Towards Design of an Automated Judge for Multi-Agent Systems

Alina Zhidkovskaya[1*], **Kirill Rapatskikh**[1], **Jerzy Kamiński**[1], **Grigorii Barakhsin**[1], **Anna V, Kalyuzhnaya**[1†], **Alexey Druzhinin**[2], **Andrey Savchenko**[2], **Julia Belikova**[2], **Konstantin Polev**[2], **Nikolay O. Nikitin**[1]

[1]AI Institute, ITMO University, Saint Petersburg, Russia
[2]Sber AI Lab, Moscow, Russia

Abstract

Evaluating multi-agent systems (MAS) requires nuanced analysis across different levels of interaction, from individual agent behaviors to emergent collective dynamics. Existing methods relying on human annotation or single LLM evaluators struggle to capture this complexity. We introduce a novel unsupervised framework that addresses this gap through a collective of specialized LLM-based evaluators. Our architecture employs a comprehensive two-tier evaluation approach. Specialized evaluators assess individual competencies at the agent level, including observation alignment, tool selection, and state consistency. At the system level, dedicated evaluators analyze collective performance across task completion, role distribution, and complexity metrics. Each evaluator receives structured trace representations with explicit task context. The proposed AutoPumpkin (Automated Performance Understanding for Multi-agents Planning, Knowldge Integration and Negotiation) judge aggregates key agent- and system-level metrics alongside evaluator justifications to determine overall MAS success (binary classification) without requiring labeled training data. On the out-of-distribution TRAIL benchmark, AutoPumpkin achieves F1 0.85, substantially outperforming the TRAIL baseline (F1 0.71). On in-distribution data, AutoPumpkin maintains competitive performance (F1 0.65) compared to TRAIL (F1 0.634), demonstrating strong cross-dataset generalization. This hierarchical evaluation framework enables precise failure identification and targeted optimization across both agent and system levels while remaining scalable and interpretable.

Introduction

The deployment of multi-agent systems (MAS) for complex problem-solving has accelerated rapidly, yet their evaluation remains challenging. Unlike single-agent systems and standalone LLM (Large Language Model) applications, where performance can be assessed through input-output validation, MAS exhibit emergent behaviors arising from inter-agent coordination, role specialization, and dynamic task decomposition. Current benchmarks predominantly use end-to-end metrics that compare outputs against ground truth, supplemented by traditional NLP metrics, such as ROUGE (Lin 2004), BLEU (Papineni et al. 2002), and perplexity (Jelinek et al. 1977). These approaches cannot diagnose whether failures stem from individual agent incompetence, coordination breakdowns, or architectural misalignment.

Recent trace analysis methods address this gap. TRAIL (Deshpande et al. 2025) introduces error taxonomies across reasoning, planning, and execution categories, achieving 11% combined accuracy on localization tasks. MAST (Cemri et al. 2025) identifies 14 failure modes through analysis of 200+ traces. While both frameworks excel at identifying what went wrong at intermediate steps, including coordination failures, MAST does not provide an overall success metric. This leaves critical questions unanswered: Did the system ultimately succeed? How efficiently do agents coordinate? Is task decomposition optimal?

Furthermore, evaluation methods for single-agent or human-AI interaction scenarios inadequately address MAS-specific phenomena. PIPA's (Kim et al. 2025) POMDP-based framework evaluates agent performance along multiple axes. Still, it focuses on individual agent behaviors in task planning contexts, lacking mechanisms to assess inter-agent coordination patterns, role distribution efficiency, or emergent system-level properties that distinguish multi-agent architectures from their single-agent counterparts.

We introduce a comprehensive trace analysis evaluation framework **AutoPumpkin (Automated Performance Understanding for Multi-agents Planning, Knowldge Integration and Negotiation)** that addresses these gaps through a collective of specialized LLM-based judges operating without ground truth requirements. Our architecture implements a two-tiered assessment: *agent-level evaluators* analyze individual competencies, including state consistency, tool selection accuracy, and observation alignment across execution traces, while *system-level evaluators* assess collective dynamics, including task transfer efficiency, role distribution optimality, planning coherence, and conflict resolution mechanisms. Critically, AutoPumpkin evaluates overall system success in addition to intermediate behavioral quality, providing a holistic assessment that determines whether the MAS accomplished its objectives alongside fine-grained diagnostic insights into how effectively it performed. Each specialized judge examines targeted trace segments using

*alina.zhdk@gmail.com
†anna.kalyuzhnaya@itmo.ru

task-agnostic criteria, enabling comprehensive diagnosis of both individual capabilities and emergent coordination patterns. Experimental validation demonstrates significant improvements over existing methods in terms of evaluation accuracy and diagnostic precision, providing actionable insights for optimizing the design of multi-agent systems.

Related Works

Multi-Agent System Evaluation

The evaluation of multi-agent systems has traditionally relied on task completion metrics that assess outcomes against ground truth labels. AgentBench (Liu et al. 2023) introduces diverse metrics, including success rate and F1 scores across multiple domains, while recent frameworks like τ-Bench (Yao et al. 2024) and TravelPlanner (Xie et al. 2024) evaluate agents through task-specific success criteria. However, these approaches provide limited insight into the quality of intermediate decision-making processes, coordination mechanisms, or the root causes of failures, which is the crucial information for systematic improvement of MAS architectures.

Trace-Based Evaluation and Observability

Recent work has emphasized the importance of execution trace analysis for understanding agent behavior. TRAIL (Deshpande et al. 2025) introduces a formal taxonomy of agentic errors and provides a benchmark of 148 human-annotated traces collected using OpenTelemetry instrumentation. Their approach focuses on localizing errors within trace spans across three categories: reasoning, planning, and system execution. TRAIL demonstrates that modern LLMs perform poorly at trace debugging, with the best model achieving a local accuracy of 0.546 and a category F1 score of 0.389. Observability platforms (Langfuse, Arize Phoenix, TruLens) support trace visualization. T-Eval (Chen et al. 2023) introduces step-by-step evaluation of tool utilization capability in LLMs, providing reference-free metrics for assessing retrieval and generation components. We extend these atomic evaluation templates to multi-agent coordination metrics while emphasizing comprehensive quality assessment beyond error localization.

Failure Taxonomies and Classification

MAST (Multi-Agent System Failure Taxonomy) provides the first empirically grounded taxonomy of MAS failures, identifying 14 distinct failure modes across three categories: specification issues, inter-agent misalignment, and task verification (Cemri et al. 2025). Developed through Grounded Theory analysis of over 200 conversation traces, MAST achieves a high inter-annotator agreement F1 score (0.8 with GPT4-o) and introduces an LLM-as-judge pipeline for automated failure detection. MAST's classification approach focuses on identifying **whether specific failure modes occur**, providing valuable diagnostic information for understanding why systems fail. However, as a binary classification framework, it does not assess the **quality or efficiency** of successful behaviors, such as coordination effectiveness, role distribution optimality, or planning coherence—dimensions essential for optimizing well-functioning systems and most importantly, overall system success.

Single-Agent and Interactive Planning Evaluation

PIPA proposes a unified evaluation protocol for interactive planning agents grounded in the Partially Observable Markov Decision Process (POMDP) framework (Kim et al. 2025). The framework evaluates agents along multiple axes, including state consistency, tool efficiency, observation alignment, and task completion, providing a fine-grained diagnosis of decision-making pipelines. Recent work on task-oriented dialogue evaluation, such as TD-EVAL (Xiao et al. 2024), adopts a multi-level assessment that combines turn-level precision with dialogue-level comparisons to capture both localized errors and overall interaction quality. While PIPA and related single-agent evaluation protocols offer comprehensive assessment for task planning scenarios, they do not address MAS-specific phenomena such as inter-agent coordination patterns, task delegation efficiency, conflict resolution mechanisms, or emergent system-level properties that distinguish multi-agent architectures from their single-agent counterparts.

Our work synthesizes these research directions by providing a comprehensive quality assessment framework that analyzes execution traces through specialized evaluators, capturing both individual agent competencies and system-level coordination dynamics without requiring ground truth labels. By combining agent-level and system-level evaluation perspectives with atomic, decomposed criteria, we address the current gap where existing methods either lack holistic success metrics (MAST) or fail to capture MAS-specific coordination phenomena (PIPA, single-agent evaluation frameworks).

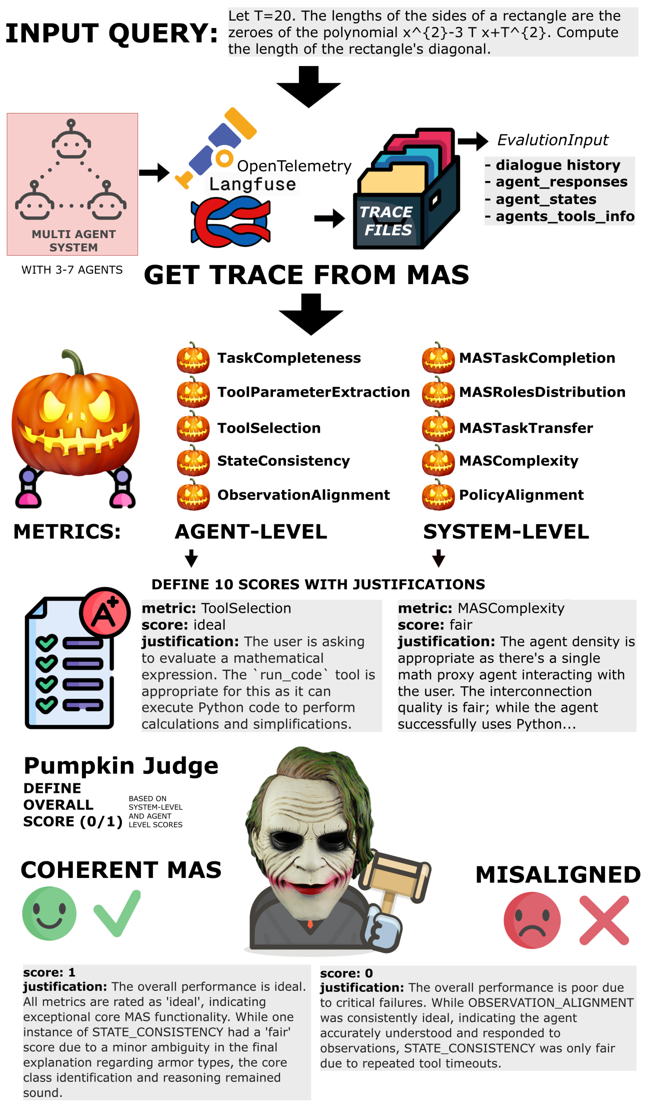
Proposed approach

We propose a hierarchical LLM-as-a-judge framework, AutoPumpkin, for evaluating MAS. The approach decomposes the evaluation problem into system-level and agent-level dimensions, with scores aggregated through the AutoPumpkin judge (Figure 1). This decomposition enables fine-grained analysis of MAS performance across orthogonal evaluation criteria.

System-level metrics

System-level metrics assess the multi-agent system as a unified entity. We define five metrics capturing distinct aspects of MAS behavior:

- **SystemTaskCompletionMetric**: Whether the system successfully achieves the primary task objective. Measures task outcome validity independent of intermediate execution steps.
- **MASRolesDistributionMetric**: Role specialization and allocation across agents. Evaluates alignment between agent capabilities and assigned roles, with consistency across execution.
- **MASTaskTransferMetric**: Effectiveness of inter-agent task handoff and information propagation. Measures con-



Let T=20. The lengths of the sides of a rectangle are the zeroes of the polynomial x^{2}-3 T x+T^{2}. Compute the length of the rectangle's diagonal.

metric: ToolSelection
score: ideal
justification: The user is asking to evaluate a mathematical expression. The `run_code` tool is appropriate for this as it can execute Python code to perform calculations and simplifications.

metric: MASComplexity
score: fair
justification: The agent density is appropriate as there's a single math proxy agent interacting with the user. The interconnection quality is fair; while the agent successfully uses Python...

score: 1
justification: The overall performance is ideal. All metrics are rated as 'ideal', indicating exceptional core MAS functionality. While one instance of STATE_CONSISTENCY had a 'fair' score due to a minor ambiguity in the final explanation regarding armor types, the core class identification and reasoning remained sound.

score: 0
justification: The overall performance is poor due to critical failures. While OBSERVATION_ALIGNMENT was consistently ideal, indicating the agent accurately understood and responded to observations, STATE_CONSISTENCY was only fair due to repeated tool timeouts.

Figure 1: Overview of AutoPumpkin pipeline.

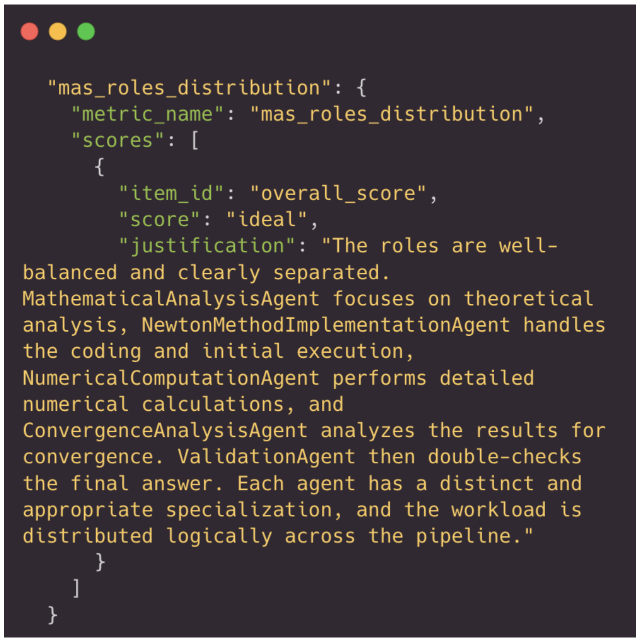
text preservation and communication fidelity during transitions.

- **MASComplexityMetric**: Capacity to handle complex reasoning tasks. Evaluates management of multi-step decomposition and hierarchical problem-solving.
- **PolicyAlignmentMetric**: Conformance to implicit and explicit behavioral policies. Measures adherence to system constraints and safety guidelines.

An example of assessment based on one of these criteria can be found at (Figure 2).

Agent-level metrics

Agent-level metrics examine the individual behavior and local decision-making quality of each agent. We define five metrics operating on per-agent trace segments:



```
  "mas_roles_distribution": {
    "metric_name": "mas_roles_distribution",
    "scores": [
      {
        "item_id": "overall_score",
        "score": "ideal",
        "justification": "The roles are well-
balanced and clearly separated.
MathematicalAnalysisAgent focuses on theoretical
analysis, NewtonMethodImplementationAgent handles
the coding and initial execution,
NumericalComputationAgent performs detailed
numerical calculations, and
ConvergenceAnalysisAgent analyzes the results for
convergence. ValidationAgent then double-checks
the final answer. Each agent has a distinct and
appropriate specialization, and the workload is
distributed logically across the pipeline."
      }
    ]
  }
```

Figure 2: Example of evaluating by MASRolesDistribution metric.

- **TaskCompletenessMetric**: Evaluates whether each agent fully addresses its assigned subtask. Measures task coverage at the individual agent level.
- **ToolParameterExtractionMetric**: Assesses accuracy and completeness of parameter extraction from observations. Evaluates whether agents correctly parse tool outputs and environment state.
- **ToolSelectionMetric**: Evaluates appropriateness of tool selection given current context. Measures whether agents invoke semantically correct tools for their current objectives.
- **StateConsistencyMetric**: Assesses consistency between agent observations and inferred system state. Detects contradictions and inconsistencies in agent reasoning.
- **ObservationAlignmentMetric**: Evaluates whether agent interpretations of observations align with actual tool outputs and semantics. Measures hallucination and misinterpretation rates.

Non-LLM metrics

Non-LLM metrics examine quantitative aspects of multi-agent systems that do not directly measure output quality but significantly impact system performance and cost. We define five such metrics:

- **ToolEfficiencyMetric**: Ratio of successful tool calls to total tool call attempts.
- **AgentTokensMetric**: Total number of tokens consumed per agent (input and output separately and combined).
- **SystemTokensMetric**: Total number of tokens consumed across the entire multi-agent system.

- **AgentTimeMetric**: Wall-clock execution time per agent in seconds.
- **SystemTimeMetric**: End-to-end wall-clock execution time for the entire system in seconds.

AutoPumpkin judge

The AutoPumpkin Judge serves as a meta-evaluator that synthesizes scores from all 10 metrics into a unified system quality assessment.

The Judge produces a binary quality label $\ell \in 0, 1$ indicating whether the trace exhibits sufficient overall system quality. This label serves as the ground truth for metric validation via correlation analysis with individual metric scores.

Implementation

All metric judges and the AutoPumpkin Judge are instantiated using LLM inference. Metric implementations are agnostic to the specific LLM backbone; we employ an instruction-tuned language model for all evaluations. Each judge is queried with a structured prompt containing the evaluation criterion definition and the relevant evaluation input.

We define a standardized trace format that captures multi-agent execution. The `EvaluationInput` schema comprises:

- **Dialogue and execution data**: User queries, dialogue history, agent responses, and agent states (inference outputs, tool invocations).
- **System context**: Agent toolsets, system policies, and agent pool definitions necessary for semantic evaluation.
- **Metadata**: Agent performance metrics (token usage, latency), error logs, trace identifiers, and session information.

This format unifies diverse MAS logging formats into a common evaluation interface. Metric computation operates on subsets of this schema relevant to each metric: system-level metrics process dialogue history and agent responses; agent-level metrics extract per-agent state sequences.

Users may either implement custom parsers to populate `EvaluationInput` from their logging infrastructure, or submit raw execution logs directly.

Experimental studies

Dataset creation

We construct the AutoPumpkin evaluation dataset by collecting execution traces from multi-agent systems solving tasks from the GAIA (Mialon et al. 2023) benchmark validation split. Dataset construction follows a two-stage pipeline: (1) trace acquisition via OpenTelemetry instrumentation, and (2) manual ground-truth annotation.

Trace collection Execution traces are obtained from multi-agent systems instantiated via MetaAgent, an agent that generates task-specific MAS configurations de novo for each question. The dataset comprises 328 traces partitioned into two groups with distinct architectural constraints:

- Group 1 ($n = 163$): Systems with agent counts in range $[3, 7]$
- Group 2 ($n = 165$): Systems with agent counts in range $[3, 4]$

Each trace is captured via OpenTelemetry instrumentation, yielding structured records of agent interactions, intermediate computational steps, and final outputs.

Ground-Truth Annotation Human annotators manually reviewed the output of each multi-agent system for every trace in the collection. During this review process, annotators verified whether the final answer generated by the system was logically correct in relation to the reference answers provided by the GAIA benchmark. For each trace, annotators assigned a binary ground-truth label: 1 if the answer was logically sound and matched the expected solution, and 0 otherwise. This manual verification process establishes a reliable ground truth that reflects the actual correctness of multi-agent system outputs.

Evaluation Framework The constructed dataset enables quantitative evaluation of the proposed metrics. We hypothesize that effective metrics should exhibit a strong correlation with the presence of correct answers in multi-agent system traces. In other words, a well-calibrated metric should assign higher scores to traces where the system produced correct answers and lower scores to traces containing incorrect answers. By measuring the correlation between each of the 11 proposed metrics and the binary ground-truth labels, we can identify which metrics most reliably capture dimensions of multi-agent system performance. Subsequently, these metric scores are then aggregated using a separate LLM Judge, which produces a final score.

Correlation Analysis of Judge-Metrics on Proposed Dataset

We evaluate the predictive accuracy of the judge for each metric by measuring the correlation between metric scores and ground-truth trace correctness labels. For each metric, we calculate the F1 score, which measures the metric's ability to distinguish between correct and incorrect multi-agent executions. Higher F1 scores indicate stronger correlation with trace quality.

AutoPumpkin dataset: Large multi-agent systems (group 1) Evaluation on traces with 3–7 agent configurations (163 traces) is shown in Table 1.

AutoPumpkin dataset: Compact multi-agent systems (group 2) Evaluation on traces with 3–4 agent configurations (165 traces) is presented in Table 2.

Key findings System-level metrics consistently outperform agent-level metrics in predicting trace correctness. On group 1 (large MAS), system metrics achieve F1 scores of 0.64–0.73, while agent-level metrics range from 0.45–0.52. This trend persists in group 2 (small MAS), where system metrics achieve F1 scores of 0.57–0.64 against agent-level F1 scores of 0.51–0.60.

Importantly, metric performance remains comparable across both datasets despite substantial differences in system

Metric	Level	F1
SystemTaskCompletion	system	0.65
MASRolesDistribution	system	0.64
MASTaskTransfer	system	0.64
MASComplexity	system	0.73
PolicyAlignment	system	—
TaskCompleteness	agent	0.52
ToolParameterExtraction	agent	0.47
ToolSelection	agent	0.45
StateConsistency	agent	0.46
ObservationAlignment	agent	0.46

Table 1: Metric F1 scores on large MAS traces (AutoPumpkin dataset gr. 1). System-level metrics demonstrate stronger correlation with trace correctness (F1 0.64–0.73) compared to agent-level metrics (F1 0.45–0.52).

Metric	Level	F1
SystemTaskCompletion	system	0.64
MASRolesDistribution	system	0.57
MASTaskTransfer	system	0.58
MASComplexity	system	0.59
PolicyAlignment	system	—
TaskCompleteness	agent	0.60
ToolParameterExtraction	agent	0.54
ToolSelection	agent	0.54
StateConsistency	agent	0.54
ObservationAlignment	agent	0.51

Table 2: Metric F1 scores on compact MAS traces (AutoPumpkin dataset group 2). System-level metrics achieve F1 0.57–0.64, with agent-level metrics showing F1 0.51–0.60.

configuration (large vs. compact MAS). The relative ranking and magnitude of metric scores are consistent across different MAS architectures, indicating that our metrics are not overfit to specific system designs. This generalization capability suggests that the proposed metrics can reliably evaluate multi-agent systems across diverse configurations and complexity profiles.

Experiments with AutoPumpkin Judge

For the final evaluation of multi-agent systems, a AutoPumpkin judge was introduced. This judge takes both Agent-level and System-level metrics as input and produces an overall quality score. Initially, all computed metrics were used as input; however, this approach did not guarantee optimal AutoPumpkin performance. To address this, a series of experiments was conducted with different metric combinations. These combinations were selected manually, since brute-forcing all 2,036 possible combinations ($2^n – n – 1$ for $n = 11$, assuming at least two metrics per combination) would be computationally expensive. Table 3 presents the results for the evaluated combinations. The study first focused on System-level metric combinations, followed by extending the best-performing System-level sets with Agent-level metrics. The resulting optimal combination included **SystemTaskCompletion**, **MASComplexity**, and **ToolSelection** metrics.

Additional experiments investigated the impact of using different LLMs as evaluators, including the AutoPumpkin judge. The models compared were Gemini 2.5 Flash, DeepSeek-R1, and GPT-5. The results, summarized in Table 4, show that DeepSeek-R1 performed the worst across multiple metrics and produced errors more frequently than the other models. While GPT-5 achieved the highest overall quality score, Gemini 2.5 Flash offered the best balance between evaluation cost and performance. Therefore, all subsequent experiments were conducted using Gemini 2.5 Flash as the primary evaluation model.

During experimentation, a challenge also emerged with the summarization prompt. For GAIA-based evaluations, the final score needed to be converted to a binary outcome, whereas the initial prompt was designed for a three-level scale: poor, fair, and ideal. This introduced ambiguity, as the fair rating did not consistently correspond to either satisfactory or unsatisfactory system performance. To resolve this, two variations of the three-level prompt were tested: (1) considering only poor as negative, and (2) treating both poor and fair as negative. Furthermore, a binary prompt version limited to poor and ideal scores was implemented. The binary prompt improved the overall judge score stability and alignment with human judgments.

Comparative Evaluation Against Existing Judges

Baseline selection A direct comparison with MAST is not feasible due to issues with data quality. MAST reports 1242 human-annotated traces, but only 19 are manually labeled; the remaining 1181 were annotated via LLM, introducing significant noise. We therefore compare against TRAIL, a benchmark with reliable ground-truth labels.

Cross-dataset evaluation We evaluate both judges on a subset of the TRAIL dataset (111 traces). TRAIL judges output continuous scores (0–5), which we convert to binary via thresholds for fair comparison with binary AutoPumpkin judge.

Threshold	TRAIL Judge F1	AutoPumpkin Judge (ours) F1
< 2.5	0.71	**0.85**
< 3.0	0.579	**0.75**
< 4.0	**0.58**	0.43

Table 6: Evaluation on TRAIL dataset. Our AutoPumpkin judge outperforms TRAIL baseline at decision boundaries < 2.5 and < 3.0.

As shown in Table 6, AutoPumpkin achieves F1 0.85 at threshold < 2.5, substantially exceeding the TRAIL baseline (F1 0.71). This demonstrates that hierarchical metric decomposition generalizes effectively to out-of-distribution reasoning traces without threshold tuning.

In-distribution evaluation We evaluate both judges on AutoPumpkin dataset using the threshold that maximizes

Table 3: Experimental results for different metric combinations used to evaluate multi-agent systems: Accuracy and F1 values for each combination.

SYSTEM-LEVEL METRICS		
Combination	*Accuracy*	*F1*
SystemTaskCompletion, MASTaskTransfer	0.6524	0.6275
MASRolesDistribution, MASComplexity	0.6768	**0.5546**
MASComplexity, PolicyAlignment	0.6707	0.6087
SystemTaskCompletion, MASComplexity	**0.6829**	0.6061
SystemTaskCompletion, MASRolesDistribution	0.6646	0.5985
SystemTaskCompletion, MASRolesDistribution, MASTaskTransfer	0.6646	0.5985
MASRolesDistribution, PolicyAlignment	0.6402	0.5816
MASRolesDistribution, MASTaskTransfer	0.6524	0.5778
SystemTaskCompletion, MASRolesDistribution, MASTaskTransfer, MASComplexity	0.6463	0.5672
MASTaskTransfer, PolicyAlignment	**0.5811**	0.5974
SystemTaskCompletion, PolicyAlignment	0.6280	**0.6303**
MASTaskTransfer, MASComplexity	0.6646	0.5802
SYSTEM-LEVEL + AGENT-LEVEL METRICS		
SystemTaskCompletion, MASComplexity, TaskCompleteness	0.6402	0.6093
SystemTaskCompletion, MASComplexity, ToolSelection	0.6829	**0.6389**
SystemTaskCompletion, MASComplexity, ToolParameterExtraction	**0.6933**	0.6377
SystemTaskCompletion, MASComplexity, StateConsistency	0.6585	0.6056
SystemTaskCompletion, MASComplexity, ObservationAlignment	0.6463	0.5915

Table 4: Comparison table of Gemini 2.5 Flash, DeepSeek R1, and GPT-5 on AutoPumpkin dataset using binary and non-binary prompts.

	Gemini 2.5 Flash			**DeepSeek R1**			**GPT-5**		
Dataset	Non-binary prompt		Binary prompt	Non-binary prompt		Binary prompt	Non-binary prompt		Binary prompt
	poor, fair	*poor*	*poor*	*poor, fair*	*poor*	*poor*	*poor, fair*	*poor*	*poor*
AutoPumpkin (gr. 1)	0.72	0.629	0.657	0.68	0.636	0.601	0.115	0.833	0.742
AutoPumpkin (gr. 2)	0.639	0.636	**0.643**	0.583	0.653	**0.658**	0.461	0.783	0.689

Table 5: Cost-effectiveness comparison of Gemini 2.5 Flash and GPT-5 across different dataset sizes using a non-binary prompt. F1/cost ratio measures evaluation quality per unit cost (higher is better).

	Gemini 2.5 Flash		**GPT-5**	
	Dataset gr. 1 *(large MAS)*	Dataset gr. 2 *(small MAS)*	Dataset gr. 1 *(large MAS)*	Dataset gr. 2 *(small MAS)*
F1 score	0.629	0.636	**0.742**	**0.783**
Evaluation cost ($)	0.129	0.127	0.965	0.945
F1/cost ratio	**4.88**	**4.99**	0.77	0.83

TRAIL judge performance on its own benchmark (threshold is less than 2.5). This ensures fair cross-judge comparison.

Dataset	**TRAIL Judge F1**	**AutoPumpkin Judge (ours) F1**
AutoPumpkin	0.634	**0.65**

Table 7: In-distribution performance on our MAS traces at threshold < 2.5. Comparable F1 scores despite different evaluation paradigms.

As presented in Table 7, on the AutoPumpkin dataset, the TRAIL judge achieves F1 0.634, while our AutoPumpkin judge achieves F1 0.65. The comparable performance despite different evaluation paradigms indicates that metric-based decomposition is competitive with continuous scoring approaches on in-distribution data.

Conclusions

We introduced AutoPumpkin, a hierarchical LLM-based framework for evaluating multi-agent systems through explicit metric decomposition. AutoPumpkin combines ten interpretable metrics—five system-level and five agent-level—aggregated by the AutoPumpkin judge to produce binary quality assessments.

Comparative evaluation demonstrates that AutoPumpkin outperforms existing approaches. On the out-of-distribution TRAIL benchmark, AutoPumpkin achieves an F1 score of 0.85, compared to the specialized TRAIL baseline (F1 score of 0.71), while maintaining competitive performance on in-domain data (F1 score of 0.65). This strong cross-dataset generalization indicates that hierarchical metric decomposition captures transferable quality dimensions beyond domain-specific optimization.

The framework provides a foundation for interpretable, automated MAS evaluation. Future work includes expanding the pool of LLM-based evaluation metrics and constructing a large-scale benchmark dataset for judge evaluation.

Acknowledgments

This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-15-2025-010.

References

Cemri, M.; Pan, M. Z.; Yang, S.; Agrawal, L. A.; Chopra, B.; Tiwari, R.; Keutzer, K.; Parameswaran, A.; Klein, D.; Ramchandran, K.; et al. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

Chen, Z.; Du, W.; Zhang, W.; Liu, K.; Liu, J.; Zheng, M.; Zhuo, J.; Zhang, S.; Lin, D.; Chen, K.; et al. 2023. T-eval: Evaluating the tool utilization capability of large language models step by step. *arXiv preprint arXiv:2312.14033*.

Deshpande, D.; Gangal, V.; Mehta, H.; Krishnan, J.; Kannappan, A.; and Qian, R. 2025. TRAIL: Trace Reasoning and Agentic Issue Localization. *arXiv preprint arXiv:2505.08638*.

Jelinek, F.; Mercer, R. L.; Bahl, L. R.; and Baker, J. K. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1): S63–S63.

Kim, T.; Singh, J.; Mehri, S.; Acikgoz, E. C.; Mukherjee, S.; Bozdag, N. B.; Shashidhar, S.; Tur, G.; and Hakkani-Tür, D. 2025. PIPA: A Unified Evaluation Protocol for Diagnosing Interactive Planning Agents. *arXiv preprint arXiv:2505.01592*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Mialon, G.; Fourrier, C.; Wolf, T.; LeCun, Y.; and Scialom, T. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

Xiao, R.; Ma, W.; Wang, K.; Wu, Y.; Zhao, J.; Wang, H.; Huang, F.; and Li, Y. 2024. Flowbench: Revisiting and benchmarking workflow-guided planning for llm-based agents. *arXiv preprint arXiv:2406.14884*.

Xie, J.; Zhang, K.; Chen, J.; Zhu, T.; Lou, R.; Tian, Y.; Xiao, Y.; and Su, Y. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

Yao, S.; Shinn, N.; Razavi, P.; and Narasimhan, K. 2024. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*.