}\}$ containing candidate actions, reasoning programs, response drafts, or trajectory continuations. Each slate is constructed independently of the evaluated selector and shared across methods.

Each supported decision role is represented by a natural-language mandate m_r grounded in its task-specific objective, authority, constraints, operational priorities, and consequences. The mandate defines how processes should be evaluated rather than specifying a conversational persona. We include roles only when their decision responsibilities are supported by the source task or operational

setting. Figure 5 presents complete mandate examples and their resulting role-dependent utility orderings.

3.2 Matched Role-Indexed Supervision

A RIVA-Bench instance is represented as $\mathcal{B}_i = \left(s_i, h_i, \mathcal{C}_i, \{m_r, \mathbf{u}_{i,r}\}_{r=1}^{R_i}\right)$, $u_{i,r,k} = A_r(s_i, h_i, \tau_{ik})$, where $\mathbf{u}_{i,r}$ contains the utilities of all candidates under role r, and A_r is the corresponding outcome anchor. Collecting these utilities across roles and candidates produces a Role–Utility Matrix for each decision context.

The state, history, and Process Slate remain fixed across roles, while the active mandate and its utility row vary. Every role therefore evaluates the same evidence and candidate alternatives. If two roles induce different candidate orderings, the difference can be attributed to their mandates rather than to changes in the underlying decision problem. The same candidate slates are also reused across evaluated selectors, separating the quality of the selector from the quality of the available proposals.

These matched comparisons define *mandate counterfactuals*. Some contexts produce similar candidate rankings across roles, while others contain ranking reversals in which the preferred process changes with the mandate. The former serve as natural controls. The latter test whether an evaluator can change its ranking in response to the active decision criterion without any accompanying change in context or proposal quality.

3.3 Outcome Grounding and Coverage

RIVA-Bench distinguishes candidate-level supervision from the held-out outcomes used to score a selector. Each utility target uses the strongest task-native anchor available, including gold-answer correctness, executable rules, simulator returns, observed decisions, forward outcomes, and benchmark-native payoffs. When these signals do not fully specify process quality, they are combined with an LLM-assisted, mandate-conditioned quality judgment, with the source of every component retained as provenance. Of the 111,805 candidate targets, 49,446, or 44.2%, use this hybrid construction. The remaining targets contain no LLM-derived component. Utilities are normalized to $[0, 1]$ for training but retain their task-specific meanings. Table 1 separately marks evaluator-free test columns, where selected candidates are scored only by gold answers, executable rules, observed decisions, or benchmark-native payoffs.

Before inclusion, targets are checked for evidential support, mandate consistency, label leakage, degenerate slates, and mandate-independent shortcuts. The resulting *Supervision Integrity Layer* preserves supported targets, corrects identifiable errors, and abstains when evidence is insufficient. On a rule-only Retail Dispute reference, a PRM trained with integrity-checked supervision reaches 0.969 selected utility, compared with 0.834 using the original labels. Appendix B.3 provides the full validation.

RIVA-Bench spans eight tasks across financial decisions, customer-facing workflows, retail disputes, email response, and multi-party negotiation. It covers candidate processes ranging from operational actions and reasoning programs to response drafts and negotiation commitments, with both weak and strong role dependence. Appendix C reports complete sources, mandates, splits, utility definitions, and construction procedures. We next instantiate this matched, role-indexed interface with PolyPRM.

4 PolyPRM: Mandate-Conditioned Process Reward Modeling

RIVA-Bench supplies the role-indexed supervision used to train PolyPRM. For each task configuration d, we train one shared evaluator across its supported decision roles $\mathcal{R}_d$. At each decision point, PolyPRM evaluates proposals for one role-specific agent under that agent's active mandate.

4.1 Mandate-Conditioned Value Estimation

For candidate process $\tau_{t,k}$, PolyPRM receives the active mandate m_r, state s_t, and history h_t. Mandate-First Process Encoding places the mandate before the decision context and candidate,

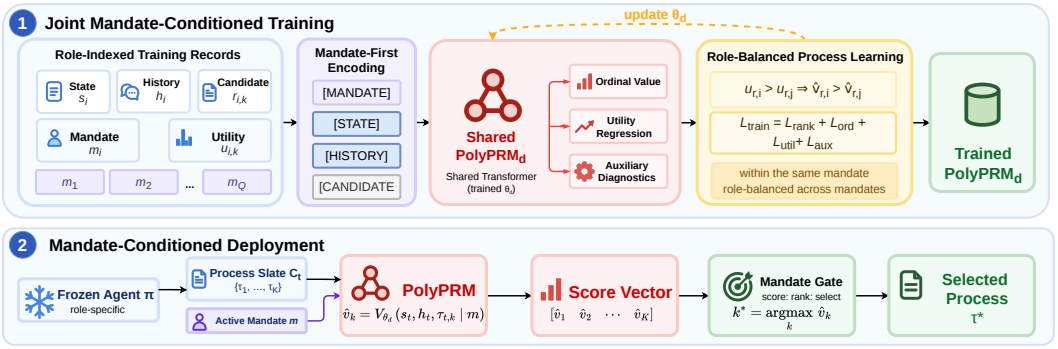

Figure 2: PolyPRM training and deployment. A shared evaluator learns mandate-conditioned process values from role-indexed supervision and selects among proposals from a frozen agent under the active mandate.

$x_{t,r,k} = m_r \,\|\, s_t \,\|\, h_t \,\|\, \tau_{t,k}$, where $\|$ denotes textual concatenation. A shared transformer backbone encodes $x_{t,r,k}$, and an ordinal-value head predicts a distribution over eleven utility levels $c_b = b/10$. The candidate score is the expected ordinal value

$$v_{t,r,k} = V_{\theta_d}(s_t, h_t, \tau_{t,k} \mid m_r) = \sum_{b=0}^{10} c_b \, p_{\theta_d}(b \mid x_{t,r,k}). \tag{1}$$

The utility target is used only as supervision and is never included in the input. Because the mandate is part of $x_{t,r,k}$, the same process may receive different scores under different decision roles. Utility-regression and auxiliary diagnostic or action-prediction heads provide additional supervision during training, while candidate selection uses the ordinal expectation in Equation 1.

4.2 Role-Balanced Process Learning

PolyPRM retains each record's mandate and role-specific utility rather than pooling different role targets into one score. Pairwise comparisons are formed only between candidates from the same state and mandate. For an ordered pair with $u_{t,r,i} > u_{t,r,j}$, the ranking loss is

$$\ell_{\text{rank}}(t, r, i, j) = -\log \sigma\left(\frac{v_{t,r,i} - v_{t,r,j} - \gamma}{T}\right), \tag{2}$$

where γ is the ranking margin and T is the ranking temperature. Tied utility targets impose no pairwise constraint.

Let $\mathcal{L}^{(r)}_{\text{rank}}$ denote the mean ranking loss for role r. PolyPRM minimizes the role-balanced objective

$$\mathcal{L}_d = \frac{1}{|\mathcal{R}_d|} \sum_{r \in \mathcal{R}_d} \left(\lambda_{\text{rank}} \mathcal{L}^{(r)}_{\text{rank}} + \lambda_{\text{ord}} \mathcal{L}^{(r)}_{\text{ord}} + \lambda_{\text{util}} \mathcal{L}^{(r)}_{\text{util}} + \lambda_{\text{aux}} \mathcal{L}^{(r)}_{\text{aux}}\right). \tag{3}$$

The ordinal term supervises the discretized utility distribution, the regression term predicts the continuous utility target, and the auxiliary term aggregates available diagnostic and action labels. Missing auxiliary targets are masked, and the ranking term is zero when a role has no eligible candidate pairs. Averaging at the role level prevents roles with more records from dominating optimization.

Joint training shares representations of the state, evidence, and process consequences across roles. Agreements provide compatible ranking constraints, while reversals over matched candidate slates associate different orderings with different mandates. The model can therefore reuse task knowledge without averaging away role-specific decision criteria.

Mandate-gated selection. At deployment, a frozen proposal engine generates a Process Slate C_t. PolyPRM scores each candidate independently under the acting agent's mandate and selects

$$\hat{k}_{t,r} = \underset{k \in \{1,\ldots,K_t\}}{\arg\max}\ V_{\theta_d}(s_t, h_t, \tau_{t,k} \mid m_r). \tag{4}$$

The selected process is returned for execution or evaluated through the task's replay protocol. This Mandate Gate changes only the selection controller and does not update the proposal model.

Table 1: Direct candidate-selection performance across eight tasks. The main table reports representative frontier judges and process evaluators, with complete results in Appendix Table 6. Bold and underlined values indicate the highest and second-highest results in the full comparison. † marks evaluator-free outcomes, and ‡ marks the privacy-restricted Email Response task.

Method	Trading	Portfolio	Market Making	Financial QA†	Customer Support†	Retail Dispute	Email Response†‡	Negotiation†
Heuristic baselines								
Random	0.642	0.699	0.644	0.153	0.500	0.500	0.621	0.462
First Candidate	0.588	0.619	0.644	0.222	0.494	0.494	0.654	0.471
Frontier LLM judges								
GPT-5.6-luna	0.730	0.749	0.683	0.867	0.686	0.600	0.655	0.474
GPT-5.6-luna + Mandate	0.625	0.750	0.683	<u>0.911</u>	0.663	0.900	<u>0.705</u>	0.766
GPT-5.6-sol	0.668	0.766	0.642	0.805	0.697	0.635	0.638	0.506
GPT-5.6-sol + Mandate	0.634	0.774	0.635	<u>0.911</u>	0.721	<u>0.964</u>	0.687	<u>0.814</u>
Claude Opus 4.8	<u>0.777</u>	0.776	0.641	0.836	0.758	0.647	0.631	0.578
Claude Opus 4.8 + Mandate	0.765	0.744	0.699	0.842	<u>0.774</u>	0.960	0.698	0.778
Existing PRMs								
Fin-PRM	0.730	0.739	0.756	0.526	0.392	0.525	0.644	0.451
ThinkPRM	0.474	0.715	0.637	0.271	0.539	0.475	0.571	0.532
ThinkPRM-FT	0.735	0.726	0.789	0.569	0.561	0.577	0.689	0.468
MRRG	0.749	0.726	0.761	0.526	0.649	0.475	0.640	0.628
Our method								
PolyPRM w/o Mandate	<u>0.777</u>	<u>0.778</u>	<u>0.818</u>	0.674	0.680	0.648	0.661	0.655
PolyPRM	**0.778**	**0.784**	**0.837**	**0.929**	**0.792**	**0.984**	**0.800**	**0.819**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate PolyPRM at two levels. Direct selection uses the eight RIVA-Bench task configurations; downstream experiments evaluate frozen agents on the corresponding source tasks—FinQAChen et al. (2021), τ^2-benchBarres et al. (2025), OlistOlist & Sionek (2018), and NegoBenchBenac et al. (2026)—under their original outcome metrics. Each task uses fixed data splits and one jointly trained PolyPRM checkpoint carrying role-specific mandates and utility targets.

Models and baselines. PolyPRM is initialized from Qwen2.5-1.5B-Instruct Qwen et al. (2025) and adapted with LoRA Hu et al. (2021). We compare it with random and heuristic selectors, general-purpose LLM judges from OpenAI OpenAI (2026), DeepSeek DeepSeek-AI et al. (2025), and Anthropic Anthropic (2026), and the Fin-PRM Zhu et al. (2025), Qwen2.5-Math-PRM-7B Zhang et al. (2025), ThinkPRM Khalifa et al. (2025), and MRRG Fu et al. (2026) process evaluators. The oracle gives the upper bound of each fixed candidate pool. LLM judges are evaluated both without role information and with the complete mandate. ThinkPRM-FT adapts ThinkPRM on our training data as an in-domain control. Fin-PRM is treated as an in-domain baseline on the finance tasks and as a transfer baseline elsewhere.

Evaluation protocols. Direct selection holds the state, interaction history, and candidate Process Slate fixed across selectors, so every method ranks the same candidates and the selected process is scored by a held-out task-native outcome. Downstream, a frozen Qwen3-8B proposal model samples six candidates at each decision point under ReAct or debate-style prompting, and the pool is generated once and reused by every controller. Selected actions are scored through outcome replay or deterministic task utilities, except on NegoBench, which executes the selected commitment in the negotiation environment.

Metrics and statistics. We report task-native metrics and compare methods only within a task: selected utility for Trading, Portfolio, Market Making, and Negotiation; answer accuracy for Financial QA; symmetrized pairwise accuracy for Customer Support and Retail Dispute; and best-of-N utility for Email Response. Downstream evaluation additionally reports normalized regret, $\mathrm{NRegret} = (u_{\max} - u_{\mathrm{selected}})/(u_{\max} - u_{\min})$, over the n_{sel} states with nonconstant candidate utilities. All results are macro-averaged across decision roles. The main tables report point estimates, while Appendix A provides paired cluster-bootstrap confidence intervals and complete details on splits, prompts, optimization, and model provenance.

Table 2: Downstream selection quality across eight tasks and two proposal strategies. Each cell reports task outcome (↑) / normalized regret (↓), with leading zeros omitted. Outcome averages over all n decision points; regret is computed over the n_{sel} points whose candidates differ in utility, so it measures how much of the attainable slate utility each controller captures. Bold and underlined mark the best and second-best non-oracle result per metric. Fin-PRM is not evaluated on the four non-finance tasks ("–"); oracle regret is .000 by definition. Random-selection values and per-task n/n_{sel} are in Appendix A.3.

		Non-learned selectors			**Transferred PRMs**		**Ours**	
Task	**Agent**	Base Agent	Self-Cons.	Task Heur.	Think PRM	Fin-PRM	**Poly PRM**	Oracle
Trading	ReAct	.347/.524	.347/.524	**.407**/**.320**	.360/.480	.344/.571	**.407**/**.320**	.451
	Debate	.347/.524	.347/.524	**.407**/**.320**	.353/.557	.341/.615	**.407**/**.320**	.451
Portfolio	ReAct	.463/.510	.463/.510	.463/.510	**.502**/.504	.426/.565	.490/**.434**	.720
	Debate	.463/.510	.463/.510	.463/.510	.478/.476	.476/.537	**.501**/**.418**	.720
Market Making	ReAct	.687/.449	.687/.449	.687/.449	.737/.345	.708/.394	**.827**/**.153**	.842
	Debate	.687/.449	.687/.449	.687/.449	.714/.417	.676/.541	**.831**/**.117**	.842
Financial QA	ReAct	.260/.364	.260/.364	.260/.364	.233/.545	.280/.227	**.287**/**.182**	.313
	Debate	.200/.533	.213/.400	.200/.533	.220/.333	**.227**/**.267**	**.227**/**.267**	.253
Retail Dispute	ReAct	.542/.500	.535/.510	.953/.044	.587/.457	–	**.972**/**.011**	.980
	Debate	.424/.665	.424/.669	.953/.044	.553/.496	–	**.972**/**.011**	.980
Customer Support	ReAct	.737/.353	.745/.306	.747/.281	.764/.275	–	**.766**/**.216**	.810
	Debate	.728/.421	.740/.323	.741/.245	.732/.392	–	**.742**/**.206**	.764
Email Response	ReAct	.533/.506	.540/.496	.535/.514	.536/.495	–	**.672**/**.274**	.845
	Debate	.556/.479	.557/.482	.545/.513	.577/.472	–	**.677**/**.269**	.812
Negotiation	ReAct	.661/.647	.692/.518	.749/.304	.641/.700	–	**.794**/**.104**	.812
	Debate	.665/.501	.631/.520	.670/.581	.651/.520	–	**.741**/**.183**	.800

5.2 Direct Candidate Selection

Our first experiment isolates selection quality by holding the decision context and candidate slate fixed across methods. Table 1 retains both role-blind and mandate-conditioned versions of representative frontier judges, allowing the effect of supplying the mandate to be compared within the same model. Appendix Table 6 additionally reports GPT-4o, DeepSeek, GPT-5.6-terra, Claude Sonnet 5, and Math-PRM. These additional baselines do not consistently outperform the frontier judges retained in the main table and do not change the overall method ranking.

PolyPRM achieves the highest score on all eight tasks. This includes four evaluator-free tasks whose outcomes are determined by gold answers, executable rules, observed decisions, or benchmark-native payoffs. PolyPRM also remains the strongest selector on NegoBench, where both the mandates and payoff functions are defined by the source benchmark. Its performance therefore cannot be attributed solely to reproducing an LLM-generated labeling function or to evaluator scale.

The no-mandate control measures dependence on an explicit evaluation criterion rather than role dependence alone. Its largest effect occurs on the single-role Financial QA task, where removing the mandate also removes the task-specific verification standard. We therefore interpret this result as criterion conditioning. Role dependence is tested separately through Mandate Counterfactual Accuracy in Section 6.1, which holds the state and candidate slate fixed while changing the mandate semantics.

5.3 Improving Agent Decisions with PolyPRM

Table 2 evaluates PolyPRM on proposals generated by frozen Qwen3-8B agents under ReAct and Debate. Within each setting every controller selects from the same stored candidate slate, so differences reflect selection quality rather than proposal generation. PolyPRM improves on the Base Agent in all 16 settings and attains the best or tied-best non-oracle outcome in 15; the exception is Portfolio under ReAct, where ThinkPRM reaches a higher average outcome.

Table 3: Mandate-use ablations with checkpoints and candidate slates fixed within each task. Higher is better; bold and underlined values denote the best and second-best results within each column, respectively. A method without access to the mandate attains 0 on MCA by construction; a method that orders each pair independently at random attains 0.25 in expectation.

	Selected Utility						Mandate Counterfactual Accuracy		
Mandate Input	**Trading**	**Portfolio**	**Market Making**	**Retail Dispute**	**Email Response**	**Negotiation**	**Retail Dispute** 1,463 pairs	**Email Response** 1,747 pairs	**Negotiation** 10,590 pairs
Full mandate	<u>0.7783</u>	**0.7840**	**0.8370**	**0.9844**	<u>0.8003</u>	<u>0.8192</u>	**0.9535**	**0.8615**	**0.5300**
Without role name	**0.7830**	<u>0.7830</u>	**0.8370**	<u>0.9690</u>	**0.8010**	**0.8310**	<u>0.9337</u>	<u>0.8152</u>	<u>0.5039</u>
Role name only	0.7780	0.7770	<u>0.8190</u>	0.7870	0.7630	0.6430	0.0820	0.3944	0.0153
No mandate	0.7772	0.7780	0.8180	0.6480	0.6610	0.6560	0.0000	0.0000	0.0020
Wrong mandate	0.7763	0.7790	0.8070	0.4330	0.5490	0.6700	0.0253	0.0939	0.1350
Shuffled mandate	0.7743	0.7771	0.8066	0.3390	0.4956	0.6607	0.0239	0.1099	0.1532

Normalized regret separates selection quality from slate difficulty, since it is computed only over the n_{sel} decision points whose candidates differ in utility. On this metric PolyPRM attains the lowest value in all 16 settings, including Portfolio under ReAct, where its regret of $.434$ against $.504$ shows that it selects closer to the slate optimum wherever the candidates are distinguishable. The largest improvements appear on Customer Support, Retail Dispute, Email Response, and Negotiation, where proposals are individually plausible but differ substantially in how well they serve the active role; on Retail Dispute and Market Making the selections approach the candidate-pool oracle, so the frozen agents already generate high-utility proposals but do not reliably identify them.

Customer Support shows why both metrics are reported: the outcome margins over the strongest baseline are .002 and .001, at the resolution of the reported estimates, while the regret margins are .059 and .039. Every Customer Support state is discriminative ($n = n_{\text{sel}} = 225$), so the gap comes from normalization rather than from equivalent candidates: regret rescales each state by its own utility range. The complementary effect appears on Market Making, Financial QA, and Negotiation, where n_{sel} is a minority of n and absolute outcome is diluted by points at which no controller can change the result. Finance gains are smaller but consistent, and on Portfolio and Market Making the three non-learned controllers coincide because self-consistency and the task heuristic reproduce the base agent's choice, leaving selection quality to the learned evaluators. Overall, PolyPRM improves frozen agents by choosing better processes from an unchanged proposal pool, without retraining the proposal policy or increasing its generation budget.

6 Analysis

6.1 Does the Model Use the Mandate?

Strong selection performance does not by itself show that PolyPRM uses the mandate: the model could learn a role-independent quality function, or treat the role name as an action prior. We separate these with mandate counterfactuals that hold the checkpoint, decision context, and candidate slate fixed and vary only the mandate presented at inference. Without Role Name drops the name but keeps the objective and constraints; Role Name Only keeps the name and drops the body; No Mandate drops both; Wrong Mandate and Shuffled Mandate substitute another role's mandate.

Table 3 separates the name from the body. Removing the role name preserves both selected utility and Mandate Counterfactual Accuracy, which scores a reversal pair correct only when a method reproduces both roles' orderings, while removing or corrupting the body degrades both. The mismatched conditions are the informative ones: on all three tasks they fall below the 0.25 expected from ordering each pair at random, so the model is not simply losing resolution. The same pattern holds on NegoBench, whose mandates and utilities are defined by the source benchmark, so mandate-dependent reversals do not require an LLM-generated outcome judge. Changes are smaller on the finance tasks, whose candidate utilities vary less across roles.

The sub-chance values reflect a reversal rather than noise. On the same pairs we measure how often the model follows the substituted role's ordering instead of the evaluated role's, for which random

ordering gives 50%. On Retail Dispute the model follows the substituted role on 3.2% of reversal pairs when it receives the correct mandate and on 97.5% $[96.8, 98.1]$ when it receives the wrong mandate; the shuffled condition, evaluated on a larger matched-pair set, gives 97.3% $[96.4, 98.1]$ against a 2.3% reference. Email Response shows the same reversal, from 5.8% to 86.4% $[83.2, 89.4]$ under the wrong mandate and from 7.8% to 87.7% $[85.2, 90.0]$ under shuffling. Every substituted interval excludes 50% by a wide margin. On control pairs where the two roles agree the error rate is unchanged (Appendix B.2). PolyPRM therefore uses the decision-relevant content of the mandate rather than the role name: corrupting the mandate does not make it uncertain, it makes it serve the role it was given.

6.2 Joint Training versus Role-Specific Experts

We compare joint training with independently trained role experts. Each expert receives the same number of training examples as the single joint checkpoint. A routed system with r roles therefore uses r training runs and r times as much aggregate training data. PolyPRM instead distributes one fixed training budget across roles and learns a single mandate-conditioned evaluator. Both approaches use the same data splits, candidate slates, backbone, optimization recipe, and per-model training budget. The routed baseline is granted oracle role routing. For Negotiation, both systems are evaluated on the same 10 party roles, while the joint checkpoint supports all 32 roles within the training budget of one expert.

Table 4: Joint training versus routed role-specific experts. Results report macro selected utility (↑). Bold marks the higher point estimate in each row. The differences on Portfolio and Market Making are below 0.005 and should be read as parity rather than improvement.

Task	**Evaluated roles (r)**	**Routed experts** (r models)	**Joint PolyPRM** (1 model)
Trading	3	0.3719	**0.3899**
Portfolio	3	0.5505	**0.5526**
Market Making	2	0.7902	**0.7916**
Email Response	3	0.8565	**0.8677**
Retail Dispute	3	0.9525	**0.9704**
Negotiation	10	0.5324	**0.8557**

Joint PolyPRM matches or exceeds the routed system on all six tasks while using one training run instead of r. Both draw on the same pooled set of labeled examples, so the saving is in training runs and deployed adapters rather than in annotation. The gain is largest on Negotiation ($+0.323$), where the joint checkpoint additionally covers all 32 party roles while the routed system is trained and evaluated on 10, so that difference combines cross-role transfer with broader role coverage. Gains fall between $+0.011$ and $+0.018$ on Email Response, Retail Dispute, and Trading, while the $+0.002$ and $+0.001$ differences on Portfolio and Market Making are within the resolution of these estimates and are best read as parity: parameter sharing preserves role-specific evaluation on those two tasks rather than improving it. Routed experts additionally receive oracle role routing yet require r role-specific adapters, 530 MB for the 10-role Negotiation system, against one 53 MB LoRA adapter on the shared 1.5B backbone with no expert selection. Joint training therefore reduces training and deployment cost without sacrificing role-specific utility.

7 Conclusion

We introduced PolyPRM, a mandate-conditioned process reward model that evaluates candidate processes according to the active decision role. PolyPRM shares process knowledge across roles while preserving their distinct objectives, constraints, and utility orderings. We also developed RIVA-Bench to evaluate this capability under matched decision contexts and candidate slates. Across diverse decision tasks, PolyPRM consistently improves candidate selection and the downstream decisions of frozen agents. Mandate counterfactuals and joint-training analyses further show that these gains arise from using role-specific mandates while sharing supervision across roles. Together, these results establish mandate conditioning as an effective foundation for process evaluation in role-specific agent systems.

AI USE STATEMENT

Large language models are integral to the research system studied in this work. PolyPRM is initialized from an LLM backbone, while other LLMs serve as proposal engines, evaluation baselines, and components of the benchmark-construction pipeline. During data construction, LLMs were used to generate portions of the candidate pools and to provide mandate-conditioned quality judgments when task-native signals did not fully specify process quality. The provenance of these components was retained, and the resulting supervision was reviewed through the validation procedures described in the paper. Generative AI tools were also used to assist with code development and improve the clarity and readability of the manuscript. The authors designed the methods and experiments, verified the data and reported results, reviewed all AI-assisted outputs, and take full responsibility for the final content and conclusions.

ETHICS STATEMENT

Seven of the eight RIVA-Bench task configurations are built from public datasets and environments—FinQA, τ^2-Bench, the Olist e-commerce release, and NegoBench—or constructed by the authors from public data and simulators, and are used under their respective licences. These seven configurations and the construction and evaluation code will be released.

The Email Response configuration is derived from records produced by an email assistant deployed within a single organization, and is used under a written data use agreement with that organization. The records arose in the ordinary course of operating the system, and the analysis reported here serves the same purpose: measuring and improving the quality of the assistant's responses. No data was collected for this paper beyond what the deployment already produced, no human subjects were recruited, and access to the source records is restricted to the team operating the system. We report only aggregate statistics and exclude raw message content, personal identifiers, and any example from which an individual message or participant could be reconstructed; Appendix C.8 represents this configuration only through placeholders. It is not redistributed and will not be publicly released.

Mandate-conditioned evaluation can also be used to optimize an agent for one stakeholder in settings where stakeholders conflict. We regard the explicitness of the mandate as a partial mitigation, since the objective, authority, and constraints under which a candidate is scored are written out and auditable rather than implicit in the evaluator.

REFERENCES

Anthropic. Introducing claude opus 4.8. https://www.anthropic.com/news/claude-opus-4-8, 2026.

Marco Avellaneda and Sasha Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Leo Benac, Jonas Raedler, Zilin Ma, and Finale Doshi-Velez. A benchmark for multi-party negotiation games from real negotiation data. *arXiv preprint arXiv:2603.14066*, 2026.

Bruce J Biddle. Recent developments in role theory. *Annual review of sociology*, 12(1):67–92, 1986.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711, 2021.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui

Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, et al. Agentprocessbench: Diagnosing step-level process quality in tool-using agents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 8823–8834, 2026.

Dazhi Fu, Jiuding Yang, Yiwen Guo, and Jicong Fan. Many voices, one reward: Multi-role rubric generation for llm judging and reward modeling. *arXiv preprint arXiv:2607.01830*, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

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

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Seungone Kim, Jay Shin, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Ryan Shin, Sungdong Kim, James Thorne, Minjoon Seo, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*, volume 2024, pp. 29927–29962, 2024.

Huayi Lai, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Zhouxing Wang, Zhiqiang Yin, and Xun Liang. Rolecde: Benchmarking and mitigating role–alignment trade-offs in role-playing agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2226–2248, 2026.

Dawei Li, Yuguang Yao, Zhen Tan, Huan Liu, and Ruocheng Guo. Toolprmbench: Evaluating and advancing process reward models for tool-using agents, 2026. URL https://arxiv.org/abs/2601.12294.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Olist and André Sionek. Brazilian e-commerce public dataset by olist, 2018. URL `https://www.kaggle.com/dsv/195341`.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. `https://openai.com/index/gpt-5-6/`, 2026.

Silviu Pitis, Ziang Xiao, Nicolas Le Roux, and Alessandro Sordoni. Improving context-aware preference modeling for language models. *Advances in Neural Information Processing Systems*, 37: 70793–70827, 2024.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 60808–60838, 2025.

Jiwoong Sohn, Tomasz Sternal, Kenneth Styppa, Torsten Hoefler, and Michael Moor. Process reward agents for steering knowledge-intensive reasoning. *arXiv preprint arXiv:2604.09482*, 2026.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8642–8655, 2024a.

Noah Wang, Zy Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, et al. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14743–14777, 2024b.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024c.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195, 2026.

Wei Yang, Muyan Weng, Jiacheng Pang, Defu Cao, Heng Ping, Peiyu Zhang, Shixuan Li, Yue Zhao, Qiang Yang, Mengdi Wang, et al. Toward evolutionary intelligence: Llm-based agentic systems with multi-agent reinforcement learning. *Available at SSRN 5819182*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, et al. Versaprm: Multi-domain process reward model via synthetic reasoning data. *arXiv preprint arXiv:2502.06737*, 2025.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning, 2025. URL https://arxiv.org/abs/2501.07301.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, et al. Genprm: Scaling test-time compute of process reward models via generative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34932–34940, 2026.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024a.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, et al. Sotopia: Interactive evaluation for social intelligence in language agents. In *International Conference on Learning Representations*, volume 2024, pp. 40975–41019, 2024b.

Jie Zhu, Yuanchen Zhou, Shuo Jiang, Junhui Li, Lifan Guo, Feng Chen, and Chi Zhang. Fin-prm: A domain-specialized process reward model for financial reasoning in large language models. *arXiv preprint arXiv:2508.15202*, 2025.

A Additional Experimental Details

A.1 Implementation Details

Data partitions. All experiments use the fixed train–validation–test partitions reported in Table 15. Splits are constructed over each task's natural evaluation unit, including asset–day decisions, portfolio states, support sessions, dispute cases, email threads, and player–state pairs. Market Making is grouped at the episode–role level, producing slates of 44 to 50 candidates. The direct and downstream Email Response experiments use the same 714 held-out decision contexts but different candidate pools. Direct selection uses benchmark-constructed response slates, whereas downstream evaluation uses proposals generated independently by the frozen agents. No candidate scores or selections are reused across the two protocols.

PolyPRM architecture and optimization. PolyPRM initializes from Qwen2.5-1.5B-Instruct and trains LoRA parameters together with ordinal-value, utility-regression, and auxiliary prediction heads while keeping the backbone weights frozen. One checkpoint is trained for each task configuration using all supported decision roles. The ordinal head predicts a distribution over 11 uniformly spaced utility levels in $[0, 1]$, whose expectation provides the candidate score.

Training combines within-state pairwise ranking, ordinal classification, utility regression, and auxiliary action or process prediction. All configurations use AdamW, a grouped batch containing one decision state, gradient accumulation of eight, weight decay 0.01, gradient clipping at 1.0, and ranking temperature 0.1. Seven tasks use the shared default configuration of LoRA rank 8, learning rate 10^{-5}, ranking weight 2.0, and ranking margin 0.05. Trading uses the stronger ranking configuration in Table 5, selected using validation pairwise accuracy before test evaluation. No configuration choice uses test selected utility.

Configuration and checkpoint selection use validation data only. Trading and Email Response use within-state pairwise accuracy because model selection in these tasks targets candidate ordering within a slate. The remaining tasks use Spearman correlation over their graded validation utilities. The mandate is rendered before the state, history, and candidate process. Sequences are truncated from the right so that the complete mandate is preserved when an input exceeds the task-specific maximum length. Training and inference use the tokenizer stored with each checkpoint.

Process-evaluation baselines. We evaluate Qwen2.5-Math-PRM-7B, DianJin-Fin-PRM, ThinkPRM-1.5B, and MRRG as transferred process evaluators. Fin-PRM is an in-domain baseline for the finance suite. Its non-finance direct-selection results are treated as transfer evaluations, and it is omitted from non-finance downstream comparisons. ThinkPRM uses prompt-based adaptation, while ThinkPRM-FT applies LoRA to DeepSeek-R1-Distill-Qwen-1.5B using the same task training data. MRRG constructs domain-relevant rubrics from multiple roles but produces a pooled evaluation score. The PolyPRM without Mandate control uses the same trained checkpoint as PolyPRM while withholding the mandate at inference.

LLM judges. The general-purpose judges include GPT-4o, DeepSeek-Chat, GPT-5.6-luna, GPT-5.6-sol, GPT-5.6-terra, Claude Sonnet 5, and Claude Opus 4.8. Each judge independently assigns a score to every candidate at temperature zero. The role-blind condition receives the decision context and candidate process without the mandate. The mandate-conditioned condition additionally receives the complete active mandate.

Direct candidate selection. Each selector receives the same state, history, and frozen candidate slate. Candidate order is shuffled with a fixed seed, so First Candidate represents a fixed random position rather than the original generation order. Held-out candidate utilities are used only after selection and are never exposed to the selector.

For Financial QA, the raw records contain a metadata field, `thesis`, that identifies the perturbation category and can reveal the gold candidate. We exclude this non-admissible metadata from every method. Baseline adapters mask the field explicitly, while PolyPRM's renderer never includes it. All selectors therefore receive the same admissible question, evidence, and candidate content.

Table 5: PolyPRM training configurations across the eight task settings.

Task	LoRA r/α	Maximum length	Epochs	Learning rate	Ranking weight / margin	Checkpoint metric
Trading	16 / 32	1,536	14	2×10^{-5}	8.0 / 0.005	Pairwise
Portfolio	8 / 16	1,536	12	10^{-5}	2.0 / 0.05	Spearman
Market Making	8 / 16	1,536	12	10^{-5}	2.0 / 0.05	Spearman
Financial QA	8 / 16	1,024	12	10^{-5}	2.0 / 0.05	Spearman
Customer Support	8 / 16	2,048	12	10^{-5}	2.0 / 0.05	Spearman
Retail Dispute	8 / 16	2,048	12	10^{-5}	2.0 / 0.05	Spearman
Email Response	8 / 16	2,048	10	10^{-5}	2.0 / 0.05	Pairwise
Negotiation	8 / 16	512	6	10^{-5}	2.0 / 0.05	Spearman

All configuration and checkpoint choices use validation data only. Seven tasks use the shared default optimization configuration, with task-specific sequence lengths and training durations shown above. Trading uses a stronger ranking configuration selected by validation pairwise accuracy before test evaluation. Pairwise accuracy is used for checkpoint selection on Trading and Email Response because their validation objective emphasizes within-slate candidate ordering. The remaining tasks use Spearman correlation over graded candidate utilities. Test selected utility is not used for configuration or checkpoint selection.

Customer Support and Retail Dispute each use 360 evaluation pairs, balanced across three roles with 120 pairs per role. We symmetrize candidate orientation and assign half credit to score ties, making the evaluation invariant to candidate order.

Downstream agent evaluation. A frozen Qwen3-8B proposal model generates the downstream candidate pools using temperature 0.9, top-p 0.95, and at most 320 new tokens. ReAct generates candidate actions directly. Debate first produces complementary analyses and then generates proposals from the resulting discussion. Candidate pools are generated once with a fixed seed and reused by every controller.

Most tasks use six candidates per decision state, while Market Making uses five. Trading, Portfolio, Market Making, Financial QA, Retail Dispute, and Email Response evaluate the selected action through outcome replay against a fixed task-specific action–utility map. Negotiation executes the selected commitment in the NegoBench environment and evaluates the resulting player payoff. The Base Agent selects its original proposal, Self-Consistency selects the most frequently proposed valid action, and the oracle selects the highest-utility candidate from the same frozen slate.

Metrics. Each task retains its native outcome metric. Trading, Portfolio, Market Making, and Negotiation report selected utility. Financial QA reports answer accuracy. Customer Support and Retail Dispute report symmetrized pairwise accuracy. Email Response reports best-of-N utility. Scores are averaged within each decision role and then macro-averaged across roles.

For candidate utilities $u_1, \ldots, u_K$, normalized regret is

$$\mathrm{NRegret} = \frac{u_{\max} - u_{\mathrm{selected}}}{u_{\max} - u_{\min}}.$$

It is computed only for states satisfying $u_{\max} > u_{\min}$, whose count is reported as n_{sel}. Random selection reports the expected utility under a uniform distribution over the slate, while the oracle reports $u_{\max}$.

NegoBench uses its native player-specific payoff

$$R(C) = G^{\top} S(C) - 0.1|C|,$$

where G contains the acting player's goals and $S(C)$ records the commitments in C. Payoffs are normalized within each decision state before selected utility and regret are computed.

Statistical uncertainty. We compute 95% paired cluster-bootstrap confidence intervals using 10,000 resamples. Resampling follows each task's natural evaluation unit, including decision states, cases, threads, and player–state pairs. Competing methods use the same resampled indices, preserving their paired candidate pools, while candidates within a state are never resampled independently.

Table 6: Complete direct candidate-selection results. Bold and underlined values indicate the highest and second-highest results in each column, respectively, using unrounded scores. † marks evaluator-free outcomes, and ‡ marks the privacy-restricted Email Response task.

Method	Trading	Portfolio	Market Making	Financial QA†	Customer Support†	Retail Dispute	Email Response†‡	Negotiation†
Heuristic baselines								
Random	0.642	0.699	0.644	0.153	0.500	0.500	0.621	0.462
First Candidate	0.588	0.619	0.644	0.222	0.494	0.494	0.654	0.471
LLM judges								
GPT-4o	0.767	0.724	0.749	0.849	0.550	0.654	0.657	0.566
GPT-4o + Mandate	0.748	0.725	0.806	0.712	0.557	0.732	0.670	0.718
DeepSeek	0.759	0.746	0.768	0.712	0.560	0.601	0.650	0.545
DeepSeek + Mandate	0.738	0.740	0.772	0.612	0.628	0.765	0.610	0.661
GPT-5.6-luna	0.730	0.749	0.683	0.867	0.686	0.600	0.655	0.474
GPT-5.6-luna + Mandate	0.625	0.750	0.683	<u>0.911</u>	0.663	0.900	<u>0.705</u>	0.766
GPT-5.6-sol	0.668	0.766	0.642	0.805	0.697	0.635	0.638	0.506
GPT-5.6-sol + Mandate	0.634	0.774	0.635	<u>0.911</u>	0.721	<u>0.964</u>	0.687	<u>0.814</u>
GPT-5.6-terra	0.724	0.762	0.658	0.768	0.682	0.646	0.633	0.527
GPT-5.6-terra + Mandate	0.694	0.732	0.721	0.905	0.678	0.899	0.689	0.786
Claude Sonnet 5	0.723	0.774	0.736	0.619	0.725	0.571	0.646	0.515
Claude Sonnet 5 + Mandate	0.733	0.703	0.716	0.712	0.669	0.875	0.696	0.750
Claude Opus 4.8	<u>0.777</u>	0.776	0.641	0.836	0.758	0.647	0.631	0.578
Claude Opus 4.8 + Mandate	0.765	0.744	0.699	0.842	<u>0.774</u>	0.960	0.698	0.778
Existing PRMs								
Fin-PRM	0.730	0.739	0.756	0.526	0.392	0.525	0.644	0.451
Math-PRM	0.713	0.715	0.655	0.706	0.628	0.333	0.617	0.380
ThinkPRM	0.474	0.715	0.637	0.271	0.539	0.475	0.571	0.532
ThinkPRM-FT	0.735	0.726	0.789	0.569	0.561	0.577	0.689	0.468
MRRG	0.749	0.726	0.761	0.526	0.649	0.475	0.640	0.628
Our method								
PolyPRM w/o Mandate	<u>0.777</u>	<u>0.778</u>	<u>0.818</u>	0.674	0.680	0.648	0.661	0.655
PolyPRM	**0.778**	**0.784**	**0.837**	**0.929**	**0.792**	**0.984**	**0.800**	**0.819**

The bootstrap seed is 20260906. These intervals quantify sensitivity to the composition of the evaluated decision units while holding models and candidate pools fixed.

Outcome provenance. Outcome supervision is constructed independently of the evaluated selector. Financial QA uses FinQA gold answers. Customer Support uses executable task and policy rules inherited from the τ^2-Bench environment. Retail Dispute uses structured signals derived from Olist order records together with a mandate-conditioned process-quality component. Email Response uses operational criteria and observed decisions from its privacy-restricted source setting. Negotiation uses the player-specific payoff function supplied by NegoBench. Trading, Portfolio, and Market Making combine task-derived outcomes or simulator returns with a mandate-conditioned process-quality component. No evaluated selector produces its own test outcome.

A.2 Complete Direct-Selection Results

Table 6 reports the complete direct-selection comparison. All judges receive the same candidate pools and are evaluated under the same held-out task outcomes. We include both role-blind and mandate-conditioned versions to distinguish general judging ability from explicit access to the decision criterion. Different judges lead the LLM group on individual tasks, but none of the additional rows exceeds PolyPRM or changes its first-place ranking across the eight tasks.

A.3 Downstream Evaluation: Complete Results

Table 7 is the full-precision version of Table 2. It reports both metrics as separate rows, includes the random-selection baseline omitted from the main text, and lists the per-setting decision-point counts. Task outcome is averaged over all n decision points in a setting. Normalized regret is averaged over the n_{sel} points whose candidate utilities are not constant: at a point where every candidate carries the same utility, all controllers receive the same value, so the point contributes to outcome but carries no information about selection quality and is excluded from the regret average. Random selection reports the expected utility under a uniform draw from the slate.

Table 7: Complete downstream results at full precision, corresponding to Table 2. The agent and candidate slate are fixed within each setting. Higher outcome and lower normalized regret are better. Bold and underlined values denote the best and second-best non-oracle result within each metric row. Fin-PRM is not evaluated on the four non-finance tasks, shown as "–".

Task	Agent	n / n_{sel}	Metric	Random	Base Agent	Self-Consistency	Task Heuristic	ThinkPRM	Fin-PRM	PolyPRM	Oracle
Trading	ReAct	66 / 66	Outcome ↑	0.3449	0.3470	0.3470	**0.4070**	0.3603	0.3441	**0.4070**	0.4509
			NRegret ↓	0.551	0.524	0.524	**0.320**	0.480	0.571	**0.320**	0.000
	Debate	66 / 66	Outcome ↑	0.3447	0.3470	0.3470	**0.4070**	0.3532	0.3411	**0.4070**	0.4509
			NRegret ↓	0.550	0.524	0.524	**0.320**	0.557	0.615	**0.320**	0.000
Portfolio	ReAct	39 / 39	Outcome ↑	0.4674	0.4629	0.4629	0.4629	**0.5020**	0.4260	0.4898	0.7197
			NRegret ↓	0.536	0.510	0.510	0.510	0.504	0.565	**0.434**	0.000
	Debate	39 / 39	Outcome ↑	0.4674	0.4629	0.4629	0.4629	0.4775	0.4756	**0.5008**	0.7197
			NRegret ↓	0.536	0.510	0.510	0.510	0.476	0.537	**0.418**	0.000
Market Making	ReAct	250 / 72	Outcome ↑	0.6921	0.6868	0.6868	0.6868	0.7372	0.7081	**0.8270**	0.8420
			NRegret ↓	0.487	0.449	0.449	0.449	0.345	0.394	**0.153**	0.000
	Debate	250 / 72	Outcome ↑	0.6921	0.6868	0.6868	0.6868	0.7142	0.6758	**0.8309**	0.8420
			NRegret ↓	0.487	0.449	0.449	0.449	0.417	0.541	**0.117**	0.000
Financial QA	ReAct	150 / 22	Outcome ↑	0.2467	0.2600	0.2600	0.2600	0.2333	0.2800	**0.2867**	0.3133
			NRegret ↓	0.455	0.364	0.364	0.364	0.545	0.227	**0.182**	0.000
	Debate	150 / 15	Outcome ↑	0.2133	0.2000	0.2133	0.2000	0.2200	**0.2267**	**0.2267**	0.2533
			NRegret ↓	0.400	0.533	0.400	0.533	0.333	**0.267**	**0.267**	0.000
Retail Dispute	ReAct	225 / 225	Outcome ↑	0.5560	0.5419	0.5353	0.9525	0.5874	–	**0.9718**	0.9798
			NRegret ↓	0.492	0.500	0.510	0.044	0.457	–	**0.011**	0.000
	Debate	225 / 225	Outcome ↑	0.5555	0.4242	0.4239	0.9525	0.5532	–	**0.9722**	0.9798
			NRegret ↓	0.496	0.665	0.669	0.044	0.496	–	**0.011**	0.000
Customer Support	ReAct	225 / 225	Outcome ↑	0.7359	0.7366	0.7450	0.7471	0.7639	–	**0.7660**	0.8101
			NRegret ↓	0.364	0.353	0.306	0.281	0.275	–	**0.216**	0.000
	Debate	225 / 225	Outcome ↑	0.7279	0.7282	0.7403	0.7408	0.7319	–	**0.7420**	0.7639
			NRegret ↓	0.418	0.421	0.323	0.245	0.392	–	**0.206**	0.000
Email Response	ReAct	714 / 506	Outcome ↑	0.5333	0.5330	0.5396	0.5346	0.5360	–	**0.6723**	0.8451
			NRegret ↓	0.502	0.506	0.496	0.514	0.495	–	**0.274**	0.000
	Debate	714 / 451	Outcome ↑	0.5659	0.5557	0.5571	0.5451	0.5772	–	**0.6773**	0.8123
			NRegret ↓	0.475	0.479	0.482	0.513	0.472	–	**0.269**	0.000
Negotiation	ReAct	64 / 21	Outcome ↑	0.6844	0.6609	0.6921	0.7494	0.6408	–	**0.7942**	0.8123
			NRegret ↓	0.539	0.647	0.518	0.304	0.700	–	**0.104**	0.000
	Debate	64 / 25	Outcome ↑	0.6465	0.6652	0.6314	0.6696	0.6510	–	**0.7407**	0.7999
			NRegret ↓	0.509	0.501	0.520	0.581	0.520	–	**0.183**	0.000

The gap between n and n_{sel} varies widely. Retail Dispute and Customer Support are fully discriminative, whereas Market Making retains 72 of 250 points, Financial QA retains 22 of 150 under ReAct and 15 of 150 under Debate, Negotiation retains 21 and 25 of 64, and Email Response retains 506 and 451 of 714. On the settings with small n_{sel}, outcome differences are compressed by a majority of decision points at which no controller can change the result. This is why normalized regret is reported alongside outcome rather than in place of it.

Random selection exceeds the Base Agent in 10 of the 16 settings, most visibly on Retail Dispute under Debate (0.5555 against 0.4242). The frozen agent's own first proposal is therefore close to uninformative as a selection rule on several tasks, which is consistent with the oracle gaps in Table 7: the proposal pools contain high-utility candidates that the generating policy does not rank first. This does not affect the comparisons themselves, since every controller ranks the same stored candidates.

B More Analysis

B.1 Evaluation-Set Uncertainty

Repeated evaluations of frontier LLM judges are costly and can be affected by changes in hosted APIs. We therefore quantify evaluation-set uncertainty for PolyPRM and three representative pro-

Table 8: Direct-selection performance with cluster-bootstrap uncertainty for PolyPRM and three representative process evaluators. Each cell reports the point estimate and bootstrap standard deviation. These deviations quantify evaluation-set uncertainty for fixed predictions rather than variation across training runs.

Task	PolyPRM	Fin-PRM	ThinkPRM	MRRG
Trading	$\mathbf{0.7783 \pm 0.0059}$	0.7303 ± 0.0129	0.4742 ± 0.0268	0.7477 ± 0.0098
Portfolio	$\mathbf{0.7835 \pm 0.0050}$	0.7392 ± 0.0179	0.7146 ± 0.0142	0.7261 ± 0.0118
Market Making	$\mathbf{0.8369 \pm 0.0072}$	0.7563 ± 0.0163	0.6366 ± 0.0193	0.7611 ± 0.0122
Financial QA	$\mathbf{0.9294 \pm 0.0181}$	0.5259 ± 0.0373	0.2714 ± 0.0322	0.5258 ± 0.0384
Customer Support	$\mathbf{0.7917 \pm 0.0185}$	0.3917 ± 0.0311	0.5389 ± 0.0265	0.6486 ± 0.0249
Retail Dispute	$\mathbf{0.9844 \pm 0.0151}$	0.5250 ± 0.0262	0.4750 ± 0.0256	0.4750 ± 0.0250
Email Response	$\mathbf{0.8003 \pm 0.0147}$	0.6445 ± 0.0171	0.5707 ± 0.0133	0.6397 ± 0.0172
Negotiation	$\mathbf{0.8192 \pm 0.0304}$	0.4508 ± 0.0295	0.5321 ± 0.0414	0.6280 ± 0.0312

Table 9: Candidate construction sources in the eight audited direct-selection pools. No Qwen-family generator is identified in their provenance records.

Task	Candidate sources
Trading	GPT-4o, DeepSeek, and Grok outputs with rule-based variants
Portfolio	DeepSeek, Grok, and GPT-4o outputs with rule-based variants
Market Making	Programmatic quoting policies
Financial QA	Gold answers and rule-based perturbations
Customer Support	Claude-3.7-Sonnet, o4-mini, GPT-4.1, and GPT-4.1-mini
Retail Dispute	Rule-based adjudication actions
Email Response	Recorded trajectory variants and constructed perturbations
Negotiation	Negotiation simulator

cess evaluators using their saved per-unit predictions. For each task, we perform 10,000 cluster-bootstrap resamples over its natural evaluation units, including trading days, scenarios, questions, negotiation states, work orders, and orders. Each replicate first averages examples within a decision role and then macro-averages across roles. All methods within a task use the same resampled clusters.

Among the four methods in Table 8, PolyPRM has the highest point estimate on all eight tasks. Relative to the three process evaluators reported in the table, its performance gaps are larger than the corresponding bootstrap dispersion in every task. The comparison is therefore not driven by a small subset of evaluated decision units. Customer Support and Retail Dispute report pairwise accuracy, while the remaining tasks use their task-specific selection outcomes.

The MRRG estimates condition on one stored API realization and do not capture server-side sampling or model-version variation. More generally, the reported deviations measure evaluation-set uncertainty for fixed saved predictions rather than variation across independent training runs.

B.2 Candidate Provenance and Source Sensitivity

We examine how candidate provenance and input presentation affect process evaluation. The analysis combines a provenance audit of the eight direct-selection candidate pools, a controlled crossover between two generators and two reward models, and a test of score stability under alternative input serializations.

All PolyPRM checkpoints in the direct-selection comparison use Qwen2.5-1.5B-Instruct as their backbone. The audited provenance records identify no Qwen-family generator in any of the eight candidate pools. Table 9 summarizes their construction sources. These comparisons therefore evaluate PolyPRM on candidates from other model families or from rule-based and simulated processes, without a same-family candidate generator in the audited pools.

Table 10: Generator–evaluator interactions on matched state–action cells. D_{src} is measured on the $[0, 1]$ predicted-score scale. Confidence intervals are clustered by state.

Task	Role	Matched cells	States	D_{src}	95% CI
Customer Support	Support agent	154	17	-0.0075	$[-0.0130, -0.0025]$
Customer Support	Merchant compliance	159	13	-0.0105	$[-0.0173, -0.0032]$
Retail Dispute	Platform	375	75	$+0.0415$	$[+0.0381, +0.0449]$

To directly examine a setting with family overlap, we generate matched candidates using Qwen3-8B and Llama-3.2-3B and score every candidate with both a Qwen2.5-based PRM and a Llama-3.2-based PRM. Within each matched state–action cell, the generators express the same action with the same gold action utility. Let S_{eg} denote the mean predicted score from evaluator e on candidates from generator g, where Q and L denote the Qwen and Llama configurations. We measure the generator–evaluator interaction as

$$D_{\mathrm{src}} = (S_{QQ} - S_{QL}) - (S_{LQ} - S_{LL}). \tag{5}$$

This contrast removes additive evaluator and generator effects. A positive value indicates that the Qwen evaluator has a stronger relative preference for Qwen-generated candidates than the Llama evaluator does. It measures an interaction in predicted scores, which can reflect both source-related preferences and differences in how evaluators assess the generated explanations.

Table 10 reports results with each checkpoint evaluated using its training-time rendering. Confidence intervals use 4,000 paired bootstrap resamples of states while retaining all matched actions and evaluator–generator combinations within each sample. All three interactions are distinguishable from zero, but their signs differ across settings. The interaction is negative for both Customer Support roles and positive for the Retail Dispute platform role, where $D_{\mathrm{src}} = 0.0415$ with a 95% confidence interval of $[0.0381, 0.0449]$. The results therefore reveal setting-dependent evaluator–generator interactions rather than a uniform preference for same-family candidates. These interactions are measured on predicted scores and may reflect both source sensitivity and differences in how evaluators assess generated explanations.

We further test sensitivity to input presentation by holding the checkpoint and candidate content fixed while applying six surface-form transformations. Across the four evaluated task–role settings, the transformed scores have Pearson correlations of 0.990–1.000 with the original scores, with top-1 selection changes ranging from 0 to 8.3%. Table 11 reports the correlations. These results establish high score agreement under the tested changes to formatting and surface form.

As a separate diagnostic, applying a full canonical rewrite to a checkpoint trained on the original rendering produces correlations of 0.206–0.729 and changes the top-ranked candidate in 23%–97% of cases. Because this rewrite changes the input distribution, we additionally train and evaluate checkpoints using matched canonical renderings. The measured generator–evaluator interaction changes sign for the merchant-compliance and platform roles, confirming that source-sensitivity estimates depend on the input representation. The mandate counterfactual, however, remains intact. On Retail Dispute, the canonically trained Qwen checkpoint obtains an MCA of 0.944 with the full mandate, compared with 0.000 without the mandate and 0.043 with the wrong mandate. Moreover, 95.7% of wrong-mandate errors follow the ordering induced by the substituted mandate. The corresponding Llama checkpoint yields 0.933, 0.000, and 0.085, with 86.9% directional agreement. Canonicalization therefore affects absolute scores and measured source interactions, but does not remove mandate-dependent ranking reversals under matched training and evaluation.

Directional agreement. On a reversal pair the evaluated and substituted roles order the candidates oppositely by construction, so among the pairs on which a non-tying model fails the evaluated role's ordering it necessarily matches the substituted role's. We therefore compute the rate over all reversal pairs rather than over errors, for which the reference level is 50% rather than 100%. Retail Dispute contributes 1,463 reversal pairs over 75 cases under the wrong mandate and Email Response 2,029 pairs over 229 work orders. Under SHUFFLED MANDATE the substitution is drawn independently per candidate, so the statistic is restricted to pairs whose two candidates receive the same substituted role, pooled over three shuffling rounds; this yields a larger set of 2,223 and 3,077 pairs respectively,

Table 11: Pearson correlation between original and transformed candidate scores. The final row applies a full canonical rewrite to an unchanged checkpoint trained on the original rendering.

	Customer Support		Retail Dispute	Email Response
Input transformation	**Support agent**	**Merchant compliance**	**Platform**	**Requester**
JSON key reordering	0.998	0.998	1.000	1.000
Markdown formatting	0.997	0.997	0.990	0.995
Whitespace changes	0.999	0.990	0.998	0.991
Tool-call formatting	0.996	0.995	1.000	1.000
Boilerplate changes	0.999	0.996	0.996	0.995
Number formatting	1.000	0.999	1.000	0.999
Full canonical rewrite	0.710	0.729	0.527	0.206

Table 12: Validation of evidence-gated supervision on 292 Retail Dispute units. The integrity layer reduces silent errors and routes evidence-insufficient cases to review.

(a) Label outcomes

Method	**Correct ↑**	**Silent errors ↓**	**Review**
Original labels	164	128	0
Direct LLM relabeling	129	163	0
Supervision Integrity Layer	**166**	**96**	30

(b) Diagnostic behavior counts

Observed behavior	**Units**
Clean label protected from harmful relabeling	103
Insufficient evidence correctly routed to review	30 / 30
True label error corrected	88
True label error left unchanged	40
Clean label incorrectly modified	28

and the correct-mandate reference is recomputed on that set. On control pairs where the two roles agree on the correct ordering, the error rate under the substituted conditions is 8.8–9.4% on Retail Dispute and 16.1–16.9% on Email Response, confirming that the flip is not a general degradation under perturbed input. Intervals use case- and work-order clustered bootstrap with $B = 5{,}000$. This analysis uses the mandate-ablation checkpoint reported in Table 3.

B.3 Validating the Supervision Integrity Layer

Role-indexed supervision may combine task outcomes, deterministic rules, and model-assisted quality judgments. The *Supervision Integrity Layer* first determines whether a proposed label is supported by the available evidence and consistent with the active mandate. It preserves supported labels, corrects identifiable errors when sufficient evidence is available, and routes uncertain cases to review. Unlike direct LLM relabeling, it does not require a replacement label when the record cannot support one.

Evaluation protocol. We evaluate the procedure on 292 Retail Dispute validation units containing clean labels, natural or injected errors, and evidence-insufficient cases. Deterministic task rules provide the reference whenever the required evidence is available. We compare retaining the original labels, direct relabeling, and applying the Supervision Integrity Layer with the same DeepSeek annotator. A silent error is an incorrect label retained without abstention. A review decision indicates that the layer declined to write a label.

Direct relabeling reduces the number of correct units by 35 relative to retaining the original labels and increases silent errors by the same amount. The Supervision Integrity Layer instead reduces silent errors by 25% and routes all 30 evidence-insufficient units to review. It prevents 103 harmful modifications to clean labels, avoids 30 unsupported replacements, and corrects 88 genuine errors. A replication with GPT-4o-mini produces the same qualitative pattern. Direct relabeling yields 130 correct units and 162 silent errors, while the integrity layer yields 155 correct units, 107 silent errors, and 30 review decisions.

Downstream evaluation clarifies which part of the procedure produces the improvement. On a held-out rule-only reference containing no LLM-generated evaluation labels, the PRM trained with integrity-checked supervision reaches 0.969 selected utility, compared with 0.834 using the original labels and 0.822 after direct relabeling. A filter-only control, which removes labels identified as unreliable without correcting them, reaches 0.976. Its slightly higher point estimate identifies unreliable-label detection and exclusion as the primary source of the downstream gain. The correction branch provides a way to retain examples whose errors can be resolved from available evidence, as demonstrated by the 88 corrected units, although this experiment does not establish an additional downstream advantage over filtering alone.

Preserving the active decision role. Direct relabeling can also weaken role-specific supervision. For the buyer role, the mean label assigned to `support_seller`, the least favorable ruling for that role, increases from 0.143 to 0.396 after direct relabeling. The Supervision Integrity Layer instead assigns a mean value of 0.061. This behavior indicates that direct relabeling can shift supervision from the buyer mandate toward a role-neutral adjudication criterion. The same shift is not observed for the seller role, identifying it as a role-specific failure mode rather than a universal property of relabeling.

Case analysis. Figure 3 illustrates the decisions available to the integrity layer. It can preserve supported supervision, abstain when evidence is missing, or correct an identifiable error. The final case shows the cost of its conservative intervention policy. Avoiding unsupported edits reduces silent corruption but can leave some recoverable errors unchanged.

B.4 Case Studies of Mandate-Conditioned Selection

We use two held-out NegoBench cases to illustrate how mandate conditioning changes process rankings and improves decisions made by a frozen agent. The cases are selected from saved test logs using fixed high-contrast criteria. No candidates, labels, or model scores are regenerated. Aggregate evidence is reported in Tables 1 and 2, while the cases below make the corresponding mechanisms directly inspectable.

Figure 4(a) isolates the effect of the mandate. Japan and Niger evaluate the same Chilean commitment menu, but their native objectives reverse the preferred action. PolyPRM selects the unique utility-maximizing commitment for both parties and places the other party's preferred commitment last. Removing the country name while retaining the mandate body preserves both selections, showing that the reversal follows the stated objectives rather than the role name. In contrast, removing the mandate produces one shared ranking and gives Niger its least-preferred commitment.

Figure 4(b) shows how this mechanism affects a frozen agent. Four of the six proposals favor the only action that visibly advances a goal, causing the base agent, self-consistency, ThinkPRM, and the mandate controls to select an action with zero native utility. PolyPRM instead evaluates that apparent progress against the complete active mandate and selects the oracle action. The gain therefore comes from choosing a better process from the same frozen proposal slate, without modifying the proposal model or increasing its generation budget.

B.5 Loss-Function Ablations on Email Response

The losses in Equation 3 supervise different properties of the learned score. Pairwise ranking directly targets the ordering of candidates within a state, while ordinal supervision additionally anchors predicted utilities to a shared eleven-bin scale, providing direct supervision for comparing scores across states. We evaluate these properties through within-state selection and cross-state selective prediction.

A. Clean label protected 98cc17e5

Evidence. The order was delivered 16 days early, and the seller late-delivery rate is 2.2%. The reference assigns `full_refund`=1 and `support_seller`=0.

Direct LLM	`full_refund`: 1.00 → 0.72 `support_seller`: 0.00 → 0.52
Integrity layer	Leaves both rule-consistent labels unchanged.

B. Insufficient evidence f346ad4e

Evidence. Order status, days late, and seller late-delivery rate are missing. The record cannot support a grounded ruling.

Direct LLM	Writes three replacement labels despite the missing evidence.
Integrity layer	Abstains and routes the unit to review without writing a new label.

C. Natural error corrected 6c295372

Evidence. The order was delivered 10 days early, and the seller late-delivery rate is 6.7%. The reference assigns `escalate`=1.

Direct LLM	Changes an unrelated label and leaves the primary error unresolved.
Integrity layer	Raises `escalate` from 0.64 to 0.97, toward the rule reference.

D. Conservative miss 5f071a85

Evidence. An injected corruption sets `full_refund`=0.21, while the rule reference assigns 1.00.

Direct LLM	`full_refund`: 0.21 → 0.80 `support_seller`: 0.00 → 0.52
Integrity layer	Avoids an unsupported secondary edit but leaves the corruption unchanged.

Figure 3: Representative decisions made by the evidence-gated Supervision Integrity Layer. Each case fixes the available evidence and compares direct LLM relabeling with evidence-gated retention, correction, or abstention. The final case illustrates the tradeoff introduced by conservative intervention.

Protocol. We use Email Response because its Institution and Requester roles provide graded candidate utilities across repeated work orders, supporting both within-state and cross-state evaluation. The binary Chair role is excluded so that every comparison uses the same graded utility structure. The resulting analysis contains 2,808 candidates from 402 states and 227 work orders. Institution contributes 1,362 candidates from 195 states, while Requester contributes 1,446 candidates from 207 states.

For each of two independent training runs, we remove one loss term while holding the data, initialization seed, optimization budget, and checkpoint-selection rule fixed within that run. We report

$$\Delta_{\text{sel}} = U_{\text{Full}} - U_{\text{Ablation}}$$

for top-1 selected utility and utility at 50% coverage. For the risk–coverage metric, lower is better, so we report

$$\Delta_{\text{AURC}} = \text{AURC}_{\text{Ablation}} - \text{AURC}_{\text{Full}}.$$

Positive values therefore favor the full objective for every endpoint.

Confidence intervals use 5,000 paired cluster-bootstrap replicates. The evaluation pipeline resamples all 229 work orders, including two that contain only Chair states and therefore do not contribute to the reported two-role statistic. Full and ablated models use the same resampling draws. Each replicate recomputes the metric separately for Institution and Requester before macro-averaging the two roles. Pooled estimates average the two run-specific deltas within each replicate. Their intervals quantify evaluation-set uncertainty conditional on the two trained checkpoints rather than variation over the population of training seeds.

Within-state selection. Panel (a) of Table 13 evaluates the utility of the highest-scoring candidate in each state. Removing pairwise ranking reduces selected utility by 0.0640 and 0.0825 in the two runs, with both intervals excluding zero. No other component shows a consistent positive contribution across runs. Ordinal supervision has no detectable effect on this endpoint, which depends only on the ordering of candidates within each state and does not evaluate their ordering across states.

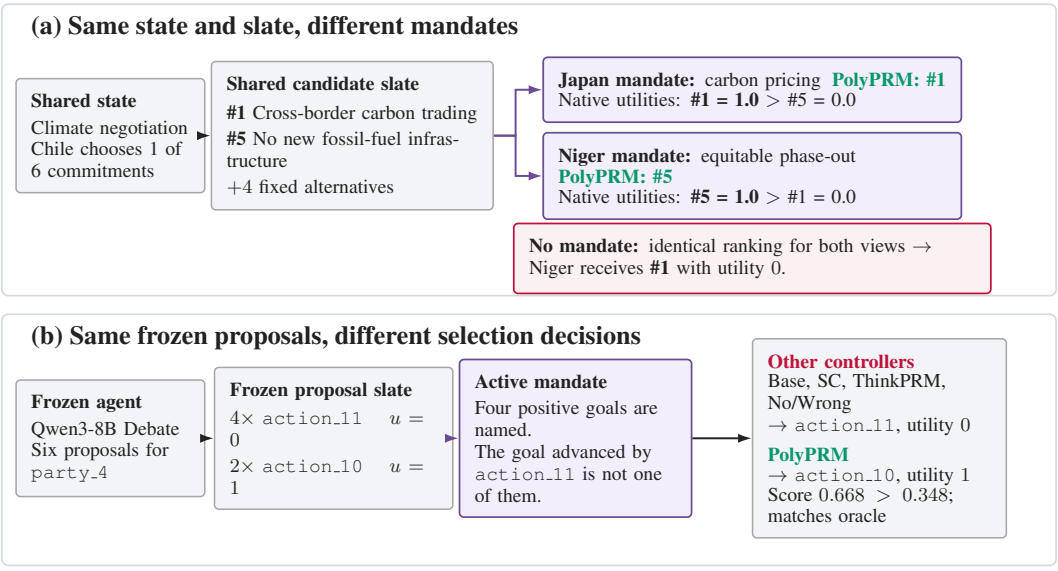

Figure 4: Two held-out NegoBench cases illustrating mandate-conditioned process selection. Panel (a) holds the game state and candidate slate fixed while changing the benchmark-native party mandate. PolyPRM reverses its selection to match the corresponding utility ordering. Panel (b) holds the proposal agent and its six proposals fixed. PolyPRM rejects the proposal majority and selects the utility-maximizing action for the acting party. Values denoted by u are benchmark-native utilities. The values 0.668 and 0.348 are PolyPRM selection scores. SC denotes self-consistency.

Removing action prediction produces a small positive difference in Seed A, but the effect does not replicate in Seed B.

Cross-state selective prediction. For each candidate k in state i, the selection score is its predicted utility

$$\hat{u}_{ik} = \sum_{b=0}^{10} p_{ikb} \frac{b}{10},$$

where p_{ikb} is the probability assigned by the ordinal head to bin b. We apply no within-slate normalization, centering, ranking, or top-two margin. Candidates are sorted separately within each role, after which the two role-level metrics are macro-averaged.

At 50% coverage, Institution retains its 681 highest-scoring candidates and Requester retains its highest-scoring 723 candidates. Within a bootstrap replicate, the retained count is rounded down when the resampled role count is odd. For role r, let $u_{r,(j)}$ denote the observed utility of the candidate with the j-th highest predicted score. We compute

$$U_r@50 = \frac{1}{\lfloor n_r/2 \rfloor} \sum_{j=1}^{\lfloor n_r/2 \rfloor} u_{r,(j)}.$$

The corresponding risk at coverage k/n_r is

$$R_r(k) = \frac{1}{k} \sum_{j=1}^{k} \left(1 - u_{r,(j)}\right),$$

and the discrete area under the risk–coverage curve is

$$\mathrm{AURC}_r = \frac{1}{n_r} \sum_{k=1}^{n_r} R_r(k).$$

Every bootstrap replicate re-sorts the resampled candidates and recomputes both the 50% retained set and the complete risk–coverage curve.

Table 13: Loss-function ablations on Email Response. Positive values favor the full objective. A superscript * denotes a 95% paired cluster-bootstrap interval that excludes zero, regardless of direction. The row labeled "All non-pairwise terms" retains only pairwise ranking. Diagnostics are inactive because Email Response provides no corresponding labels.

(a) Within-state selection: top-1 selected utility

Removed component	Seed A	Seed B
Pairwise ranking	$+0.0640$ $[+0.0346, +0.0951]^*$	$+0.0825$ $[+0.0479, +0.1166]^*$
Ordinal supervision	-0.0137 $[-0.0361, +0.0080]$	$+0.0123$ $[-0.0182, +0.0431]$
Utility regression	-0.0003 $[-0.0043, +0.0029]$	-0.0021 $[-0.0054, +0.0000]$
Action prediction	$+0.0056$ $[+0.0003, +0.0125]^*$	-0.0001 $[-0.0197, +0.0182]$
All non-pairwise terms	-0.0078 $[-0.0271, +0.0104]$	$+0.0118$ $[-0.0178, +0.0405]$
Diagnostics	N/A	N/A

(b) Cross-state selective prediction: utility at 50% coverage

Removed component	Seed A	Seed B	Pooled
Pairwise ranking	$+0.0665$ $[+0.0469, +0.0876]^*$	$+0.0778$ $[+0.0512, +0.1007]^*$	$+0.0721$ $[+0.0540, +0.0887]^*$
Ordinal supervision	$+0.0171$ $[+0.0062, +0.0335]^*$	$+0.0433$ $[+0.0196, +0.0613]^*$	$+0.0302$ $[+0.0182, +0.0420]^*$
Utility regression	-0.0006 $[-0.0051, +0.0040]$	-0.0008 $[-0.0052, +0.0032]$	-0.0007 $[-0.0040, +0.0024]$
Action prediction	-0.0031 $[-0.0074, +0.0041]$	$+0.0039$ $[-0.0094, +0.0119]$	$+0.0004$ $[-0.0061, +0.0055]$
All non-pairwise terms	$+0.0250$ $[+0.0117, +0.0425]^*$	$+0.0511$ $[+0.0276, +0.0661]^*$	$+0.0381$ $[+0.0248, +0.0495]^*$
Diagnostics	N/A	N/A	N/A

(c) Cross-state selective prediction: AURC improvement

Removed component	Seed A	Seed B	Pooled
Pairwise ranking	$+0.0499$ $[+0.0370, +0.0636]^*$	$+0.0549$ $[+0.0355, +0.0746]^*$	$+0.0524$ $[+0.0387, +0.0663]^*$
Ordinal supervision	$+0.0032$ $[-0.0081, +0.0141]$	$+0.0261$ $[+0.0064, +0.0449]^*$	$+0.0147$ $[+0.0040, +0.0252]^*$
Utility regression	-0.0023 $[-0.0041, -0.0004]^*$	-0.0001 $[-0.0015, +0.0014]$	-0.0012 $[-0.0024, +0.0001]$
Action prediction	$+0.0002$ $[-0.0029, +0.0036]$	-0.0028 $[-0.0087, +0.0031]$	-0.0013 $[-0.0050, +0.0026]$
All non-pairwise terms	$+0.0141$ $[+0.0016, +0.0261]^*$	$+0.0270$ $[+0.0105, +0.0429]^*$	$+0.0205$ $[+0.0090, +0.0318]^*$
Diagnostics	N/A	N/A	N/A

Panels (b) and (c) show that pairwise ranking also contributes to cross-state ranking quality. Removing it lowers utility at 50% coverage and increases AURC in both runs. Ordinal supervision provides an additional cross-state ranking benefit. Removing it reduces utility at 50% coverage by 0.0171 and 0.0433, with a pooled difference of 0.0302. Its AURC effect is significant in Seed B and in the pooled estimate, although not in Seed A alone. Removing all non-pairwise terms also degrades both selective-prediction endpoints, supporting a contribution beyond pairwise ranking.

Utility regression and action prediction provide no consistent positive contribution on this task. Removing utility regression slightly improves AURC in Seed A, but the effect is small, does not replicate, and is not detectable in the pooled estimate. Email Response contains no labels for the diagnostic heads, so that ablation receives no training-gradient change and is reported as N/A. The 50% operating point and AURC were specified before computing these ablation endpoints, although an earlier analysis had examined a related cross-state rank-correlation diagnostic. Both endpoints depend only on the ordering of predicted utilities across states and are invariant to strictly monotone transformations of the score. They therefore measure cross-state discrimination rather than numerical agreement between predicted and realized utility.

Post-hoc state-level robustness. The primary selective-prediction analysis treats each candidate as an evaluation unit. We additionally conduct a post-hoc analysis that reduces every state to the candidate selected by the corresponding model and uses that candidate's predicted utility as the state-level selection score. This analysis ranks 402 states and retains the highest-scoring 201 at 50% coverage. It removes the within-slate observations available to the candidate-level analysis while retaining the same work-order clustering structure.

The directional pattern remains similar. Removing pairwise ranking reduces state-level utility at 50% coverage by 0.0595, with a 95% interval of $[0.0237, 0.0895]$, and worsens AURC by 0.0618, with an interval of $[0.0271, 0.0962]$. Removing ordinal supervision yields differences of 0.0325 $[-0.0013, 0.0561]$ and 0.0175 $[-0.0098, 0.0458]$, respectively. These estimates have the

same direction as the candidate-level results but are not distinguishable from zero after each slate is reduced to one selected candidate. Removing all non-pairwise terms remains detectable for utility at 50% coverage, with a pooled difference of 0.0413 $[0.0091, 0.0713]$, but not for AURC.

Overall, pairwise ranking provides the dominant contribution to both within-state selection and candidate-level cross-state ranking. Ordinal supervision provides an additional cross-state ranking benefit that is detectable in the prespecified candidate-level analysis, while the state-level robustness check provides directionally consistent but less precise evidence. The remaining supervised heads show no consistent positive contribution on Email Response. Because the ablation is limited to one privacy-restricted task and two training runs, it identifies the objective components supported in this setting rather than establishing their contribution across all RIVA-Bench tasks.

C RIVA-Bench Construction and Validation Details

RIVA-Bench, short for *Role-Indexed Value Assessment Benchmark*, organizes heterogeneous agent decisions around a common process-evaluation interface. Each example pairs a decision context and candidate Process Slate with an explicit decision mandate and role-indexed utility targets. This representation supports controlled candidate selection without requiring tasks to share an action space, output format, or definition of utility.

RIVA-Bench provides two complementary forms of evaluation. All tasks support selector-matched evaluation, in which every method receives the same state, history, and candidate slate for a given role. Tasks with shared decision substrates additionally support mandate-counterfactual evaluation, in which the context and candidates remain fixed while the active mandate changes. The first setting measures candidate-selection quality under controlled proposal support. The second directly measures whether an evaluator recognizes that the value of the same process may change across decision roles.

C.1 Benchmark Scope and Unified Decision Interface

RIVA-Bench contains eight task configurations spanning financial decision making, customer-facing workflows, and multi-party interaction. Trading, Portfolio, and Market Making are controlled tasks constructed for this study. Financial QA, Customer Support, Retail Dispute, and Negotiation adapt public datasets or environments, while Email Response is derived from a privacy-restricted operational setting. The benchmark standardizes their decision interface without treating their roles, candidate spaces, or utility definitions as originating from a single source.

The tasks cover three forms of role structure. Financial QA retains a conventional single-role setting and tests compatibility with standard process evaluation. Trading, Portfolio, Market Making, Customer Support, Retail Dispute, and Email Response use operational decision roles defined during task adaptation. Negotiation provides benchmark-native player roles whose objectives and utilities are specified by the underlying game. Retail Dispute, Email Response, and Negotiation additionally support matched mandate-counterfactual evaluation.

Every task is mapped to a common representation containing a state s_i, a decision history h_i, a candidate Process Slate

$$C_i = \{\tau_{i1}, \ldots, \tau_{iK}\},$$

one or more decision mandates $\{m_r\}$, and role-indexed utility targets $\{u_{i,r,k}\}$. We represent one benchmark instance as

$$(s_i, h_i, C_i, \{m_r\}, \{u_{i,r,k}\})\,.$$

The common interface specifies what an evaluator observes and predicts while preserving the native states, candidate processes, and outcome semantics of each task.

A decision mandate states the objective associated with a role, the evidence it should prioritize, its authority and constraints, and the outcomes or actions it should avoid. The mandate determines the criteria used to value a candidate process. It does not describe a conversational persona or stylistic identity. For example, the same dispute ruling may be evaluated under buyer, seller, and platform mandates. In Negotiation, each mandate is derived from the corresponding player's benchmark-native objectives and payoff structure.

Table 14: Task structure in RIVA-Bench. The table reports the decision unit, candidate process, decision roles, and utility source used by each task configuration.

Task	Decision Unit	Candidate Process	Decision Roles	Utility Source
Trading	Asset–day	Buy, sell, or hold decision	Trader, retail investor, quant	Forward outcome and process quality
Portfolio	Rebalancing state	Portfolio allocation	Portfolio manager, risk manager, LDI	Risk-adjusted outcome and process quality
Market Making	Market scenario	Quote or execution policy	Market maker, execution	Simulator return and process quality
Financial QA	Question	Numerical reasoning process	Analyst	Gold-answer correctness and process quality
Customer Support	Task turn	Tool-mediated action	Customer, support agent, compliance	Policy and task-goal rules
Retail Dispute	Dispute case	Resolution ruling	Buyer, seller, platform	Structured outcome and process quality
Email Response	Support ticket	Response draft	Requester, institution, chair	Operational rules and observed decision
Negotiation	Game state	Commitment action	32 player roles	Benchmark-native payoff function

C.2 Task Sources and Construction

RIVA-Bench combines internally constructed financial tasks, adaptations of public resources, and one privacy-restricted operational task. We distinguish the upstream data or environment from the roles, candidate slates, and utility targets constructed for evaluation.

Financial tasks. Trading and Portfolio use internally constructed asset–day and rebalancing states, with candidate pools drawn from multiple model families and controlled rule-based variants. Market Making uses an internal limit-order-book simulator and programmatic policies, including an Avellaneda–Stoikov-style strategy Avellaneda & Stoikov (2008). Financial QA adapts FinQA questions and gold reasoning programs Chen et al. (2021), using controlled program perturbations and gold-answer evaluation.

Public-source tasks. Customer Support adapts interaction states and executable task rules from τ^2-Bench Barres et al. (2025), with role-indexed decision views constructed by RIVA-Bench. Retail Dispute constructs buyer, seller, and platform adjudication tasks from structured Olist transaction records Olist & Sionek (2018). Negotiation preserves the player identities, objectives, and payoff functions native to NegoBench Benac et al. (2026) and maps its game states and commitments into the RIVA-Bench interface.

Email Response. Email Response is derived from scenarios observed during the deployment of an email agent in a large-scale organizational setting serving more than 20,000 participants. Its response slates and role-indexed targets reflect operational criteria and decisions encountered in real use rather than simulated email exchanges. Because the source records contain privacy-sensitive information, the task remains under restricted access and will not be publicly released. We report only aggregate statistics and exclude raw messages, personal identifiers, and reconstructable user examples.

C.3 Benchmark Scale

Across its eight tasks, RIVA-Bench contains 111,805 labeled candidate processes grouped into 12,872 decision states. The fixed training, validation, and test partitions contain 8,860, 1,869, and

Table 15: Scale of the eight RIVA-Bench tasks. Split columns report decision states, and K denotes the number of candidates per state.K statistics are pooled over the training, validation, and test splits.

Task	Roles	Train / Valid / Test states	Candidate processes	Slate size K median [range]
Trading	3	345 / 63 / 72	5,399	12 [2, 13]
Portfolio	3	192 / 39 / 39	3,312	13 [11, 13]
Market Making	2	420 / 90 / 90	28,235	47 [44, 50]
Financial QA	1	700 / 150 / 150	5,000	5 [5, 5]
Customer Support	3	566 / 121 / 119	8,299	12 [2, 16]
Retail Dispute	3	1,050 / 225 / 225	7,500	5 [5, 5]
Email Response	3	3,348 / 714 / 714	33,420	7 [6, 7]
Negotiation	32	2,239 / 467 / 734	20,640	6 [6, 6]
Total	**50**	**8,860 / 1,869 / 2,143**	**111,805**	2–50 overall

2,143 states, respectively. Candidate slates range from 2 to 50 processes and cover 50 task-specific role entries. Table 15 reports the distribution of decision states and candidates across tasks.

C.4 Matched Role-Indexed Supervision

RIVA-Bench controls candidate availability separately from role dependence. In selector-matched evaluation, every method receives the same decision substrate

$$D_i = (s_i, h_i, C_i)$$

for a given decision role. Candidate generation is completed before evaluation, and the resulting slate is reused across selectors. Performance differences therefore reflect which candidate a method selects rather than which candidates it generates.

Where the source task supports a shared decision substrate across roles, RIVA-Bench also constructs mandate counterfactuals. The state, history, and candidate slate remain unchanged while the active mandate and role-indexed utility targets vary. For two roles r and r', a candidate pair may satisfy

$$u_{i,r,a} > u_{i,r,b}, \qquad u_{i,r',a} < u_{i,r',b}.$$

We call such a pair a *reversal pair* and write it as the triple $(i, \{a, b\}, \{r, r'\})$. Mandate Counterfactual Accuracy scores a method correct on a reversal pair only if it reproduces *both* orderings: the method is queried twice on the same state and candidate pair, once under m_r and once under $m_{r'}$, and must satisfy $v_{i,r,a} > v_{i,r,b}$ and $v_{i,r',a} < v_{i,r',b}$. Satisfying one ordering and not the other scores zero. Exact ties in the predicted score are counted as incorrect. MCA is the fraction of reversal pairs scored correct, with reversal pairs weighted equally within a task; the pair counts reported in Table 3 are counts of these triples.

Two reference points make the scale interpretable. A scorer that produces a single fixed ordering for a candidate pair, as any method without access to the mandate must, satisfies exactly one of the two conditions and attains 0 by construction. A scorer that orders each candidate pair independently at random under each mandate attains 0.25 in expectation. Values below 0.25 therefore indicate that a method orders reversal pairs systematically in the wrong direction rather than at chance.

A role-independent scorer cannot represent both orderings with one fixed ranking. These reversals therefore provide a direct test of whether an evaluator conditions its judgment on the active mandate.

The two matching protocols answer different questions. Selector matching measures candidate-selection quality under fixed proposal support. Role matching measures whether the preferred process changes appropriately with the decision role. Tasks with role-specific states or action spaces support the first comparison, while tasks with matched cross-role substrates support both.

C.5 Role-Indexed Utility Anchors

For each decision state, role, and candidate process, RIVA-Bench associates a normalized utility

$$u_{i,r,k} = A_r(s_i, h_i, \tau_{ik}) \in [0, 1],$$

Matched retail dispute: same public transaction evidence and same five candidate rulings

Delivered 20 days early · ambiguous fault · low-review and high-freight signals

Buyer — verbatim mandate

[TASK]
marketplace_dispute_adjudication
[ROLE]
buyer
[OBJECTIVE]
Maximize the buyer's recovery and satisfaction.
[FOCUS]
refund/replacement · not stuck with a bad order · low friction
[AVOID]
siding with the seller against a valid buyer claim
[CONSTRAINTS]

Platform — verbatim mandate

[TASK]
marketplace_dispute_adjudication
[ROLE]
platform
[OBJECTIVE]
Evidence-justified, trust-preserving adjudication.
[FOCUS]
match fault attribution to evidence · minimize dispute cost · balanced when ambiguous
[AVOID]
blindly pro-buyer or pro-seller
[CONSTRAINTS]

Seller — verbatim mandate

[TASK]
marketplace_dispute_adjudication
[ROLE]
seller
[OBJECTIVE]
Protect the seller from unwarranted loss.
[FOCUS]
avoid unjust refunds/chargebacks · preserve revenue when not at fault
[AVOID]
refunding at the seller's expense when not at fault
[CONSTRAINTS]

Role-specific utility over the shared candidate slate

Candidate ruling	Buyer	Platform	Seller
full_refund	**1.000**	0.240	0.000
partial_refund	0.640	**1.000**	0.172
replace	0.915	0.519	0.384
support_seller	0.000	0.899	**1.000**
escalate	0.339	0.640	0.921

The evidence and candidate slate are fixed. The active mandate determines which ruling has highest value.

(a) RIVA-Bench Retail Dispute mandates.

Matched negotiation state: same game history and same six candidate commitments

Role identifiers are anonymized; mandate bodies and utilities are unchanged.

Party A — native benchmark mandate

[OBJECTIVE]
Maximize the value of the commitments reached, weighted by your own goal valuations; commitments are binding once made.
[FOCUS]
Ensuring climate finance for developing nations (value +10) · Developing carbon pricing and market-based solutions (value +10) · Supporting national sovereignty in determining fossil fuel phase-out timelines (value +10) · Regulating carbon capture and storage (CCS) technology (value +5)
[AVOID]
Preventing the expansion of fossil gas infrastructure under the guise of transition energy (value -10) · Implementing global moratoriums on new fossil fuel exploration and production (value -10) · Establishing a legally binding Fossil Fuel Non-Proliferation Treaty (value -5)
[CONSTRAINTS]
each commitment you make costs 0.1 payoff · you value 13 of 46 goals positively, 3 negatively

Party B — native benchmark mandate

[OBJECTIVE]
Maximize the value of the commitments reached, weighted by your own goal valuations; commitments are binding once made.
[FOCUS]
Phasing out fossil fuels in a just and equitable manner (value +10) · Ensuring climate finance for developing nations (value +10) · Promoting renewable energy and energy efficiency (value +10) · Implementing just transition policies for workers and communities (value +10)
[AVOID]
Strengthening trade mechanisms to discourage high-carbon imports (value -5)
[CONSTRAINTS]
each commitment you make costs 0.1 payoff · you value 27 of 46 goals positively, 1 negatively

The preferred commitment reverses under the two mandates

Shared candidate commitment	Party A	Party B
Cross-border carbon trading	**1.00**	0.00
No new fossil fuel infrastructure	0.00	**1.00**

The state and candidate slate are fixed. Only the task-defined decision responsibility changes.

(b) Native NegoBench mandates with anonymized role identifiers.

Figure 5: Examples of complete decision-mandate texts and their effects on process value. Within each panel, the decision context and candidate slate are fixed while the active mandate changes. Panel (a) shows mandates constructed by RIVA-Bench, while Panel (b) demonstrates compatibility with mandates defined by an external benchmark. Mandates include only the elements supported by each setting and role rather than requiring all five fields. For example, Retail Dispute does not define a separate constraints field.

where A_r is the utility anchor for decision role r. Collecting these values across roles and candidates produces the Role–Utility Matrix

$$U_i[r, k] = u_{i,r,k}.$$

Each row defines the ordering of the Process Slate for one role and its preferred candidate

$$k^*_{i,r} = \arg\max_k u_{i,r,k}.$$

Differences between rows represent differences in decision objectives rather than annotation disagreement to be averaged away.

Source-prioritized grounding. The source and adaptation of each task are described in Appendix C.2. Utility construction prioritizes the most direct outcome signal available within each task configuration. Utility anchors are derived from gold answers, executable policy checks, simulator returns, observed operational decisions, forward task outcomes, structured outcome rules, or benchmark-native payoff functions. When an outcome signal does not fully characterize process quality, it is combined with an explicit mandate-conditioned quality judgment.

Every candidate process is associated with a task-native utility target. Of the 111,805 candidate records, 49,446, or 44.2%, combine the task-native anchor with an LLM-derived process-quality component. The remaining records rely exclusively on gold answers, executable rules, simulator returns, observed decisions, forward outcomes, or native payoff functions. The source and composition of each target are retained as provenance metadata.

Task-native utility semantics. Normalization places utility targets on a common numerical interval but does not make their meanings interchangeable. Financial QA emphasizes answer correctness and supported reasoning. Customer Support measures progress under task and policy constraints. Market Making incorporates simulated execution outcomes. Negotiation derives value directly from each player's payoff function. RIVA-Bench standardizes the evaluation interface while retaining these task-native definitions of value.

The utility targets used for process ranking are also distinct from the task-level metrics used to report final performance. Financial QA, for example, may use graded process supervision during training while reporting answer accuracy at evaluation. Utility values are therefore interpreted within each task and are not compared across task columns.

C.6 Supervision Validation and Provenance

RIVA-Bench retains the provenance of every candidate and utility target. Each record identifies the candidate source, anchor type, utility components, and validation path. This representation makes deterministic, observed, simulated, and mixed supervision separately auditable.

Rubric-derived components are checked against the evidence available in the state and candidate process. The applied criterion must agree with the active mandate and remain within its authority and constraints. Unsupported or ambiguous examples are excluded or marked as uncertain. Local inconsistencies in scope or evidence linkage may be corrected when the intended target remains identifiable. Such corrections do not introduce a new role preference, override a task-native outcome, or modify the decision mandate.

We also screen for shortcuts that could inflate selection performance without requiring meaningful process evaluation. These checks cover constant-utility slates, label-revealing surface features, high-performing constant policies, duplicated candidates, and role definitions that induce indistinguishable rankings. Candidate rendering is checked across training and evaluation paths so that mandate ablations change only the intended input fields.

Customer Support and Retail Dispute additionally provide 360 role-balanced comparison pairs per task, with 120 pairs for each role. Pairwise evaluation symmetrizes candidate order and assigns half credit to ties.

C.7 Coverage of Decision and Conflict Regimes

RIVA-Bench varies in candidate form, slate size, and the degree to which the active role changes the preferred process. Its candidates include operational actions, portfolio allocations, quote and

Table 16: Prompt and decoding protocols used in the experiments.

Protocol	Input	Generation or scoring	Output
PolyPRM	Plain-text mandate, state, history, and one candidate	Independent deterministic forward pass per candidate	Expected value from the 11-bin ordinal head
LLM judge	One candidate with or without the complete mandate	Temperature 0; three retries after API or parsing failure	Scalar quality or four-level pairwise verdict
ReAct proposals	Role-conditioned system message and task state	Qwen3-8B, temperature 0.9, top-p 0.95	Six stored candidate actions
Debate proposals	Three role-blind analyses followed by a role-conditioned decider	Analyst temperature 0.8; decider temperature 0.9	Six stored candidate actions

execution policies, numerical reasoning processes, dispute rulings, response drafts, and negotiation commitments. This variation prevents success from being reduced to one shared action vocabulary or output format.

The source of role dependence also varies across tasks. Retail Dispute contains direct conflict among buyer, seller, and platform objectives. Customer Support and Email Response represent differences in user progress, institutional responsibility, policy compliance, grounding, and escalation. Financial tasks vary objectives through risk exposure, investment horizon, portfolio constraints, and execution quality. Negotiation provides benchmark-native role dependence because each player's utility follows from its own goals and payoff function.

RIVA-Bench does not require every decision state to produce a role-dependent reversal. High-conflict states test whether an evaluator preserves distinct role-indexed rankings over the same candidates. Low-conflict states provide a control in which different mandates should yield similar decisions. Seven tasks contain no constant-utility test slates. Email Response contains 20.4% constant-utility states. These states remain in outcome evaluation but are excluded from normalized-regret computation because the controller cannot change utility within the slate.

RIVA-Bench defines the decision contexts, candidate slates, mandates, and role-indexed supervision used throughout the paper. We next instantiate this interface with PolyPRM, a shared process evaluator trained to estimate candidate value under the active decision mandate.

C.8 Prompt Templates

We report one canonical template for each independent protocol. Angle brackets denote substituted fields. Privacy-restricted Email content is represented only by placeholders such as <EMAIL_CONTEXT> and <CANDIDATE_REPLY>.

PolyPRM input rendering. PolyPRM is a discriminative scorer and therefore receives a plain-text serialization rather than a conversational prompt. Training and inference use the same rendering function and tokenizer configuration. Each candidate is scored independently, and the candidate with the highest expected ordinal value is selected. Supervision targets and auxiliary labels are stored outside the text.

```
[TASK] <task>
[ROLE] <role>
[OBJECTIVE] <mandate.objective>
[FOCUS] <mandate.focus, joined by " · ">
[AVOID] <mandate.avoid, joined by " · ">
[CONSTRAINTS] <mandate.constraints, joined by " · ">

<STATE BLOCK>
<HISTORY BLOCK>
<CANDIDATE BLOCK>

-> utility_ex_ante, recommended_action
```

Inputs use right truncation with task-specific maximum lengths of 512–2,048 tokens. No evaluated input exceeds its assigned limit. Complete rendered mandates are shown in Figure 5. Mandate ablations modify only the mandate block. The anonymous condition withholds the role name, the

no-mandate condition hides the complete block, and the wrong and shuffled conditions substitute another mandate before rendering.

LLM judges. General-purpose judges score each candidate independently in one system–user exchange. The blind and mandate-conditioned variants receive identical state and candidate content. They differ only in their evaluation instruction and the inclusion of the mandate block.

```
[system]
Trading-step quality judge. JSON only.

[user]
Rate this trading step's quality
(risk-adjusted merit).

<STATE_LINE>
<CANDIDATE_LINE>

Return ONLY JSON {"quality": <0-100>}.
```

```
[system]
Trading decision judge. You are given a ROLE
with an explicit mandate; rate the step by how
well it serves THAT mandate, not by generic merit.
JSON only.

[user]
Rate how well this step serves YOUR ROLE's
objective, given your mandate.

[YOUR ROLE] <role>
[YOUR OBJECTIVE] <mandate.objective>
[YOU CARE ABOUT] <mandate.focus>
[YOU WANT TO AVOID] <mandate.avoid>
[HARD CONSTRAINTS] <mandate.constraints>
<STATE_LINE>
<CANDIDATE_LINE>

Return ONLY JSON {"quality": <0-100>}.
```

Selection judges run at temperature 0 and return a score between 0 and 100. Customer Support and Retail Dispute instead use task-specific pairwise prompts with four ordered verdicts, mapped to 1, 0.66, 0.33, and 0. Score ties receive half credit. Calls are retried up to three times. Persistent parsing failures receive the neutral score of 50 for scalar selection and 0.5 for pairwise evaluation.

Agent proposal generation. Table 2 uses proposals generated once by `Qwen/Qwen3-8B` with its chat template, `enable_thinking=False`, top-p 0.95, top-k 20, and seed 17. Every controller subsequently selects from the same stored pool. Invalid outputs remain in the pool with an error flag, and no candidate is deduplicated.

The active mandate enters through the following system-prefix template.

```
You are the <role>.
Objective: <mandate.objective>.
You care about: <mandate.focus>.
You avoid: <mandate.avoid>.
```

ReAct directly produces a decision at temperature 0.9.

```
[system]
<ROLE_LINE>
You reason briefly, then commit to one decision.
<OUTPUT_CONTRACT>

[user]
<STATE_BRIEF>

Task: <task description>.
Think step by step, then decide.
```

Debate first generates one optimistic, skeptical, and risk-focused analysis without role information at temperature 0.8. A role-conditioned decider then combines the three analyses at temperature 0.9.

Table 17: Input adapters and score readouts for process-evaluation baselines.

Baseline	Input adapter	Mandate access	Selection score
Fin-PRM	State, trajectory, and action rendered as `##Question`, `##Thinking Trajectory`, and `##Final Answer`	Objective and focus for finance roles	Mean step-level and trajectory-level positive-token probability
Math-PRM	Chat query containing the state and proposed action, followed by thesis and rationale steps	None	Mean step-level positive-token probability
ThinkPRM	R1 verification template beginning with `Let's verify step by step:`	Role name and, for Trading, its objective	Normalized probability of `Yes` over `Yes`/`No`
ThinkPRM-FT	Supervised problem–solution prompt using the ThinkPRM backbone	None	Same normalized verification probability as ThinkPRM
MRRG	Per-role rubric construction followed by one judge call per candidate	Mandates pooled during rubric construction	Weighted fraction of satisfied rubric criteria

```
[system]
<ROLE_LINE>
You receive a short debate.
Weigh it AS THIS ROLE and commit.
<OUTPUT_CONTRACT>

[user]
<STATE_BRIEF>

[DEBATE]
[OPTIMISTIC] <analysis>
[SKEPTICAL] <analysis>
[RISK-FOCUSED] <analysis>

Task: <task description>. Decide.
```

Six candidates are obtained either through repeated sampling or action-conditioned calls. Trading and Retail Dispute use one call per action bucket. Financial QA samples six answers and retries non-numeric outputs. Portfolio and Market Making sample legal action labels with a fixed seed and use the model to generate their rationales. Email Response and Negotiation draw six samples from the same prompt.

Process-reward baselines. Table 17 summarizes the input adapters and score readouts used for existing process evaluators. All baselines select the highest-scoring candidate, with the first candidate breaking exact ties.

D PolyPRM Evaluation Harness

We provide the PolyPRM Harness, a lightweight library for reproducing our evaluation protocol and applying mandate-conditioned selection to new agent outputs. The harness loads the frozen benchmark pools using the same input renderers, candidate ordering, metadata sanitization, role-level aggregation, and task-specific metrics used in our experiments. It accepts the released PolyPRM checkpoints, user-provided checkpoints in the same format, or any external scorer implementing a minimal scoring interface. Given a decision state, candidate slate, and natural-language mandate, the harness scores the candidates, returns the selected process, and records an auditable trace. It also supports the mandate ablations used in the paper. Candidate generation, model training, tool execution, and environment interaction remain the responsibility of the evaluated agent.