DOI: 10.15514/ISPRAS-2026-38(6)-37

MAESTRO: Multi-turn Agent Environment Synthesis for Training and Optimization

T.R. Ionov, ORCID: 0009-0002-2708-6353 <trionov@itmo.ru>
V.A. Malykh, ORCID: 0000-0002-4508-2527 <v.malykh@mts.ai>
MWS AI, ITMO University,
49, Kronverkskiy Ave., Saint Petersburg, 197101, Russia.

Abstract. Manual construction of evaluation benchmarks for tool-calling conversational agents does not scale: each domain requires days of expert effort to design data models, implement tools, write policies, and create evaluation tasks. We present MAESTRO, a framework in which code-generating LLM agents produce complete benchmark domains – including executable tools, databases, policies, and evaluation tasks – through a structured pipeline with simulation-based quality filtering. To our knowledge, no prior system produces complete, human-validated evaluation benchmarks for multi-turn policy-following tool-calling agents. The framework evaluates agents using a multiplicative triple metric combining database state verification, tool call matching, and LLM-judged natural language assertions (95.6% judge-expert agreement with a group of 11 annotators). On a held-out test set (3 domains, 90 tasks), group relative policy optimization (GRPO) significantly outperforms supervised fine-tuning and direct preference optimization, and the resulting 9B model surpasses the 27B baseline, demonstrating that targeted RL on domain-specific tasks is more effective than parameter scaling. We release MAESTRO as open-source at the repository.

Keywords: large language models; benchmark generation; tool calling; conversational agents; reinforcement learning; code agents.

For citation: Ionov T.R., Malykh V.A. MAESTRO: Multi-turn Agent Environment Synthesis for Training and Optimization. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 6, part 2, pp. 275-284. DOI: 10.15514/ISPRAS-2026-38(6)-37.

MAESTRO: Синтез многошаговых агентных окружений для обучения и оптимизации

Т.Р. Ионов, ORCID: 0009-0002-2708-6353 <trionov@itmo.ru>
В.А. Малых, ORCID: 0000-0002-4508-2527 <v.malykh@mts.ai>
Центр искусственного интеллекта МВС, университет ИТМО,
Россия, 197101, Санкт-Петербург, Кронверкский проспект, д.49, литер А.

Аннотация. Ручное создание наборов задач для оценки диалоговых агентов с вызовом инструментов не масштабируется: каждая предметная область требует дней экспертной работы для проектирования модели данных, реализации инструментов, написания правил работы агента и создания оценочных задач. В статье представлена система MAESTRO, которая автоматизирует эти этапы с помощью агентов на основе больших языковых моделей, способных создавать программный код. Система формирует базы данных, инструменты, правила и задания, а затем проверяет их посредством моделирования диалогов и исправляет обнаруженные ошибки. Насколько нам известно, ранее не предлагалось систем, объединяющих автоматическое создание всех перечисленных компонентов, экспертную проверку и использование полученных сред для обучения многошаговых диалоговых агентов. Система оценивает агентов по мультипликативной тройной метрике, сочетающей проверку состояния базы данных, сопоставление вызовов инструментов и оценку естественно-языковых утверждений (95,6% согласия судьи и группы из 11 экспертов). На отложенной тестовой выборке (3 предметные области, 90 задач) обучение методом группового сравнения (GRPO) значимо превосходит обучение с учителем (SFT) и оптимизацию по предпочтениям (DPO), а полученная модель 9B превосходит базовую 27B, что демонстрирует большую эффективность обучения с подкреплением на доменно-специфичных задачах по сравнению с масштабированием параметров. MAESTRO опубликован в открытом доступе в репозитории.

Ключевые слова: большие языковые модели; генерация бенчмарков; вызов инструментов; диалоговые агенты; обучение с подкреплением; кодовые агенты.

Для цитирования: Ионов Т.Р., Малых В.А. MAESTRO: Синтез многоходовых агентных окружений для обучения и оптимизации. Труды ИСП РАН, 2026, том 38, вып. 6, часть 2, стр. 275–284 (на английском языке). DOI: 10.15514/ISPRAS-2026-38(6)-37.

1. Introduction

Large language models (LLMs) are increasingly deployed as conversational agents that must follow organizational policies, invoke external tools, and maintain coherent multi-turn dialogues. Evaluating such agents demands benchmarks that test tool-calling correctness, policy compliance, and state management across conversation turns.

Existing benchmarks face three scaling limitations. First, manual scenario creation is expensive: a domain with 15 tasks requires days of expert effort to design data models, tools, policies, databases, gold-standard actions, and NL assertions. Second, manual benchmarks reflect creator biases, limiting scenario diversity. Third, most benchmarks – API-Bank [1], ToolLLM [2], WorkBench [3] – are English-centric, leaving a gap for Russian.

We address these limitations with MAESTRO [4], a framework that automates benchmark generation using code-generating LLM agents. Our contributions:

1. **An automated generation pipeline** in which a code agent produces complete benchmark domains – data models, tools, databases, policies, and evaluation tasks – in approximately 30 minutes, with simulation-based quality filtering.
2. **A triple evaluation metric** combining deterministic state verification (DB), tool call matching (ACTION), and LLM-judged NL assertions with 3x majority voting.

3. **Human validation** showing 95.6% judge-expert agreement on a moderate-complexity domain after targeted fixes.
4. **A training study** showing that GRPO outperforms SFT and DPO (+6.1pp, per task paired CI [+0.025, +0.098]), with the 9B model surpassing the 27B baseline (+3.8pp, per task paired CI [+0.006, +0.071]).

2. Related Work

Tool-calling benchmarks. API-Bank [2] provides 73 APIs with 314 dialogues. ToolLLM [3] scales to 16,000+ APIs but relies on live endpoints; StableToolBench [5] addresses this with virtual servers. WorkBench [4] targets workplace tasks with 690 single-turn scenarios using outcome-centric evaluation. The tau-bench family [6] introduces dual-control simulation where both agent and user are LLM-controlled. DialogTool [7] benchmarks stateful tool use across multi-turn dialogues with fine-grained action-level metrics. T1 [8] provides a tool-oriented dataset emphasizing multi-turn agentic planning and inter-tool dependencies. TRACE [9] proposes multi-faceted rubric-based LLM evaluation for tool-augmented dialogues; our majority-voted assertions trade rubric flexibility for reproducibility.

Synthetic data and environment generation. Self-Instruct [10] and Evol-Instruct [11] generate individual training examples; GAIA [12] generates evaluation questions. These approaches produce data artifacts (instructions, examples, questions) but not executable infrastructure. AppWorld [13] uses LLM-inferred proxy states to accelerate environment development but still requires manual tool implementation. To our knowledge, no prior system produces complete, human-validated evaluation benchmarks – data models, tool implementations, databases, policies, and multi-component evaluation criteria – for multi-turn policy-following tool-calling agents, reusing the same artifact as an RL training environment. MAESTRO fills this gap.

Agent training via RL. DeepSeekMath [14] introduced GRPO, eliminating the value network from PPO. AgentGYM [15] provides diverse environments for agent RL. Recent work [16] demonstrates that strict reward schemas improve tool-use accuracy. Prior work [17] showed that GRPO with deterministic rewards outperforms partial-credit schemes for dialogue state tracking. Our GRPO training extends this to multi-step tool-calling agents.

Positioning. MAESTRO extends tau-bench [6] along four axes: (a) automated domain generation via code agents, (b) a triple evaluation metric with majority-voted NL assertions, (c) Russian language support, and (d) a closed-loop RL training pipeline using the benchmark as a gym environment.

3. MAESTRO Framework

3.1 Domain Architecture

Each benchmark domain consists of six artifacts: (1) a data model as Pydantic schemas with typed fields and cross-references; (2) tool implementations categorized as READ, WRITE, or GENERIC – critically, write tools do not enforce policy constraints, requiring the agent to apply decision rules from the policy text; (3) a database seed with 8–15 records per collection including boundary values and edge cases; (4) an agent policy specifying decision rules with numeric thresholds; (5) evaluation tasks with dialogue schemes, gold-standard tool calls, and NL (natural language) assertions; (6) registration metadata.

The separation between tool capabilities and policy enforcement is a key design choice: tools provide the mechanism (e.g., submit_leave_request), while the policy specifies the conditions (e.g., “at least 14 days advance notice”). This tests whether agents can internalize textual rules rather than relying on tool-level guardrails.

3.2 Domain Generation via Code Agents

A code-generating LLM agent (Claude Code powered with Opus 4.6) produces a complete domain through a structured pipeline (approx. 30 minutes). The key phases:

1. **Planning** – database entities, tool signatures, policy outline, decision boundaries.
2. **Code generation** – Pydantic models, READ/WRITE tools, database seed with boundary values and cross-reference traps.
3. **Policy writing** – numbered rules, numeric thresholds, conditional logic, exception handling.
4. **Task generation** – three parallel subagents with disjoint entity subsets generate tasks across complexity tiers: simple (1-2 actions), standard (3-4), and complex (5-6 with write operations).
5. **Validation** – import test, DB hash round-trip, tool instantiation, task integrity, cross-task analysis.

NL assertions must not duplicate ACTION checks: they test communication quality (whether the agent explained results, applied policy reasoning) rather than restating tool calls.

3.3 Automated Quality Assurance

The quality assurance pipeline operates in three stages (Fig. 1). **Simulation-based filtering**: all tasks are run 10 times with a reference model (Qwen3.5-27B). **Quality gate**: tasks with pass rate 0/10 are flagged for fix-rerun cycles. **Task review agent**: flagged tasks are examined by per-task subagents that execute gold actions against the database, verify NL assertions, and check argument correctness.

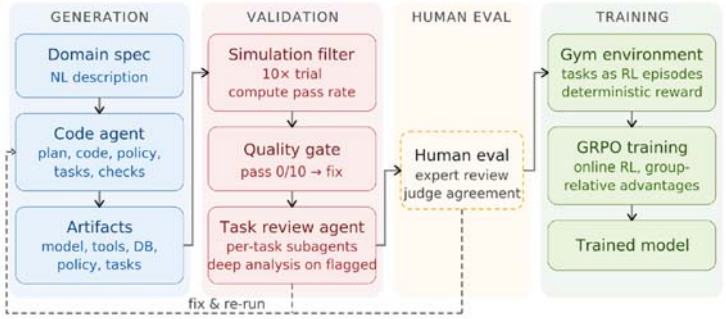


Fig. 1. MAESTRO pipeline: generation via code agents, automated validation with simulation-based filtering, and GRPO training.

3.4 Domain Coverage

The generation pipeline has produced 15+ domains across industries. For the experiments in this paper, we use a held-out test set of 3 domains (Table 1) and 8 training domains (Table 2), all in Russian. Each domain may exist in two interaction modes: **multi-turn (MT)**, where an LLM user simulator follows a structured scenario, and **single-turn (ST)**, where a scripted user provides all information in one message and the agent sequences tool calls for up to 30 steps. Training and test domains are strictly disjoint – each is a separate organizational function with its own tools, database, and policy.

Table 1. Held-out test set (90 tasks). Avg. act = mean gold actions per task.

Domain	MT	ST	R / W	Avg. act	Description
Company analysis	15	15	4 / 1	2.4	Financial data, risk flagging
Risk assessment	15	15	4 / 1	2.5	Business transfer risk, 8 criteria
Personnel management	15	15	5 / 1	2.8	Leave requests, salary, records

Table 2. Training domains. SFT/DPO use all; GRPO uses the 4 with ST tasks (69 tasks).

Domain	MT	ST	R / W	Avg. act	Description
IT support	38	26	7 / 2	2.5	Helpdesk, tickets, equipment
Legal contracts	40	19	5 / 2	2.4	Prolongation, termination
Telecom support	38	14	6 / 1	2.6	Tariffs, balance, packages
Credit assessment	–	10	5 / 2	3.8	Loan applications, scoring
Incident management	15	–	6 / 3	4.0	SLA, escalation, changes
Medical reception	15	–	6 / 2	5.2	Appointments, insurance
Warehouse logistics	15	–	7 / 3	5.6	Stock, orders, suppliers
Budget planning	15	–	7 / 2	5.3	Requests, approval chains

3.5 Human Validation

Two rounds of expert evaluation assessed benchmark quality and evaluation reliability. **V1**: complex domains (company analysis + risk assessment), 30 traces. **V2**: moderate domain (telecom support), 23 traces. 11 trained annotators with cross-verification.

Table 3. Human evaluation across two iterations.

Metric	V1 (n=30)	V2 (n=23)	Delta
Scenario quality (mean)	2.33/5	3.83/5	+1.50
Score=5 (excellent)	20%	47.8%	+27.8pp
Score=1 (broken)	53.3%	17.4%	-35.9pp
Judge-expert agreement	77.1%	95.6%	+18.5pp
Role collapse rate	43%	4.3%	-38.7pp
Judge false positives	9	1	-8
Judge false negatives	18	2	-16

Between iterations, fixes targeted three system components. *User simulator*: stricter prompt with explicit persona boundaries, replacement with Qwen3.5-9B to reduce role collapse – a failure where the user simulator breaks character and attempts to solve its own problem. *NL judge*: replacement with Qwen3.5-9B at lower temperature (0.3) and 3x majority voting; refined assertion formulations. *Domain quality*: guided agent validation – an agent receiving gold actions as hints achieved primary=0.936 on MT tasks, confirming that generated tasks are well-formed and solvable.

The dominant V1 failure was role collapse (43%), a user simulator issue rather than a pipeline flaw. After fixes, V2 failures shifted to distributed minor categories (premature termination, tokenizer artifacts), indicating structurally sound domain generation. V1 and V2 used different domains by design to avoid overfitting; the complexity difference is a confound acknowledged in limitations.

3.6 Evaluation Metric

Each simulation is evaluated on three orthogonal axes, combined multiplicatively:

$$primary = DB \times ACTION \times NL_frac$$

DB (Database State). The entire database is serialized, sorted, and hashed with SHA-256. Free-text fields are excluded from the hash and evaluated via NL assertions. Binary: 1.0 if hashes match, 0.0 otherwise.

ACTION (Tool Call Matching). Gold actions are matched via greedy ordered matching with argument agreement. Supports alternatives for valid alternative tool paths. Binary.

NL_ASSERTION. An LLM judge evaluates each assertion 3 times independently; majority vote determines the verdict. Fractional: NL_frac = met / total.

The multiplicative combination means failure on any axis zeroes the score.

3.7 Simulation and Statistical Methods

Both agent and user are LLM-controlled following the dual-control paradigm [6]. In multi-turn (MT) mode, the user simulator follows a structured scenario with persona, knowledge boundary, and dialogue scheme. In single-turn (ST) mode, a scripted user provides all information in one message; the agent sequences tool calls for up to 30 steps.

Model comparison uses per-task paired deltas with bootstrap resampling (10,000 iterations) for 95% CIs.

The pass^k metric uses strict evaluation (all components = 1.0): pass^k = C(s,k) / C(n,k), where s is the number of successful trials out of n total, measuring the probability that k randomly selected trials all succeed.

4. Experiments

4.1 Setup

Seven models were evaluated on the test set (Table 1; 90 tasks, 5 trials each, Russian): Qwen3.5 family (4B, 9B, 27B, 122B MoE), Qwen3-VL (8B, 32B), and GPT-OSS-120B [18]; reasoning disabled for all Qwen3.5 models including the NL judge. Agent temperature 0.7; NL judge: Qwen3.5-9B at temperature 0.3 with 3x majority voting.

Training base model: Qwen3.5-9B [19]. All methods use LoRA [20] (r=32) on 8×A100-40GB. SFT and DPO use all 8 training domains (Table 2) in both MT and ST modes. GRPO uses the 4 domains with ST tasks (69 tasks) – ST mode is 5-10 times faster than MT (no user simulator API calls).

4.2 Training Methods

SFT. 530 passing traces from Qwen3.5-27B [19], deduplicated to ≤3 per task. TRL SFTTrainer, 1 epoch, LoRA r=32 / α=32, learning rate 2e-5, max sequence length 8192.

DPO [21]. 561 preference pairs (27B chosen vs 9B rejected), assistant-masked loss. Each pair shares the same prompt; chosen and rejected are full dialogues. Qwen3.5's 248K vocabulary required a memory-efficient trainer with chunked LM head computation (16 times memory reduction) and length-normalized log-probabilities. We sweep β in {1, 3, 5} with RPO α=0.05, LoRA r=32 / α=64, learning rate 5e-6, 3 epochs, DeepSpeed ZeRO-2.

GRPO. We adapt verl [22] to work with custom tool-calling gym environments. The reward is fully deterministic (no LLM judge), with weights summing to 1.0: format (0.20), action match (0.30), parameter correctness (0.25), DB state (0.25). Weights prioritize tool-calling correctness (action + param = 0.55) over format compliance, following prior work [17]. Training on 69 ST tasks, 8 tasks/step × 4 rollouts = 32 episodes/step, 25 steps (∼3 epochs). Learning rate 5e-6, KL coefficient 0.001, LoRA r=32 / α=64.

4.3 Model Evaluation

Primary scores (Table 4) increase from 4B (0.669) to 122B MoE (0.833) within the Qwen3.5 family, though paired CIs between 9B, 27B, and 122B all contain zero. Qwen3-VL underperforms Qwen3.5 at comparable scales (VL-8B: 0.592 vs 3.5-9B: 0.794). Most models perform better on MT than ST – interactive clarification provides implicit error correction – with the 4B model showing the largest gap (MT 0.817 vs ST 0.520).

Table 4. Model leaderboard (90 tasks, 5 trials, Russian).

Model	Params	ALL	MT	ST	Delta	95% CI
Qwen3.5-122B-A10B	122B MoE	0.833	0.844	0.822	+0.039	[-0.017, +0.095]
Qwen3.5-27B	27B	0.817	0.855	0.779	+0.023	[-0.023, +0.069]
GPT-OSS-120B	120B MoE	0.802	0.817	0.787	+0.025	[-0.054, +0.104]
Qwen3.5-9B (baseline)	**9B**	**0.794**	**0.861**	**0.727**	–	–
Qwen3-VL-32B	32B	0.729	0.724	0.733	-0.066	[-0.149, +0.017]
Qwen3.5-4B	4B	0.669	0.817	0.520	-0.126	[-0.188, -0.063]
Qwen3-VL-8B	8B	0.592	0.564	0.620	-0.202	[-0.302, -0.102]

4.4 Training Results

SFT achieves ALL = 0.819 (+0.024, CI [-0.022, +0.071]), improving structural correctness (DB×ACTION) by 2.4pp but unevenly across domains. DPO reaches ALL = 0.827 (+0.033, CI [-0.012, +0.078]) but produces a more cautious agent: write tool frequency drops ∼20%, improving read-heavy tasks but hurting DB correctness on write-required tasks.

GRPO training dynamics (Fig. 2) reveal that the base model already knows *which* tools to call (Action ≈ 0.91) but fails to complete write operations: DB starts at 0.528 and improves to 0.972 – the key learning signal. Format compliance remains at 1.000 throughout. GRPO achieves ALL = 0.856 (+0.061, CI [+0.025, +0.098]) – the only method with CI excluding zero.

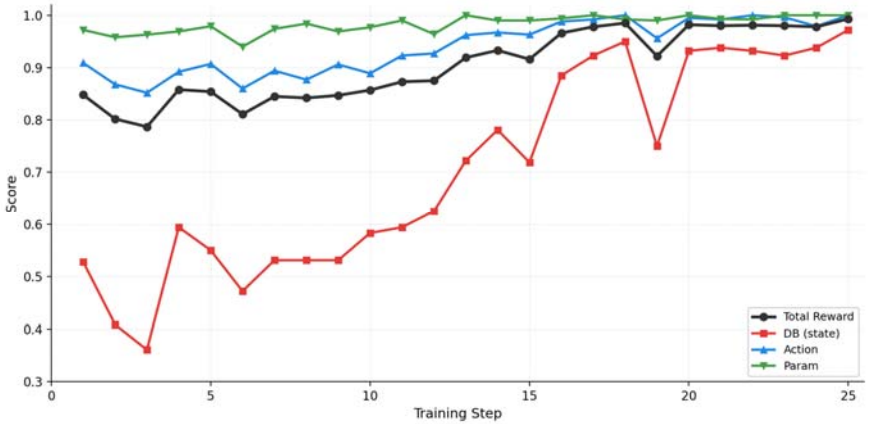


Fig. 2. GRPO training dynamics (25 steps). DB is the primary learning axis: 0.53 → 0.97.

Direct paired bootstrap (Table 5) confirms that the GRPO-trained 9B model significantly outperforms both the 9B baseline (+6.1pp, CI [+0.025, +0.098]) and the 27B model (+3.8pp, CI [+0.006, +0.071]). The 27B itself does not significantly differ from 9B (+2.3pp, CI [-0.023, +0.069]), confirming that GRPO training is more effective than scaling. GRPO trained exclusively on ST tasks yet improved MT performance by 2.9pp (0.861 to 0.890). While the MT-only CI

contains zero at the 45-task level, the consistent direction suggests partial cross-mode transfer of tool-calling skills.

The pass^k curves (Fig. 3) show that GRPO maintains the highest consistency across k=1-5, with the gap widening at higher k.

Table 5. Training method comparison on held-out test set (90 tasks, 5 trials).

Method	ALL	MT	ST	Delta	95% CI
Baseline (9B)	0.794	0.861	0.727	–	–
SFT	0.819	0.867	0.770	+0.024	[-0.022, +0.071]
DPO	0.827	0.907	0.748	+0.033	[-0.012, +0.078]
GRPO	**0.856**	**0.890**	**0.821**	**+0.061**	**[+0.025, +0.098]**
Qwen3.5-27B (ref)	*0.817*	*0.855*	*0.779*	*+0.023*	–
Qwen3.5-122B MoE (ref)	*0.833*	*0.844*	*0.822*	*+0.039*	–

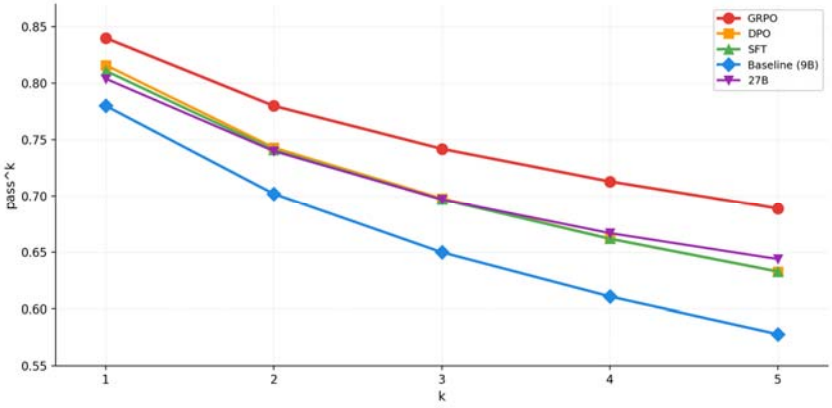


Fig. 3. pass^k consistency (strict: DB × ACT × NL = 1.0). Higher k demands all k trials succeed.

5. Discussion

Why GRPO outperforms SFT and DPO. Tool-calling agents must execute stateful workflows where correct actions depend on intermediate results. SFT imitates fixed demonstrations and cannot discover alternative valid tool sequences. DPO learns from 561 offline pairs with limited coverage of the action space. GRPO explores online, discovering strategies through trial and error – consistent with evidence that online RL outperforms offline methods for agentic tasks [15, 14].

Code agents as benchmark creators. The quality assurance loop – generate, validate, simulate, fix – is the key enabler: without simulation-based filtering, ∼15% of tasks contain errors detectable only through execution. The pipeline has produced 15+ domains with 400+ tasks; equivalent manual construction would require months.

Efficient training versus scaling. GRPO on 9B significantly outperforms 27B (+3.8pp, CI excludes zero) while requiring 3 times less inference compute. For deployment-constrained settings, domain-specific GRPO is more cost-effective than parameter scaling.

Limitations. (1) The benchmark is primarily Russian; cross-lingual evaluation is planned. (2) V1/V2 human validation used different domains, confounding fixes with complexity. (3) At 90 tasks, moderate effects (2–4pp) may not reach significance per mode, though aggregate comparisons have sufficient power. (4) The NL judge belongs to the same model family as evaluated agents; 3x voting mitigates but cannot eliminate potential bias (human evaluation confirms 95.6% agreement).

6. Conclusion

We presented MAESTRO, a framework for automated generation and evaluation of benchmarks for tool-calling LLM agents. Code-generating agents produce complete benchmark domains with simulation-based quality filtering; human evaluation confirms 95.6% judge-expert agreement. The triple metric (DB × ACTION × NL_ASSERTION) captures state correctness, tool accuracy, and communication quality.

Training experiments demonstrate a clear hierarchy: GRPO significantly outperforms SFT and DPO, with gains transferring from single-turn to multi-turn settings. The GRPO-trained 9B model outperforms the 27B model, showing that targeted RL is more effective than parameter scaling for policy-following agent tasks. We release MAESTRO as open-source with a subset of domains; remaining domains are held out to prevent contamination at repository [1].

References

[1]. Li M., Zhao Y., Yu B., Song F., Li H., Yu H., Li Z., Huang F., Li Y. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023. pp. 3102-3116. DOI: 10.18653/v1/2023.emnlp-main.187.

[2]. Qin Y., Liang S., Ye Y., Zhu K., Yan L., Lu Y., Lin Y., Cong X., Tang X., Qian B., Zhao S., Hong L., Tian R., Xie R., Zhou J., Gerstein M., Li D., Liu Z., Sun M. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. International Conference on Learning Representations (ICLR 2024). 2024. Available at: https://openreview.net/forum?id=dHng2O0Jjr, accessed: 10 Apr 2026.

[3]. Styles O., Miller S., Cerda-Mardini P., Guha T., Sanchez V., Vidgen B. WorkBench: a Benchmark Dataset for Agents in a Realistic Workplace Setting. Conference on Language Modeling (COLM 2024). 2024. arXiv preprint. DOI: 10.48550/arXiv.2405.00823.

[4]. Ionov T.R., Malykh V.A. MAESTRO: Multi-turn Agent Environment Synthesis for Training and Optimization. GitHub repository, 2026. Available at: https://github.com/sir-timio/MAESTRO, accessed: 12.05.2026.

[5]. Guo Z., Cheng S., Wang H., Liang S., Qin Y., Li P., Liu Z., Sun M., Liu Y. StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models. Findings of the Association for Computational Linguistics: ACL 2024. 2024. pp. 11143-11156. DOI: 10.18653/v1/2024.findings-acl.664.

[6]. Yao S., Shinn N., Razavi P., Narasimhan K. tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. arXiv preprint, 2024. DOI: 10.48550/arXiv.2406.12045.

[7]. Wang H., Huang W., Wang Y., Xi Y., Lu J., Zhang H., Hu N., Liu Z., Pan J.Z., Wong K.-F. Rethinking Stateful Tool Use in Multi-Turn Dialogues: Benchmarks and Challenges. Findings of the Association for Computational Linguistics: ACL 2025, 2025. pp. 5433-5453. DOI: 10.18653/v1/2025.findings-acl.284.

[8]. Chakraborty A., Dashore P., Bathaee N., Jain A., Das A., Zhang S.-X., Sahu S., Naphade M., Winata G.I. T1: A Tool-Oriented Conversational Dataset for Multi-Turn Agentic Planning. arXiv preprint, 2025. DOI: 10.48550/arXiv.2505.16986.

[9]. Hou Z.J., Shourya T., Wang Y., Roy S., Bannihatti Kumar V., Gangadharaiah R. Multi-Faceted Evaluation of Tool-Augmented Dialogue Systems. arXiv preprint, 2025. DOI: 10.48550/arXiv.2510.19186.

[10]. Wang Y., Kordi Y., Mishra S., Liu A., Smith N.A., Khashabi D., Hajishirzi H. Self-Instruct: Aligning Language Models with Self-Generated Instructions. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, 2023, pp. 13484-13508. DOI: 10.18653/v1/2023.acl-long.754.

[11]. Xu C., Sun Q., Zheng K., Geng X., Zhao P., Feng J., Tao C., Lin Q., Jiang D. WizardLM: Empowering Large Pre-Trained Language Models to Follow Complex Instructions. International Conference on Learning Representations (ICLR 2024), 2024. Available at: https://openreview.net/forum?id=CfXh93NDgH, accessed: 10 Apr 2026.

[12]. Mialon G., Fourrier C., Swift C., Wolf T., LeCun Y., Scialom T. GAIA: a benchmark for General AI Assistants. International Conference on Learning Representations (ICLR 2024). 2024. Available at: https://openreview.net/forum?id=fibxvahvs3, accessed: 10 Apr 2026.

[13]. Trivedi H., Khot T., Hartmann M., Manku R., Dong V., Li E., Gupta S., Sabharwal A., Balasubramanian N. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024), 2024, pp. 16022-16076. DOI: 10.18653/v1/2024.acl-long.850.

[14]. Shao Z., Wang P., Zhu Q., Xu R., Song J., Bi X., Zhang H., Zhang M., Li Y.K., Wu Y., Guo D. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv preprint, 2024. DOI: 10.48550/arXiv.2402.03300.

[15]. Xi Z., Ding Y., Chen W., Hong B., Guo H., Wang J., Yang D., Liao C., Guo X., He W., Gao S., Chen L., Zheng R., Zou Y., Gui T., Zhang Q., Qiu X., Huang X., Wu Z., Jiang Y.-G. AgentGym: Evolving Large Language Model-based Agents across Diverse Environments. arXiv preprint, 2024. DOI: 10.48550/arXiv.2406.04151.

[16]. Ye J., Jiang C., Du Z., Xu Y., Yao X., Xi Z., Fan X., Zhang Q., Gui T., Huang X., Chen J. Feedback-Driven Tool-Use Improvements in Large Language Models via Automated Build Environments. arXiv preprint, 2025. DOI: 10.48550/arXiv.2508.08791.

[17]. Ionov T., Marshalova A., Malykh V. Call, Reward, Repeat: Advancing Dialog State Tracking with GRPO and Function Calling. Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop), 2026. pp. 292-303. DOI: 10.18653/v1/2026.eacl-srw.21.

[18]. OpenAI. gpt-oss-120b & gpt-oss-20b Model Card. arXiv preprint, 2025. DOI: 10.48550/arXiv.2508.10925.

[19]. Yang A., et al. (Qwen Team). Qwen3 Technical Report. arXiv preprint, 2025. DOI: 10.48550/arXiv.2505.09388.

[20]. Hu E.J., Shen Y., Wallis P., Allen-Zhu Z., Li Y., Wang S., Wang L., Chen W. LoRA: Low-Rank Adaptation of Large Language Models. International Conference on Learning Representations (ICLR 2022), 2022. Available at: https://openreview.net/forum?id=nZeVKeeFYf9, accessed: 10 Apr 2026.

[21]. Rafailov R., Sharma A., Mitchell E., Manning C.D., Ermon S., Finn C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Advances in Neural Information Processing Systems 36 (NeurIPS 2023), 2023, pp. 53728-53741. DOI: 10.5555/3666122.3668460.

[22]. Sheng G., Zhang C., Ye Z., Wu X., Zhang W., Zhang R., Peng Y., Lin H., Wu C. HybridFlow: A Flexible and Efficient RLHF Framework. arXiv preprint, 2024. DOI: 10.48550/arXiv.2409.19256.

Информация об авторах / Information about authors

Тимур Русланович ИОНОВ – аспирант института прикладных компьютерных наук Университета ИТМО и научный сотрудник института AIRI.

Timur Ruslanovich IONOV is a postgraduate student at the Institute of Applied Computer Science at ITMO University and a research scientist at AIRI.

Валентин Андреевич МАЛЫХ, кандидат технических наук, руководитель Центра разработки больших языковых моделей MWS AI. Сфера научных интересов: обработка естественного языка, большие языковые модели.

Valentin Andreevich MALYKH – Cand. Sci. (Tech.), Head of the Large Language Model Development Center at MWS AI. Research interests: natural language processing, large language models.