Advanced Machine Learning Progress Evaluation Report

Process-aware Evaluation with Unified Trace Logging and Robustness Metrics

M.A.C. L Mallikarachchi
210363C

Table of Content

1. Introduction

This project develops an improved way of evaluating AutoGen math agents by focusing on the reasoning process rather than final accuracy alone. Popular benchmarks such as AgentBench have shown the value of evaluating agents in diverse environments, but they mainly judge outcomes: whether the task was completed successfully and sometimes at what cost. While useful, this overlooks inefficiencies, recovery behavior, and robustness under different conditions.

To address these gaps, this work introduces AutoGen-TraceKit, a lightweight trace logger that records each interaction step without interfering with the agent. From these logs, new process-aware and robustness metrics are derived, covering efficiency, stability, control-flow health, and resilience. Together these additions make evaluation more transparent, reproducible, and informative, offering a clearer picture of how agents actually behave in practice.

2. Background

Agent evaluation has become an important area in the study of large language model (LLM) systems. Current evaluation methods mainly focus on whether an agent can complete a given task, which makes it easy to compare different systems using standardized benchmarks. Frameworks such as DeepEval, AgentBench and the IBM Agentic Evaluation Toolkit provide different ways to test agents, ranging from simple correctness checks to process-oriented analysis. While these tools have helped set a foundation for evaluating agents, they often place more weight on final results than on the steps the agent takes to reach them. A few Agentic Evaluation Frameworks are as follows:

2.1 DeepEval

DeepEval is an open-source framework made for testing LLM agents. It is easy to add into coding projects and works well for continuous testing. It checks things like correctness, reliability, and consistency of the agent's answers. Its main advantage is making evaluation simple and practical for developers.

2.2 AgentBench

AgentBench is one of the most widely used benchmarks for LLM agents. It places agents in simulated environments and measures whether they achieve the final goal. It standardizes evaluation and allows comparisons between agents. However, it focuses mainly on the final outcome and does not deeply analyze the process of reaching it.

2.3 IBM Agentic Evaluation Toolkit

This toolkit from IBM Watsonx focuses on how agents perform tasks, not just if they finish them. It looks at reasoning steps, decision-making, and error handling. This makes it useful for situations where stability and trust in the agent are important, such as business or enterprise use.

3. Problem description

Although existing frameworks like AgentBench, DeepEval, and IBM's Agentic Evaluation Toolkit have advanced agent evaluation, they still leave important gaps. Current methods focus mainly on outcomes and provide limited insight into the efficiency, reasoning quality, and stability of agents across different conditions. This means that two agents with the same final score may have taken very different paths, with one wasting far more steps or failing to adapt to small changes. Without process-aware and robustness-oriented metrics, evaluations miss these differences, making it difficult to understand the true strengths and weaknesses of an agent.

4. Novel Contribution

The novelty of this project lies in turning raw trajectories into structured, measurable signals that expose hidden trade-offs in agent behavior. Where most agent evaluators evaluate what agents achieve, this work evaluates how they achieve it.

By introducing unified trace logging and new metrics, the project brings three key advances. First, it captures efficiency in a reproducible way, showing how tokens, turns, and time are consumed per solved problem. Second, it measures stability by recording error rates, retries, and recovery effectiveness. Third, it introduces checks to examine how sensitive outcomes are to randomness or small perturbations.

5. Methodology

The evaluation will be conducted in a structured manner to systematically capture different dimensions of agent performance. All experiments are carried out on the MATH benchmark, which serves as a challenging yet reproducible testbed for assessing reasoning ability. The plan is divided into two stages, each designed to progressively build insight into agent behavior.

5.1 Stage I: Baseline Establishment

In the first stage, the agent operates under a standard configuration with fixed parameters. Every interaction is traced in detail, including tokens, turns, latency, and execution outcomes. This stage serves as the foundation for the study, providing a stable reference point against which later variations can be compared. The collected traces yield not only traditional accuracy scores but also the new process-aware metrics of efficiency, stability, and control-flow health. This ensures that the baseline is described in much greater depth than simple correctness alone.

5.2 Stage II: Sensitivity Analysis

The second stage systematically explores the sensitivity of the agent to controlled changes in its operating conditions. Runs are repeated under varying temperature settings, multiple random seeds, and different maximum turn limits. This structured variation allows the evaluation to capture how the agent's performance shifts when exposed to stochasticity and altered constraints. By comparing these runs to the Stage I baseline, the analysis identifies trade-offs between accuracy, efficiency, and stability, revealing conditions under which the agent is more reliable or resource-efficient.

Across both stages, results are analyzed using a unified framework that integrates accuracy with process-aware metrics. This approach ensures that the evaluation highlights not just outcomes but also the pathways and behaviors that lead to those outcomes, offering a more advanced understanding of agent performance than existing benchmarks alone.

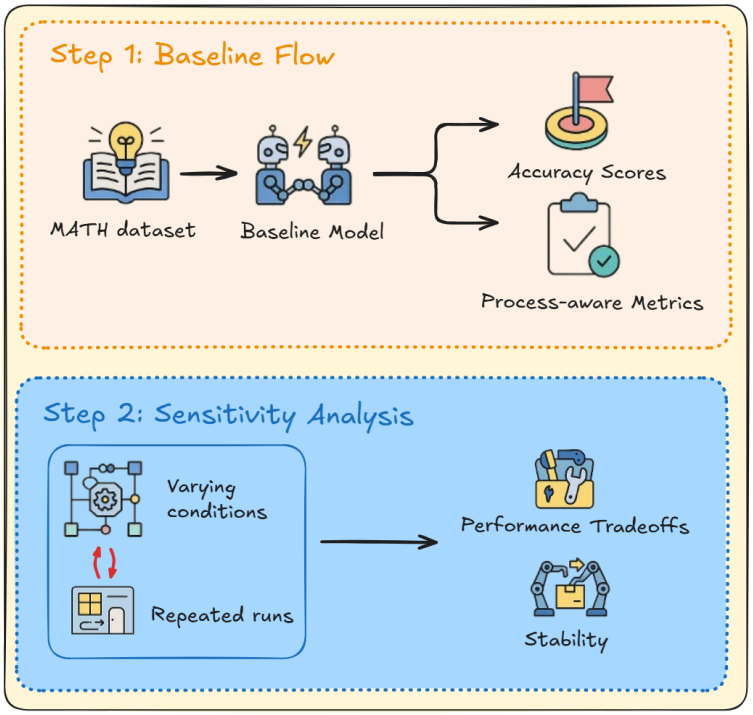

Figure 1: Methodology Overview

6. Experimental plan

The experimental plan is designed to ensure that evaluation runs are both reproducible and practical, while taking advantage of modern infrastructure for efficiency. All experiments will be conducted in a controlled Jupyter Notebook environment, with agents executed under consistent conditions. This allows for structured logging, clear visualization, and systematic metric computation.

The dataset chosen for the evaluation is MATH, consisting of 120 challenging problems. This dataset is widely used in benchmarking and provides a balanced test of advanced reasoning ability. Because it is fixed and public, it ensures that results can be reproduced and directly compared with baseline evaluations.

To make testing feasible and efficient, the experiments will use the Groq API rather than relying on more expensive, paid endpoints such as OpenAI's.

The model powering the assistant agent is DeepSeek-R1-Distill-Llama-70B, a distilled version of DeepSeek's R1 model, fine-tuned from the Llama-3.3-70B-Instruct base. Distillation has preserved strong reasoning capabilities while significantly improving efficiency. Combined with Groq's industry-leading inference speed, this model is well suited for mathematical and logical reasoning, making it an appropriate choice for MATH experiments.

Dataset	MATH, 120 fixed problems, ensuring reproducibility.
Setup	Jupyter Notebook environment for code execution, logging, and visualization.
Inference API	Groq API endpoints to provide fast and reliable inference.
Model	DeepSeek-R1-Distill-Llama-70B

Table 1: Specifications for the experimental plan

7. Feasibility

This project is feasible within the given constraints, both in terms of engineering effort and resource requirements. The evaluation framework builds directly on an existing AutoGen setup, with a logging module and metric definitions developed as extensions. Because these components are observational rather than disruptive, they integrate smoothly without altering the core agent logic.

Running the experiments does not demand extensive hardware. All evaluations are conducted in a Jupyter Notebook environment, with model inference handled externally through API calls instead of local GPUs. This design avoids reliance on high-end hardware and ensures that the workflow remains accessible and portable.

The choice of the Groq API for LLM calls further strengthens feasibility. Groq's endpoints are both fast and cost-efficient, making it possible to repeat experiments many times without creating a financial burden. Wrappers can be integrated into the evaluation code to manage requests and configurations, simplifying the process and improving reliability.

In addition, many resources are already in place to support the work. The dataset is fixed and widely recognized, the experimental pipeline can be adapted from existing AutoGen implementations, and visualization tools are readily available within the notebook environment. Together, these factors provide stability, reduce risk, and ensure smooth progress throughout the project timeline.

8. Timeline

The project timeline, shown in Figure 2, outlines the progression from planning to final submission. The early phase focuses on research work and the development of the methodology. This is followed by the experiment setup stage, where the baseline implementation and dataset runs are completed. The evaluation phase introduces metric computation, sensitivity experiments, and error handling to test the agent under different conditions. In parallel, analysis and writing activities produce the mid-evaluation and final papers. The final stage is dedicated to code finalization and submission, ensuring that all deliverables are completed within the required deadlines.

Figure 2: Project timeline

9. References

1. H. Liu, C. Zheng, et al., "AgentBench: Evaluating LLMs as Agents," *arXiv preprint* arXiv:2308.03688, 2023. [Online]. Available: https://arxiv.org/abs/2308.03688

2. Microsoft Research, *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework*. GitHub repository. [Online]. Available: https://github.com/microsoft/autogen

3. IBM, *Agentic AI Evaluation (watsonx)*. IBM Documentation. [Online]. Available: https://www.ibm.com/docs/en/watsonx/saas?topic=sdk-agentic-ai-evaluation

4. Confident AI, *DeepEval: Open-source evaluation framework for LLM applications*. GitHub repository. [Online]. Available: https://github.com/confident-ai/deepeval

5. D. Hendrycks, C. Burns, S. Basart, et al., “Measuring Mathematical Problem Solving With the MATH Dataset,” *arXiv preprint* arXiv:2103.03874, 2021. [Online]. Available: https://arxiv.org/abs/2103.03874

6. Hugging Face, *Hendrycks MATH Benchmark Dataset*. [Online]. Available: https://huggingface.co/datasets/nlile/hendrycks-MATH-benchmark

7. DeepSeek AI, *DeepSeek-R1-Distill-Llama-70B*. Model card. [Online]. Available: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-70B

8. Groq Inc., *Groq API Documentation*. [Online]. Available: https://groq.com