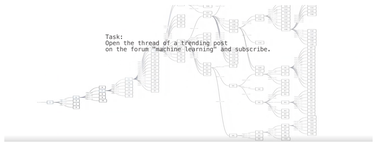

Multi-Modal LLM Reasoning and Agent Modeling

Trevan Nguyen, Nathaniel del Rosario, So Hirota, Aaryan Agrawal, Zihan Liu, Samuel Zhang, Zhiting Hu

ttn077@ucsd.edu, nadelrosario@ucsd.edu, shirota@ucsd.edu, aaagrawal@ucsd.edu, zil065@ucsd.edu, saz004@ucsd.edu, zhh019@ucsd.edu

UC San Diego™
HALICIOĞLU DATA SCIENCE INSTITUTE

Introduction

Large language models (LLMs) address limitations of traditional heuristic approaches in unpredictable environments by leveraging textual and structural web data, allowing for more flexible decision-making through exploring the Tree of Thought [4]. Unlike static heuristics, LLM-based agents can reason dynamically and handle open-ended tasks. Additionally, scaling computation improves their performance, though determining the optimal way to do so remains a challenge.

LLM-Reasoners [2] is a standardized, library for creating reasoning agents with a modular framework for customizing the LLM, search algorithm, search configuration, world model, and benchmark architecture. We leverage LLM-Reasoners to investigate this behavior in Monte Carlo Tree Search (MCTS) scaling experiments.

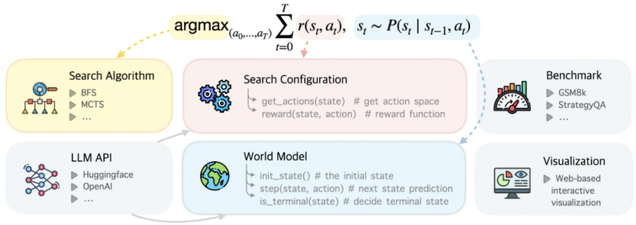

Fig. 1: Custom Reinforcement Learning via LLM-Reasoners

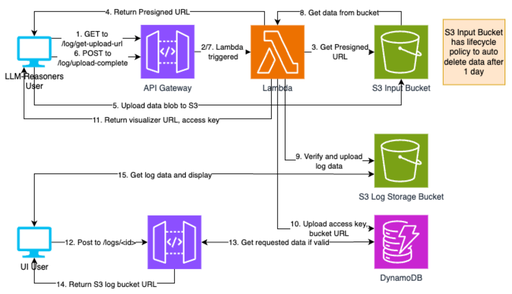

Fig. 2: Visualizer Architecture

Implicit Search: Agent attempts to correct its own mistakes through direct actions, such as clicking an undo button.

- Fast and computationally cheap.
- Recover with LLM but often fails with complex scenarios & incorrect recovery.
- Equivalent to single-iteration MCTS with no accumulated visitation statistics.

Explicit Search: Uses MCTS & LLM prompting/self-evaluation to guide backtracking and exploration, leveraging cached states and structured rollouts.

- More reliable, structured decision-making.
- Computationally expensive due to multiple rollouts and evaluations.
- Requires assumptions about the ability to backtrack

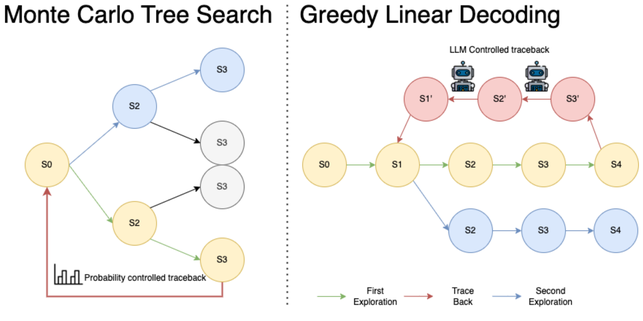

Fig. 3: Test-Time Scaling

Methods & Results

Implicit vs. Explicit experiments were conducted on a subset of 106 / 50 tasks from the WebArena / OSWorld benchmarks using GPT-4o-mini / R1-Distilled and UITARs 7B / 72B respectively.

Browsergym [1] is a web-based environment designed to evaluate web agents, functioning similarly to OpenAI Gym but for browser interactions.

Name	Successes	Failures	Errors
4o-mini MCTS(depth=5, iterations=10)	15	89	2
4o-mini MCTS(depth=10, iterations=10)	21	82	3
4o-mini MCTS(depth=20, iterations=10)	35	67	4
4o-mini MCTS(depth=100, iterations=1)	23	71	4
r1-distill-32b MCTS(depth=5, iterations=10)	25	81	0
r1-distill-32b MCTS(depth=10, iterations=10)	34	69	3
r1-distill-32b MCTS(depth=20, iterations=10)	35	70	1
r1-distill-32b MCTS(depth=100, iterations=1)	28	75	3

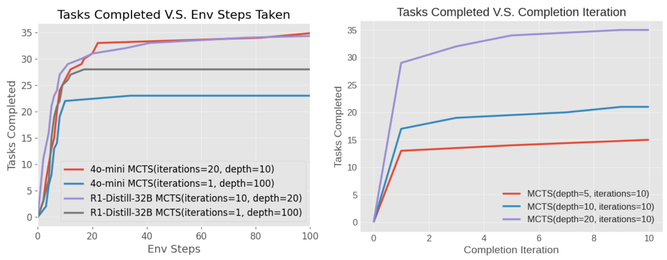

Fig. 5: Implicit vs Explicit Performance

OsWorld [3] is a desktop environment for evaluating agents on operating system benchmarks such as Chrome, VSCode, Gimp, etc.

Name	10 steps	15 steps	30 steps	40 steps
UI-TARS 7b	6.1	6.38	8	14
UI-TARS 70b	8	14	14.58	19

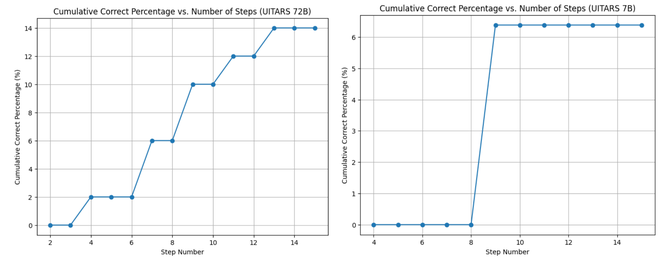

Fig. 7: UITARs 7B / 72B Explicit Scaling

Discussion

Scaling Consideratinos The total cost of the BrowserGym experiments was $44.63 USD, with a full dataset evaluation on GPT-4o estimated at $5,355.60 USD, making large-scale runs prohibitively expensive. Token usage increased significantly with search depth, with cached prompt tokens helping to mitigate costs. Execution time closely followed token consumption trends.

Explicit Search on Environment Performance improves with increased iterations and depth, as seen in Figures 5 & 7. Greater depth allows agents to correct suboptimal actions, increasing task success rates. However, most successful completions occur in early iterations, suggesting that while iterations help, depth is the more crucial factor. Failure cases tend to max out iterations, indicating that being stuck in a poor search space is a major issue.

Comparing Explicit and Implicit Search Implicit search benefits from additional depth but suffers from diminishing returns beyond depth 20, often leading to repetitive or noop() actions. Explicit search, leveraging MCTS for structured exploration and backtracking, consistently outperforms implicit search. However, implicit search remains preferable for real-world applications where external server states prevent reliable backtracking.

Conclusion

Our experiments reveal that explicit search via MCTS outperforms implicit search by improving backtracking and exploration-exploitation trade-offs, though its reliance on resettable states limits real-world applicability. Token and time costs scale significantly with depth and iterations, with most successful completions occurring early in explicit search. Depth is crucial for WebArena tasks, aiding recovery from suboptimal actions, but both search methods struggle when trapped in poor subspaces. To conclude, the R1-Distilled-32B model performed substantially better than both 4o-mini and 4o, and in general task success saw an increased correlation with the number of MCTS iterations and depth.

Acknowledgements

Thank you to our advisor, Professor Zhiting Hu, and his two PhD students, Zhoujun Cheng and Shibo Hao.

References

[1] De Chezelles et al. “The Browsergym Ecosystem for Web Agent Research”. In: *arXiv preprint arXiv:2412.05467* (2024).

[2] Shibo Hao et al. “LLM Eeasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models”. In: *arXiv preprint arXiv:2404.05221* (2024).

[3] Tianbao Xie et al. “Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments”. In: *arXiv preprint arXiv:2404.07972* (2024).

[4] Shunyu Yao et al. “Tree of thoughts: Deliberate problem solving with large language models”. In: *Advances in Neural Information Processing Systems* 36 (2024).

Website