AgentDiagnose: An Open Toolkit for Diagnosing LLM Agent Trajectories

Tianyue Ou Wanyao Guo Apurva Gandhi Graham Neubig Xiang Yue
Carnegie Mellon University

Abstract

Large Language Model (LLM) agents produce rich, multi-step trajectories that interleave observations, internal reasoning, and tool actions. However, most evaluation pipelines focus solely on end-task success, leaving the agent's decision-making process opaque and poorly understood. We introduce AgentDiagnose, an open-source, modular framework for diagnosing agent trajectories. The present release fully supports the web domain, and AgentDiagnose is architect as an extensible, open platform with compatibility for most agent trajectories. AgentDiagnose consists of (i) an *evaluation* module that quantifies five core agentic competencies—backtracking & exploration, task decomposition, observation reading, self-verification, and objective quality—and (ii) a *visualization* module that highlights trajectory semantics through t-SNE action embeddings, interactive word clouds, and state-transition timelines. On a set of 30 manually annotated trajectories, our automatic metrics achieve a mean Pearson correlation of 0.57 with human judgments, rising to 0.78 for task decomposition. Furthermore, filtering the 46k-example NNetNav-Live dataset with AgentDiagnose and fine-tuning a Llama-3.1-8B model on the top 6k trajectories improves WebArena success rates by 0.98, despite using only 13% of the original data. AgentDiagnose thus serves as both a diagnostic lens for agent analysis and a practical tool for curating higher-quality training data. The toolkit and demo are publicly available.[1][2]

1 Introduction

Large Language Model (LLM) agents are rapidly gaining traction in domains such as web navigation (Xu et al., 2025; Murty et al., 2025), GUI automation (Xie et al., 2024; Qin et al., 2025), software engineering (Wang et al., 2025; Jimenez et al., 2024), social interaction (Jhamtani et al., 2025), and even healthcare (Tang et al., 2024).

Unlike traditional models that produce a single output, LLM agents operate over multi-step interaction trajectories. Each trajectory interleaves external observations, internal reasoning steps, and executable actions, producing an output sequence of the form: $T = \{(o_1, r_1, a_1), (o_2, r_2, a_2), \ldots, (o_n, r_n, a_n)\}$, where o_i represents the observation at step i, r_i the agent's reasoning, and a_i the resulting action. These trajectories can span hundreds or thousands of steps (Anthropic, 2025), making it difficult to analyze what drives an agent's success or failure.

Existing agent evaluation pipelines focus primarily on whether an agent completes a task successfully. While this outcome-oriented evaluation is useful, it leaves the agent's decision-making process opaque. Inspection tools such as AgentXRay (Murty et al., 2024) and AgentOps (AgentOps-AI, 2025) present step-level breakdowns of trajectories, visualizing the agent's observations, reasoning, and actions. However, they mainly provide readable replays of what the agent did, rather than diagnosing how well the agent reasoned about the task or how effectively it corrected mistakes along the way.

Complementary to these, evaluator frameworks such as AgentEval (LangChain-AI, 2025) judge an agent's trajectory against a reference solution, often using an LLM-as-a-judge. These approaches provide holistic scores but require gold-standard trajectories for comparison and overlook agent-internal competencies such as task decomposition or exploration.

In this paper, we introduce AgentDiagnose, an open-source diagnostic toolkit designed to fill this gap. AgentDiagnose helps researchers and practitioners go beyond success/failure judgments and surface key properties of agentic behavior. It consists of two core modules: 1) **Evaluation module:** Automatically scores five agentic competen-

[1] https://AgentDiagnose.live/
[2] https://github.com/oootttyyy/AgentDiagnose

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 207–215

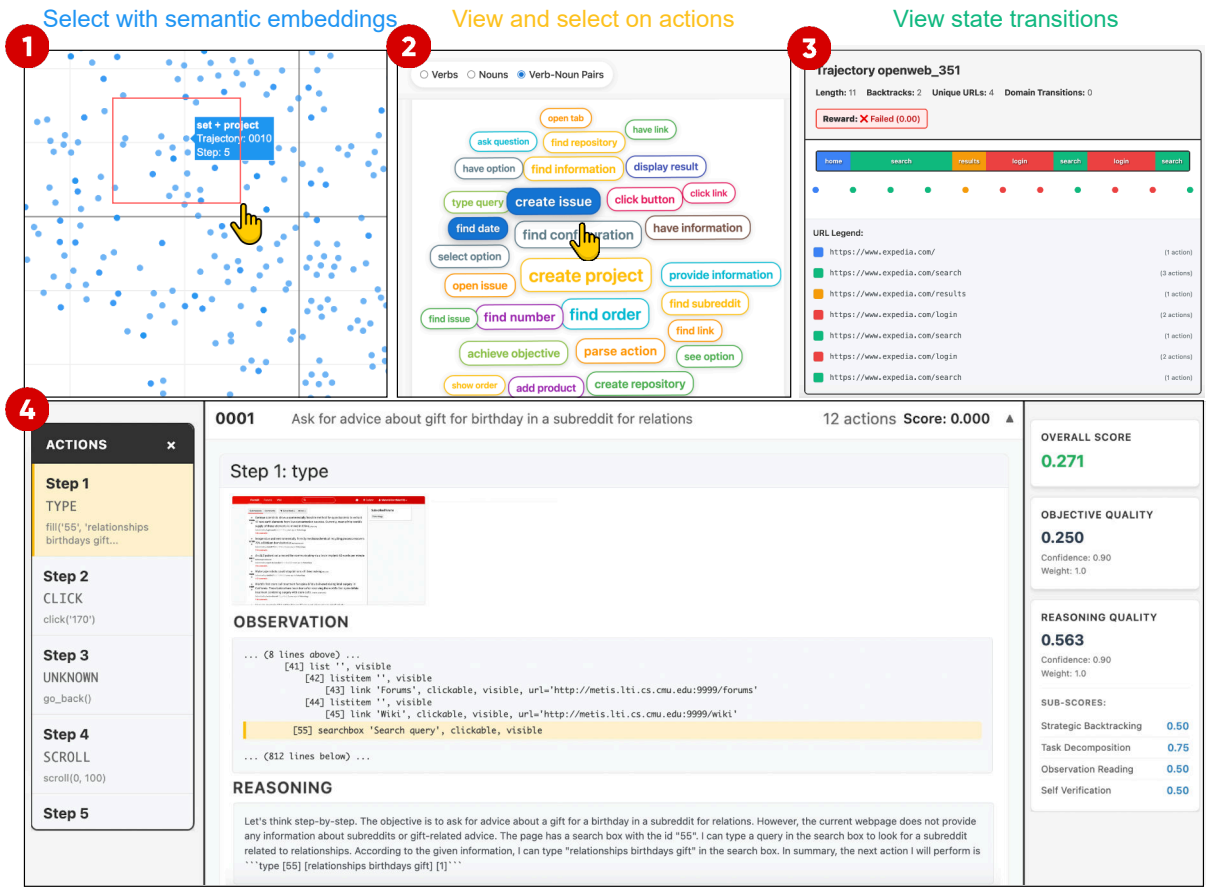


Figure 1: Overview of dashboard. 1 shows selecting points of interests in semantic embedding plot. 2 shows selecting on phrases of interests in word cloud. Selecting in 1 and 2 will display corresponding trajectories in trajectory_viewer 4. 3 shows navigation graph. 4 the trajectory_view panel displays details of all trajectories with evaluator scores summaries.

cies—backtracking & exploration, task decomposition, observation reading, self-verification, and objective quality—without requiring reference trajectories. 2) **Visualization module:** Provides interactive plots (e.g., t-SNE embeddings, word clouds, state transition timelines) that make it easy to explore semantic patterns in trajectories and pinpoint problematic behaviors.

We demonstrate AgentDiagnose's utility both as an analytical tool and as a data selection pipeline. Automatic trajectory scores correlate well with human annotations (mean Pearson $r = 0.57$, up to 0.78 for task decomposition). Furthermore, fine-tuning an LLM agent on AgentDiagnose-filtered high-quality trajectories from the NNetNav-Live dataset leads to a significant performance boost: a Llama-3.1-8B model trained on only the top 13% of trajectories outperforms one trained on the full dataset. In summary, AgentDiagnose provides a much-needed microscope for diagnosing agentic reasoning and offers practical levers for improving dataset quality and agent performance.

2 Related Work

Trajectory Inspection and Monitoring : A number of tools have been developed to visualize and monitor agent trajectories. For example, AgentLab's "Agent X-Ray" interface (Chezelles et al., 2025), IBM's "Agent Trajectory Explorer" (Desmond et al., 2025), OpenHand's "Trajectory Visualizer" (All-Hands-AI, 2025) provide step-by-step replay of an agent's interactions with a web environment, showing each observation, the agent's intermediate reasoning, and the action taken. Similarly, these inspection frameworks focus on rendering the raw trajectory in a human-readable form, often as a sequence of browser screens or tool API calls with the agent's thoughts. In parallel, agent observability platforms like AgentOps (AgentOps-AI, 2025), SEAVIEW (Bula et al., 2025) emphasize monitoring metrics and logging each decision step. AgentOps provides a developer dashboard to trace

agent behavior and detect anomalies or policy violations in real time. However, while these tools excel at presenting or recording the trajectory, they do not analyze the agent's decision-making quality beyond basic logging. For instance, they typically do not quantify how well an agent backtracked from errors or whether its plan was well-structured – gaps that our work aims to fill.

Trajectory Evaluation Frameworks : Beyond visualization, recent efforts have looked at evaluating the quality of agent trajectories. One approach is to compare the agent's sequence of actions against an expected or ideal sequence for the task. AgentEvals (Chase and Contributors, 2023), Vertex AI Agent Eval (Google Cloud, 2025) are frameworks that automates such comparisons. It can perform a trajectory match evaluation or use an LLM-as-a-judge to score how closely an agent's trajectory aligns with a reference solution. This method provides a holistic success measure of a trajectory when a reference is available. Our proposed toolkit differs by diagnosing trajectories along general-purpose dimensions without needing a predetermined correct sequence.

3 AgentDiagnose

AgentDiagnose can be used as a terminal tool or a visualization dashboard. It consists of two modules. (1) An evaluation module that gives a numeric evaluation score on key properties of trajectories. (2) A visualization module that directly displays key properties that are difficult to quantize into scores.

Key Trajectory Metrics Trajectories contain rich information that can be easily buried in the long chain of observation, reasoning, and actions. For instance, agents' adeptness in exploring alternatives (Shen et al., 2025) is a key measure of an agent's ability but it is not immediately clear how good an agent is in this regard just by looking at the raw sequence of steps. AgentDiagnose supports adding customized evaluators to access any property of interest. And as an example, we implemented five such evaluators to expose key agentic properties from trajectories as listed in Figure 2:

- **Backtracking and Exploration**: exploring alternatives is a key aspect of good agents. AgentDiagnose measures and exposes how good an agent is in backtracking when it heads into a wrong path. AgentDiagnose looks for patterns of backtracking and exploration in agents' sequence of actions and their corresponding reasoning. It uses a designed criterion to assign numeric scores based on the patterns it has identified.

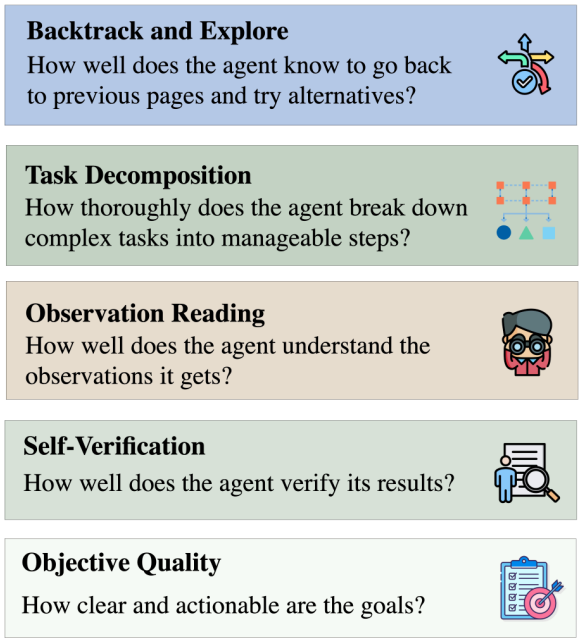


Figure 2: Description of key trajectory properties.

- **Task Decomposition**: knowing how to best decompose complex tasks into manageable sub-tasks is a key ability of successful agents (Prasad et al., 2023). Most of the decomposition happens at the first step, but studies have shown that revisiting and updating plans as agent interacts with the environment further improves performance. To help expose agents' abilities to decompose tasks, AgentDiagnose finds patterns in reasoning traces of the trajectory and assign numeric scores with our evaluation criteria. It exposes agents' abilities in decomposing complex tasks into manageable steps.

- **Self-verification**: AgentDiagnose measures how well the agent is in self-verifying its course of actions against the task goal. Self-verification is beyond checking if the final answer is satisfying the task requirement. Reasoning within each step of execution should reflect agent's awareness of the overarching goal.

- **Observation Reading**: correctly finding the right information from the observation is a major challenge for agents (Cheng et al., 2024). AgentDiagnose helps evaluate on the

agents' abilities to correctly understand the observation and capture relevant information.

- **Objective Quality**: the clarity and actionability of the task goal are a deciding factor in a trajectory's quality. Without a well-defined goal, it is difficult to determine how well the agent has completed the task. AgentDiagnose also utilizes a scoring criterion to measure how clear and actionable an objective is.

Key Property Visualizer However, not all information is best suited to be summarized by a numeric score. For instance, to understand the action distribution of agents, a t-SNE embedding plot of action verbs provides a more comprehensive view (van der Maaten and Hinton, 2008). AgentDiagnose comes with a visualizer that displays trajectories' intrinsic information through a combination of t-SNE embedding plots, word cloud, and timeline plots.

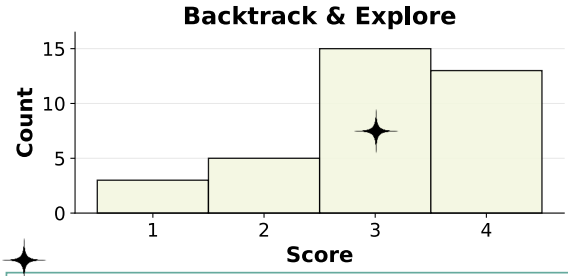
AgentDiagnose provides methods to extract out action verbs and nouns within agents' reasoning as shown in Figure 1 1. Currently AgentDiagnose has six options built in: root verbs, root nouns, root verb-noun pairs, all verbs, all nouns, and all verb-noun pairs. Verbs and nouns are extracted from agents' reasoning using Berkeley Neural Parser (benepar) (Kitaev et al., 2018). They are embedded with embedding models of choice, with the default being Qwen3-Embedding-0.6B (Zhang et al., 2025). AgentDiagnose generates t-SNE plots for each embedding. Users can select areas of interest on the semantic embedding plot. Once selected, users can inspect further on the trajectories of interests in the 4 view_trajectory tab, as shown in Figure 1.

AgentDiagnose also provides word cloud visualization on various components in agents' output. As shown in Figure 1 2, users can view the root verb, root noun, and root verb-noun pairs in the form of word cloud. In addition to action terms, AgentDiagnose builds word cloud for n-grams of reasoning output of agents. User can customize n-grams of n to display in a word cloud to uncover patterns in agents' reasoning. Similar to t-SNE plots, AgentDiagnose's word cloud supports selection of any one or multiple of the phrases in the word cloud to display their corresponding trajectories in the view_trajectory tab.

As shown in Figure 1 3, AgentDiagnose provides visualization of navigation paths within trajectories for direct view of the state transitions in each trajectories. Each color block on the timeline represents a state (a URL in this case), while each dot represents an action performed while in this state. Agent behaviors such as backtracking and parallel exploration can be easily identified through this view.

4 Usage Cases

4.1 Curate High Quality Trajectories



Goal: Find the best Peloton bike seat.

...

Step 9:

Let's think step-by-step. We've already browsed through Peloton Bike accessories on Amazon and sorted them by price. We also seem to have visited a product page with 2,826 ratings. ..... We can see that its value is 'Peloton Bike accessories', which confirms our search query. However, to determine if we're looking at the best product, we need to analyze the product's details and possibly other customer reviews. Since I couldn't find any extra details about the product in this accessibility tree and I also can't find the 'next' button for looking at other products, I think it's good idea to go back and try a different search term or sort option. In summary, the next action I will perform is ```go_back```

...

Figure 3: Top: Backtrack & explore score distribution from the evaluator. Bottom: corresponding step that exhibits exploratory behavior.

The automatic evaluator of key properties is a fast way to identify strengths and weaknesses on a trajectory sample. AgentDiagnose's evaluator is implemented both as a python extension that can be invoked with a single command line in the terminal, as shown in 1, and as a part of the dashboard for visualization. Upon running the evaluator, results are immediately stored locally. Users can further choose to spawn them as interactive bar plots in separate tabs in the visualization dashboard. One use-case of the evaluated scores is to filter for high quality training data. Here we show an example of evaluation results. As seen in Figure 3, this set of trajectories exhibits nice backtracking and exploration ability, with most of the scores in the threes and fours. As in shown in the detail reasoning, one such trajectory demonstrates the agents' intent to explore additional Peloton products to make sure it is finding the "best one."

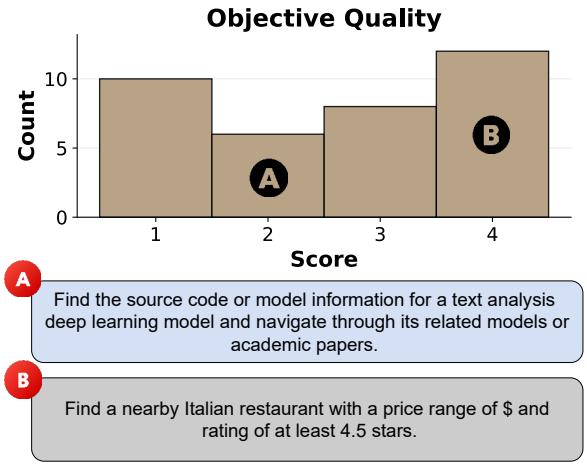


Figure 4: Top: Objective quality score distribution from the evaluator. Bottom: corresponding objectives that are scored above.

Evaluation on objective quality of the batch indicates objective of this batch has mixed quality. As shown in Figure 4, around half of the objectives in the batch have scores of ones and twos. Corresponding objectives are shown in the figure. Higher scores of objective correspond to more actionable and specific task objectives. Objective A, as shown in the figure, does not have a clear end goal and therefore is not a good training sample. Based on these scores, we are able to select high quality training data that demonstrates good backtrack and explore behaviors as well as high quality task objectives. Users can further display the plots on dashboard. To select and view the trajectories of a particular score, users can select score ranges on the bar plot displayed in the dashboard, corresponding trajectories will then be displayed in the view_trajectory tab. As users go through the trajectories, a mini-panel of scores will also be displayed for reference, as shown in Figure 1. Detailed scoring criteria are shown in Appendix A.

4.2 Visualize Intrinsic Properties

We follow the trajectory inspection paradigm:

- Rapidly skim trajectories to identify emerging patterns.
- Select key patterns for detailed, in-depth analysis.

AgentDiagnose provides methods to visualize various intrinsic properties of trajectories that would otherwise buried in sequence views.

Embedding Plot As shown in (1) of Figure 1, users can inspect clustering of actions verb-noun

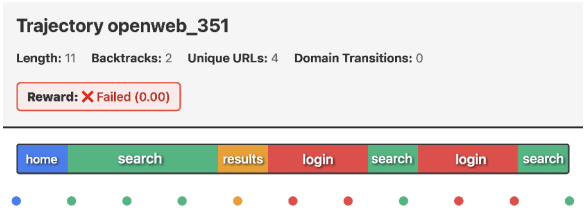


Figure 5: Screenshot of the navigation graph.

embeddings. For example, when an agent fails in setting up GitLab projects, users want to inspect trajectories that involved handling GitLab projects. To do so, they can select from the embedding plot as in (1) of Figure 1, on the semantic embeddings surrounding the point of "set project". Surrounding the point of "set project" are points of "create project", "fork project" etc. By selecting this region, users can pull up the corresponding trajectories in the view_trajectory page for detailed investigation. Semantic embedding plot allows users to scan through patterns and easily focus on any interesting semantic patterns and their corresponding trajectories.

Word Cloud As shown in (2) of Figure 1, users can use word clouds to identify common patterns in an agent's reasoning. AgentDiagnose supports word cloud for actions and n-grams in reasoning. In the example of (2) Figure 1, verb-noun pairs in agents' reasoning are displayed by their frequencies of appearance. We can see the verb-noun pair "create project" has appeared frequently. We can click on any one or multiple of the tabs to display their corresponding trajectories in the view_trajectory tab.

Listing 1: Command-line to run evaluator.

```
python evaluate_trajectories.py \
  --input ./input.json \
  --scorers reasoning_quality \
            objective_quality \
            navigation_path \
  --output-json output.json
```

Navigation Graph Users can use navigation path to quickly spot navigation-related errors. As shown in (3) of Figure 1, the navigation path abstracts out the state transitions from the trajectories so it is clear where the agent has been. One use case is to identify navigation-related errors. In the example shown Figure 5, the failed trajectory of openweb_351 is stuck in a loop between login and search page. Considering that the agent has used search page successfully early in the trajectory, the possible error may be in handling login page. The

Dimension	r	ρ	τ
backtrack & explore	0.39	0.43	0.36
task decomposition	0.78	0.86	0.76
observation reading	0.62	0.63	0.60
self-verification	0.56	0.58	0.55
objective quality	0.54	0.61	0.56
Overall	0.57	0.62	0.56

Table 1: Correlation coefficients (Pearson r, Spearman ρ, Kendall τ) between human annotated scores and evaluator's on five key properties.

abstraction view of navigation path allows users to quickly scan through the path agent has taken within each trajectory.

5 Evaluation

5.1 Compare to Human Evaluation

To evaluate the accuracy of our evaluator, we manually annotated 30 trajectories on the five key properties and compare with the model evaluated scores. We give the same input of agent reasoning-action sequence and scoring criteria to both the model and annotator. We measured the agreement with Pearson (Pearson, 1895), Spearman (Spearman, 1904), and Kendall-Tau correlation coefficients (Kendall, 1938). Overall, AgentDiagnose's evaluator achieves a 0.57 Pearson correlation coefficient, 0.62 Spearman correlation and 0.56 Kendall-Tau correlation, as shown in Table 1. Overall, it showcases a positive correlation between human scoring and AgentDiagnose's evaluator scoring. In particular, agreement rate is the highest in the case of task decomposition, reaching 0.78 Pearson correlation, showing that task decomposition behavior is relatively easy to identify and differentiate. In contrast, backtracking and exploration only reaches a 0.39 Pearson correlation, indicating the intrinsic difficulty in extracting agents backtracking behaviors and evaluating their precision and correctness.

5.2 Improve Agent Performance With AgentDiagnose

AgentDiagnose's evaluator act as effective training data selector. We experimented by fine-tuning on the full NNetNav-Live dataset and a high quality subset selected by AgentDiagnose's evaluator. We applied AgentDiagnose's evaluator on the 46k training samples of the NNetNav-Live dataset, using Gemini2.5-pro as the LLM judge (Gemini Team, Google, 2025), and we selected

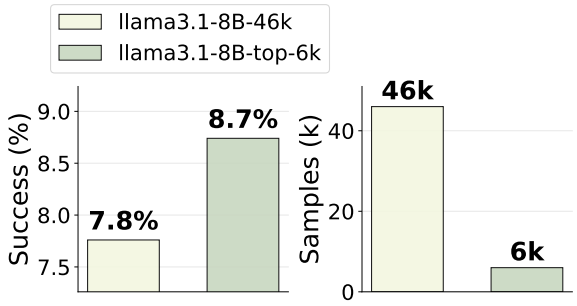


Figure 6: Comparison of WebArena success rate(left) and number of samples used in training(right) by llama3.1-8B-46k and llama3.1-8B-top-6k.

6k top scoring samples as our high quality subset. Following the setup in NNetNav, we finetuned a LLama3.1-8B model on our high quality subset and evaluated on WebArena with browserGym (Zhou et al., 2023), (Chezelles et al., 2025). Results are shown in Figure 6: llama3.1-8B-top-6k outperforms llama3.1-8B-46k despite only using 13% of the training samples. llama3.1-8B-top-6k achieves 0.98 higher success rate than Llama-3.1-8B -46k. With AgentDiagnose evaluator's filtering ability, it could act as data quality controller in training data collection and provide valuable signal on trajectory's quality beyond the success and failure measurement.

6 Conclusion

We demonstrated AgentDiagnose, an open-source framework that brings much-needed diagnostic visibility to the rapidly growing field of LLM agents. By coupling a lightweight, LLM-powered evaluation module with an interactive visualization module, AgentDiagnose exposes five critical competencies: backtracking & exploration, task decomposition, observation reading, self-verification, and objective quality—that are largely invisible to traditional trajectory inspectors. And by improving agent's performance on web navigation task, we showcase our trajectory-centric quality signals can translate directly into performance gains.

7 Limitations

AgentDiagnose is designed primarily with web-navigation tasks in mind. Its applicability to other use cases, such as coding, as well as multimodal settings.

Current AgentDiagnose's evaluator system leverages LLM-based evaluation. Future iterations could improve reliability by training specialized evaluators with fine-grained human annotations.

8 Ethics Statement

Human annotation Thirty trajectories were annotated for the correlation study. Annotators are college student volunteers. No demographic attributes beyond language proficiency were collected. The task required no sensitive information.

Broader Impact By shining light on how agents reason, rather than solely whether they succeed, AgentDiagnose can help researchers identify failure modes, debug unsafe behaviors, and design more transparent systems.

References

AgentOps-AI. 2025. Agentops: Python sdk for ai agent monitoring, llm cost tracking, benchmarking, and more. https://github.com/AgentOps-AI/agentops. MIT License.

All-Hands-AI. 2025. All-hands-ai/trajectory-visualizer. https://github.com/All-Hands-AI/trajectory-visualizer. GitHub repository, accessed July 3, 2025.

Anthropic. 2025. Claude opus 4. https://www.anthropic.com/claude/opus. Accessed: 2025-07-01.

Timothy Bula, Saurabh Pujar, Luca Buratti, Mihaela Bornea, and Avirup Sil. 2025. Seaview: Software engineering agent visual interface for enhanced workflow. *Preprint*, arXiv:2504.08696.

Harrison Chase and LangChain Contributors. 2023. Langchain. https://www.langchain.com/. Accessed: 2025-07-03.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Annual Meeting of the Association for Computational Linguistics*.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. 2025. The browsergym ecosystem for web agent research. *Preprint*, arXiv:2412.05467.

Michael Desmond, Ja Young Lee, Ibrahim Ibrahim, James M. Johnson, Avirup Sil, Justin MacNair, and Ruchir Puri. 2025. Agent trajectory explorer: Visualizing and providing feedback on agent trajectories. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(28):29634–29636.

Gemini Team, Google. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. Technical report, Google DeepMind, Mountain View, CA. Version 2.5: Pro and Flash variants; introduces million-token context window, multimodal input, advanced 'Deep Think' reasoning.

Google Cloud. 2025. Evaluate gen ai agents. Documentation for evaluating generative AI agents on Vertex AI.

Harsh Jhamtani, Jacob Andreas, and Benjamin Van Durme. 2025. Lm agents for coordinating multi-user information gathering. *Preprint*, arXiv:2502.12328.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Maurice G. Kendall. 1938. *A New Measure of Rank Correlation*, volume 30. Biometrika Trust.

Nikita Kitaev, Lukasz Kaiser, and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686. Association for Computational Linguistics.

LangChain-AI. 2025. Agentevals: Readymade evaluators for agent trajectories. https://github.com/langchain-ai/agentevals. Version 0.0.8.

Shikhar Murty, Dzmitry Bahdanau, and Christopher D. Manning. 2024. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. 2025. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *Preprint*, arXiv:2410.02907.

Karl Pearson. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2023. Adapt: As-needed decomposition and planning with language models. In *NAACL-HLT*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, and Aviral Kumar. 2025. Thinking vs. doing: Agents that reason by scaling test-time interaction. *Preprint*, arXiv:2506.07976.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2024. Medagents: Large language models as collaborators for zero-shot medical reasoning. *Preprint*, arXiv:2311.10537.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, and 5 others. 2025. Openhands: An open platform for ai software developers as generalist agents. *Preprint*, arXiv:2407.16741.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. 2025. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. *Preprint*, arXiv:2412.09605.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2023. Webarena: A realistic web environment for building autonomous agents. *ArXiv*, abs/2307.13854.

A Evaluation Criteria

```
Backtracking (1-4): How well does the agent know to go back to previous pages and try alteratives?
- 4: Excellent - The agent accurately recognizes when it has taken a wrong path and take explicit actions to go back to a previous page to try alteratives
- 3: Good - The agent takes explicit actions to go back to try alteratives most of the time when it takes a wrong path
- 2: Mediocre - The agent has considered going back or trying alteratives, but has made mistakes in doing so
- 1: Poor - The agent has never considered trying alternatives or going back to previous states
- N/A: There is not a need to go back to previous states because the agent has taken the right path throughout the trajectory
```

```
Task decomposition (1-4): How thoroughly does the agent break down complex tasks into manageable steps?
- 4: Excellent - The agent breaks down complex tasks into detailed steps that cover the entire task
- 3: Good - The agent breaks down complex tasks, but not in all cases or leaves out steps
- 2: Mediocre - The agent breaks down complex tasks, but in very poor way
- 1: Poor - The agent makes no attempt in breaking down complex tasks
```

```
Observation reading (1-4): How well does the agent understands the observations it gets?
- 4: Excellent - The agent summarizes the observation accurately in each step and immediately notice the important information on the page
- 3: Good - The agent summarizes the observation in each step, but sometimes misses important information
- 2: Mediocre - The agent only summarizes the observation in some steps
- 1: Poor - The agent almost never summarizes the observation
```

```
Self-verification (1-4): How well does the agent verify its results?
- 4: Excellent - The agent checks carefully on its results against the objective throughout the trajectory
- 3: Good - The agent checks its results against the objective sometimes, but has room to improve. If it has done better checking, it could have done better on the task
- 2: Mediocre - The agent shows signs of attempting to verify its results
- 1: Poor - The agent never verifies its results against the objective
```

```
Objective-quality (1-4): How clear and actionable are the goals?
- 4: Excellent - Objective contains clear, specific, actionable goals with concrete success criteria
- 3: Good - Objective is mostly actionable with some clear goals
- 2: Mediocre - Objective has a mix of actionable elements and vague exploratory elements
- 1: Poor - Objective is entirely about exploration with no concrete targets
```