AI Agents That Matter

Kapoor et. al.

Princeton University

arXiv, 2024.07

2024.08.20

Presenter: Hawon Jeong

Introduction

Agent evaluation differs from language model evaluation in fundamental ways 🤖

1. AI agent evaluations must be cost-controlled
2. Jointly optimizing accuracy and cost can yield better agent design
3. Model developers and downstream developers have distinct benchmarking needs
4. Agent benchmarks enable shortcuts
5. Agent evaluation lack standardization and reproducibility

Introduction

What is an AI agent?

"Agentic"

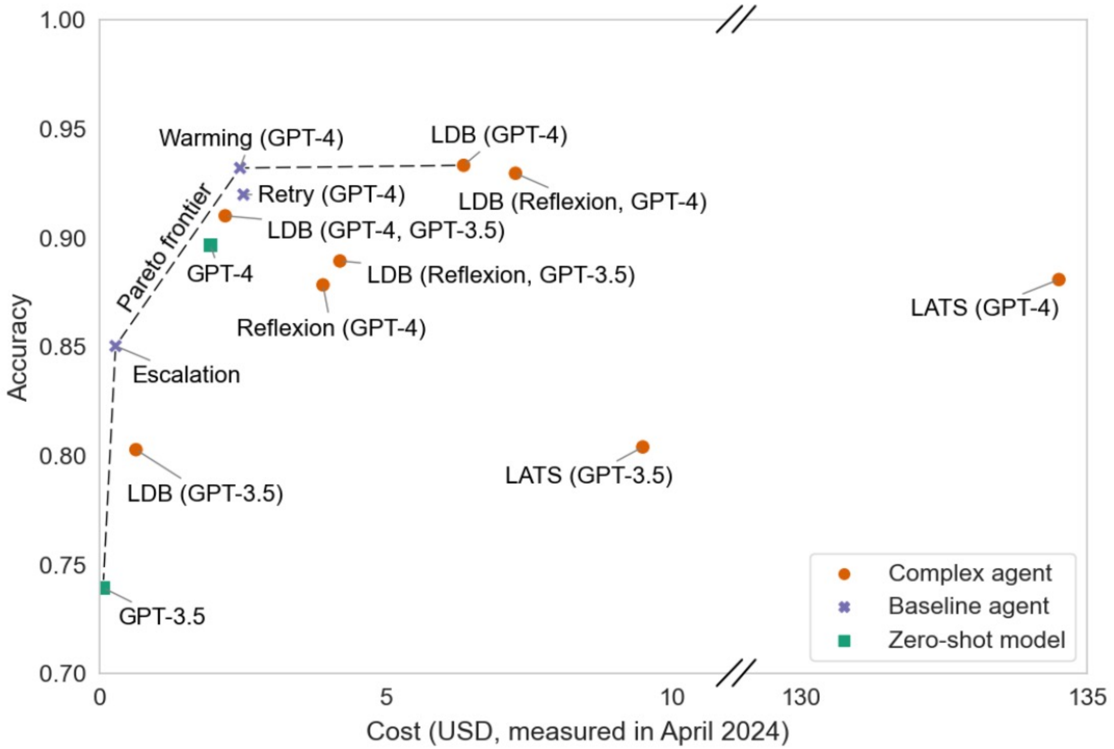
- More complex **environment and goals**
- **User interface and supervision**
  - Instructed in natural language and act on the users' behalf
  - Less user supervision
- **System design**
  - Design patterns such as tool use or planning

1. AI agent evaluation must be cost-controlled

- **Maximizing accuracy can lead to unbounded cost**
  - Agent developers can keep sampling from a model until the solution passes the test cases
- **Visualizing the accuracy-cost tradeoff using a Pareto curve**
  - Agents: LDB, LATS, and Reflexion (from the HumanEval leaderboard)
  - Models: GPT-3.5, GPT-4
  - Approach:
    - Retry
    - Warming
    - Escalation: Llama-3 8B → GPT 3.5 → Llama-3 70B → GPT-4

1. AI agent evaluation must be cost-controlled

- **Two-dimensional evaluation yields surprising insights**
  - “State-of-the-art” agent architectures for HumanEval do not outperform simple baselines
  - Agents differ drastically in terms of cost
  - Lack of evidence that System 2 approaches are responsible for performance gains

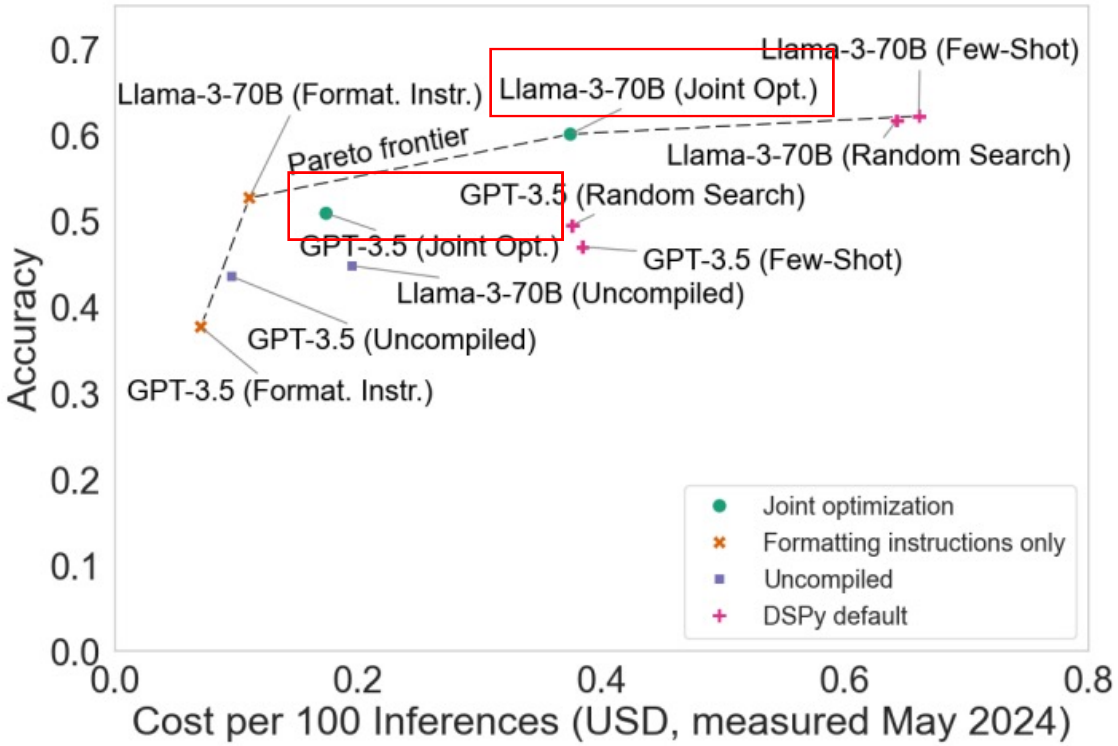


2. Jointly optimizing cost and accuracy can yield better agent designs

- **HotPotQA evaluation setup**
  - Model: Llama-3-70B and GPT-3.5
  - Data: 100 samples for training, 200 samples for evaluation
  - Agent architectures:
    - Uncompiled: question, context, reasoning
    - Formatting instructions only
    - Few-shot: example selection using DSPy
    - Random Search: using DSPy's random search optimizer
    - **Joint optimization**: Search temperature, # of few-shot example, selection of examples, formatting instruction using DSPy(Optuna)

2. Jointly optimizing cost and accuracy can yield better agent designs

- **HotPotQA results**
  - DSPy offers accuracy improvements over uncompiled
  - Joint optimization models are cheaper than the default DSPy implementation
  - Joint optimization allows for efficient agent design



3. Model and downstream developers have distinct benchmarking needs

- The difference between model evaluation and downstream evaluation is underappreciated
  - **Model evaluation**: scientific question of interest to researchers
  - **Downstream evaluation**: engineering question; cost is the actual construct of interest
- **Proxies for cost are misleading for downstream evaluation**
  - Mixtral 8x7B actually costs twice as much as Llama-2-13B
  - # parameter in API… 🤔
- **Addressing challenges to cost evaluation**
  - Making evaluation results customizable to adjust the cost of running models

3. Model and downstream developers have distinct benchmarking needs

- **Implications for benchmark design using a case study of NovelQA**
  - Actual users would ask questions about novels individually in practice
  - Table shows the misleading for downstream evaluation
  - Downstream evaluation benchmarks must be separate from model evaluation benchmarks

	RAG	Long-Context
Total Cost	\$52.80	\$99.80
Accuracy	67.89	67.81
RAG Specific:		
Cost of embedding 88 novels	\$2.512	-
Cost of embedding one novel	\$0.0285	-
Cost per Question	\$0.0222	-
Cost per QA for a new novel	**\$0.051**	-
Long-Context Specific:		
Mean prompt tokens per novel		690.807
Total tokens of questions and options		110,094
Total tokens (prompt + questions + options)		170885.016
Total long-context question cost		\$1.709
Long-context novel cost		\$98.09
Long-context cost per novel (single question)		**\$1.115**
Comparison:		
Cost Ratio (Long-Context/RAG)	≈ 21.86	

4. Agent benchmarks allow shortcuts

- Many agent benchmarks do not include held-out test sets
- Four levels of generality:
  1) Distribution-specific benchmarks
  2) Task-specific benchmarks
  3) Domain-general benchmarks
  4) General-purpose benchmarks

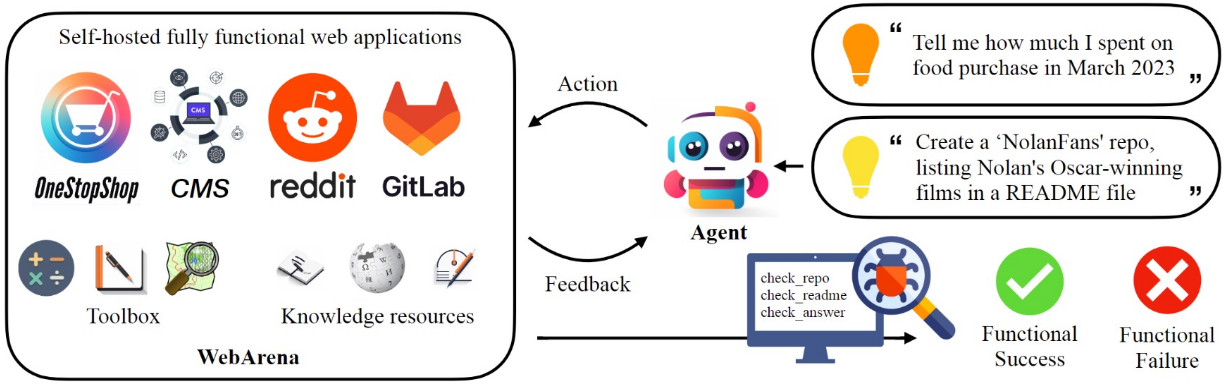
Level of generality	What should be held out	Num. benchmarks with appropriate holdouts
Distribution-specific	In-distribution samples	1 / 1
Task-specific	Out-of-distribution samples	3 / 6
Domain-general	Tasks	1 / 8
Fully general	Domains	0 / 2

Table 1: Appropriate holdouts based on level of generality. See Appendix for full details.

- Analysis of 17 agent benchmarks into the four levels of generality

Benchmark	Domain	Benc	Leve	Hold	Wha	Hold	Exa
MLAgentBench[19]	Programming	Measures the accuracy of agents specifically on machine learning experimentation.	Task-specific		N/A	Lacks a test set and doesn't indicate plans to make one.	Research tasks in languages other than Python.
SWE-Bench[59]	Programming	Measures the accuracy of agents specifically on solving software engineering problems. Authors intend to include repositories in the benchmark beyond the 12 initially sampled.	Task-specific	○	In distribution samples	The held-out set currently contains repositories not seen during training but are otherwise of a similar distribution as training. Authors mention plans to collect repositories in different programming languages, though not exclusively for the held-out set.	Repositories in languages other than Python.
WebArena[66]	Web task automation	Measures the accuracy of agents on many different web tasks.	Domain-general		N/A	Lacks a holdout set and doesn't indicate plans to make one.	New websites & tasks not seen during training, such as making plane or train travel bookings.

4. Agent benchmarks allow shortcuts



- **Case study of the STeP agent on WebArena**
  - WebArena’s core selling point: “Realism”
  - Top agent: STeP, 35.8% acc., 10% more than the next-best agent
- How does STeP achieve this high accuracy?
  - STeP hardcodes policies to solve the specific tasks included in WebArena

→ Is it useful agent to solve real-word tasks?

https://webarena.dev/

4. Agent benchmarks allow shortcuts

- Agent benchmarks don’t account for humans in the loop
  - Current evaluation focus on two extremes
    - Evaluating the capacity of chatbots to answer questions correctly (e.g., MMLU)
    - Whether agents can perform a task without supervision (e.g., agent benchmarks)
  - Human supervision, feedback, and intervention can be seen as a **spectrum**
  - The lack of human-in-the-loop evaluation of agents might lead underestimation of their usefulness

5. Inadequate benchmark standardization leads to irreproducible agent evaluations

- 5 root causes
    1. Evaluation scripts make assumptions about agent design that aren’t satisfied by all agents
    2. Repurposing LLM evaluation benchmarks for agent evaluation introduces inconsistencies
    3. The high cost of evaluating agents makes it is hard to estimate confidence intervals
    4. Agent evaluation relies on external factors such as interacting with an environment which can lead to subtle errors
    5. The lack of standardized evaluation leads to subtle bugs in agent evaluation and development

→ The need for a standardized evaluation framework!

Conclusion

- **AI agent benchmarking is new and best practice haven’t yet been established**
- Agents are sufficiently different from model
- Cost control, separating model and downstream evaluation, appropriate hold-outs, and standardization should be considered