Introduction to Agentic AI

-- Agentic AI Evaluation

Instructor: Guangjing Wang

guangjingwang@usf.edu

Last Lecture

- Jailbreaking Attack
- Vulnerabilities in Agentic Tool Usage
- Social Engineering Attack on AI Agent

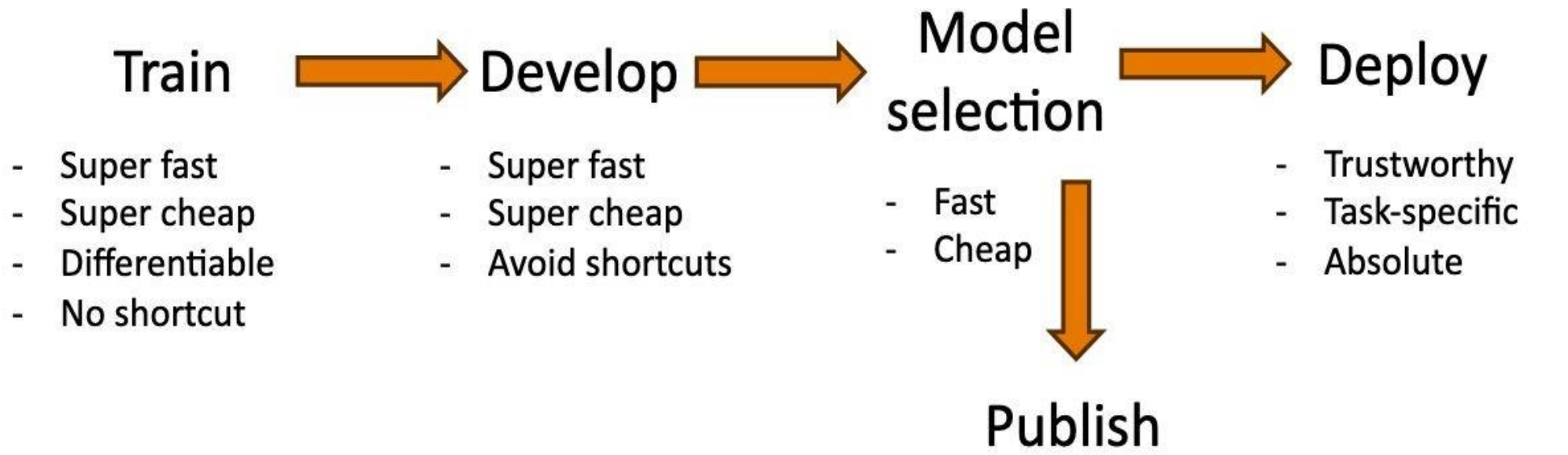
This Lecture

- Agentic AI Evaluation
- Human Evaluation and LLM as a Judge
- Benchmark Examples
- Agent Testing

What is Evaluation

- Evaluation is the systematic, repeatable measurement of models and agents.
- It provides a structured way to measure performance across benchmarks and environments.
- This helps
  - Measure capability progress that is grounded in reproducible evidence.
  - Risk assessment
  - It enables fair comparisons across models and agents.
  - Guides safe & effective deployment decisions by exposing weaknesses and strengths.

When Do We Need Evaluation



From LLM Eval to LLM Agent Eval

- LLMs are static, text-to-text systems.
- Agents extend LLMs with planning, tool-use, memory, and multi-step reasoning.
- Agents operate in dynamic environments, requiring more complex evaluation.
- Types of Evaluation
  - Close-ended: limited number of potential and correct answers.
  - Open-ended
    - Verifiable
    - Non-verifiable

Close-Ended VS Open-Ended

Close-ended evaluations

Example

Text: Read the book, forget the movie!

Label: Negative

Open ended evaluations

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Close-Ended Task

- Sentiment analysis: SST / IMDB / Yelp …

Example

Text: Read the book, forget the movie!
Label: Negative

- Entailment: SNLI

Example

Text: A soccer game with multiple males playing.
Hypothesis: Some men are playing sport.
Label: Entailment

- Name entity recognition: CoNLL-2003
- Part-of-Speech: PTB

Metrics:
- Accuracy
- Precision
- Reall
- F1
- …

Open-Ended Tasks

- Long generations with too many possible correct answers to enumerate
  - => can't use standard ML metrics
- There are now better and worse answers (not just right and wrong)
- Example:
  - Summarization: CNN-DM / Gigaword
  - Translation: WMT
  - Instruction-following: Chatbot Arena / AlpacaEval / MT-Bench

Verifiable Tasks and Non-Verifiable Tasks

Verifiable: Tasks that have an “oracle” or clear criteria to test if the result is correct or not.

- Math proof
- Code generation
- Only need to build an eval that follows the criteria

Non-Verifiable: Tasks without clear test criteria or objective ground-truth answer

- Storytelling
- Writing style adaptation
- Often need human eval or LLM-as-a-judge

Human Evaluations

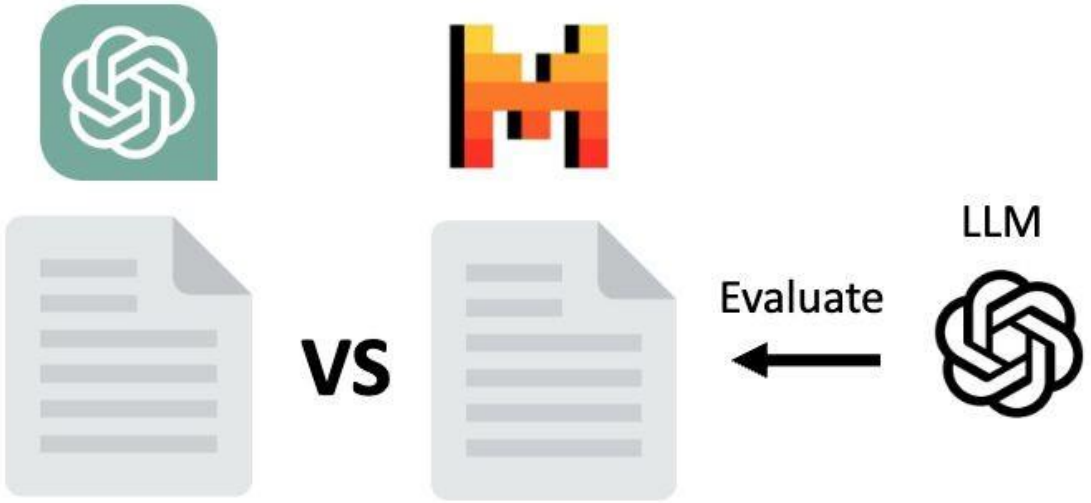
- Ask humans to evaluate the quality of the generated text

- Overall or along some specific dimension:
  - Fluency
  - Coherence / consistency
  - Factuality and correctness
  - Commonsense
  - Style / formality
  - Grammaticality
  - Redundancy

Human Evaluations: Issues

- Human evaluations are regarded as gold standards
- But it also has issues:
  - Slow
  - Expensive
  - Inter-annotator disagreement
  - Intra-annotator disagreement across time
  - Not reproducible

LLM as a Judge



- Use a LM as a reference free evaluator
- Surprisingly high correlations with human

- Common versions: AlpacaEval, MT-bench

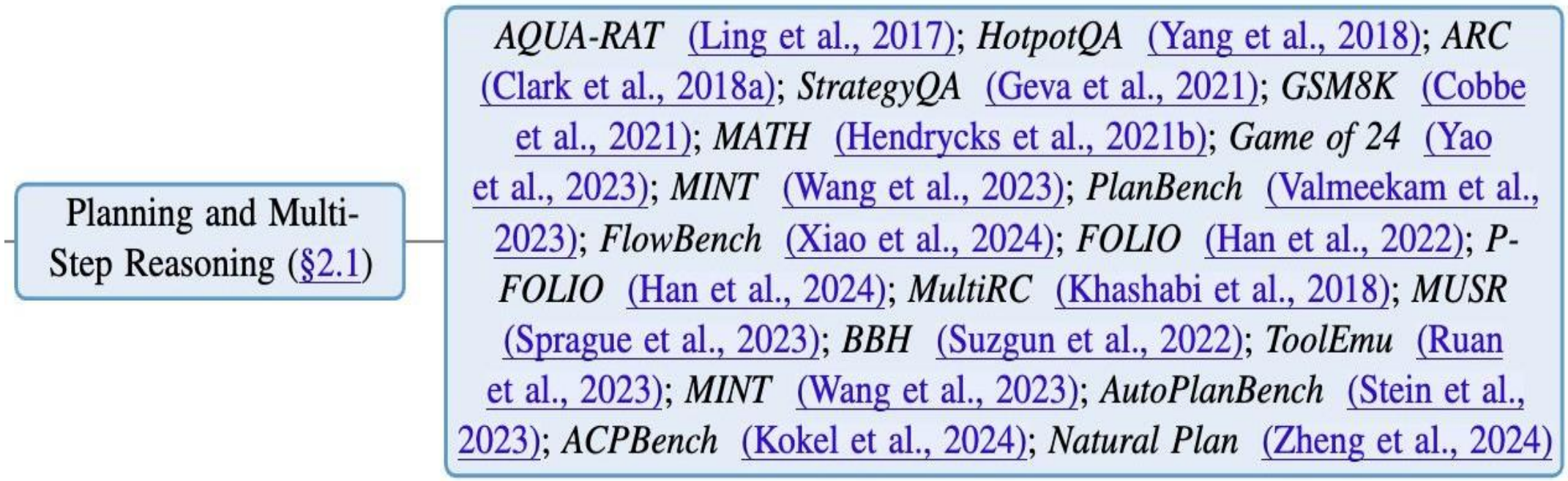
LLM as a Judge: Issues

- LLM sometimes is nor reliable
  - Cross-check with human eval
- LLM output has randomness
  - Repeat judging with the same LLM for multiple times
- Different LLMs have different biases
  - Majority vote between different LLMs
- Interpretability of scores is poor
  - Try using continuous instead of discrete scoring
  - Score from multiple perspectives / set multiple rubrics
- Sensitive to vague prompts
  - Try design as detailed prompts as possible
  - Use chain-of-thought prompt or reasoning mode to get better evaluation

Static VS. Dynamic Eval

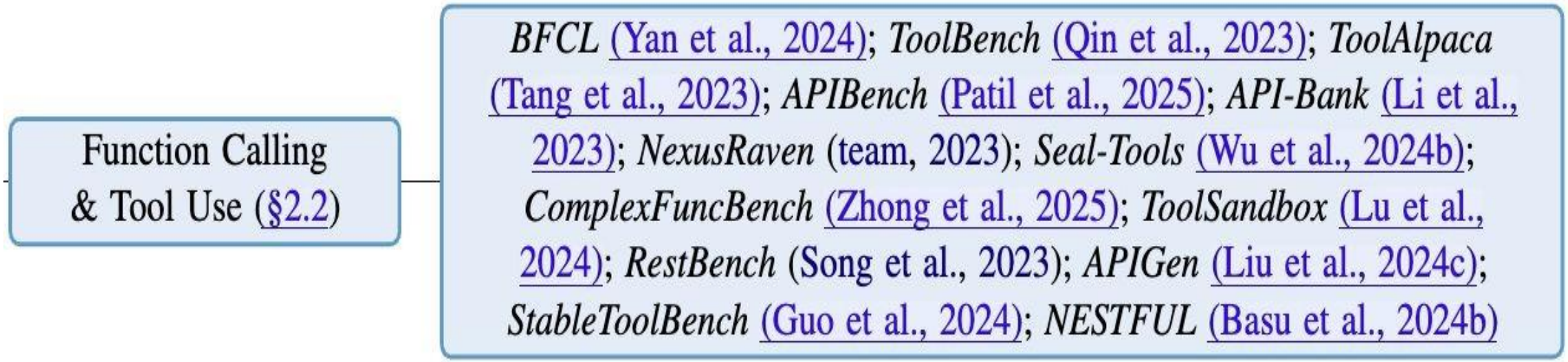
- Static benchmarks have **fixed** test cases and metrics
  - Enable direct, reproducible comparisons between models
  - Examples: ImageNet, GLUE, MMLU
- Dynamic benchmarks have continuously updated or periodically re-generated data
  - Stay relevant with real-world data shifts
  - Harder to overfit or be contaminated
  - Examples: DynaBench, LiveCodeBench

Evaluating a Specific Agent Capability



https://arxiv.org/pdf/2503.16416

Evaluating a Specific Agent Capability



https://arxiv.org/pdf/2503.16416

Evaluating a Specific Agent Capability

https://arxiv.org/pdf/2503.16416

Evaluating on A Specific Application

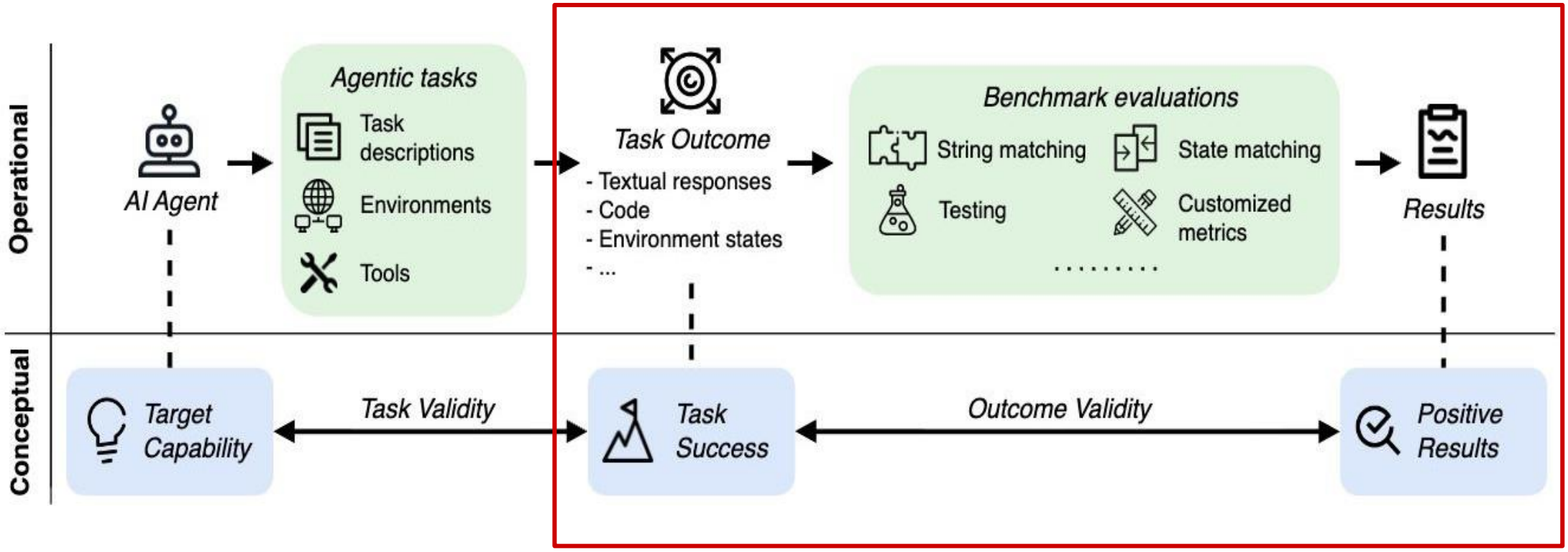
https://arxiv.org/pdf/2503.16416

Evaluating on A Specific Application

Scientific Agents (§3.3)	*ScienceQA* (Lu et al., 2022); *QASPER* (Dasigi et al., 2021); *MS*^{2} (DeYoung et al., 2021); *ScienceWorld*(Wang et al., 2022a); *SUPER* (Bogin et al., 2024); *Ideation* (Si et al., 2025); *AAAR-1.0* (Lou et al., 2025); *ScienceAgentBench* (Chen et al., 2024); *CORE-Bench* (Siegel et al., 2024); *SciCode* (Tian et al., 2024b); *MLGym-Bench* (Nathani et al., 2025); *DiscoveryWorld* (Jansen et al., 2024); *LAB-Bench* (Laurent et al., 2024)
Conversational Agents (§3.4)	*ABCD* (Chen et al., 2021a); *MultiWOZ* (Budzianowski et al., 2018); *SMCalFlow* (Andreas et al., 2020); *ALMITA* (Arcadinho et al., 2024); τ-*Bench* (Yao et al., 2024); *IntellAgent* (Levi and Kadar, 2025a); *LTM* (Castillo-Bolado et al., 2024b)

https://arxiv.org/pdf/2503.16416

Outcome Validity Makes a Good Eval



https://arxiv.org/pdf/2507.02825

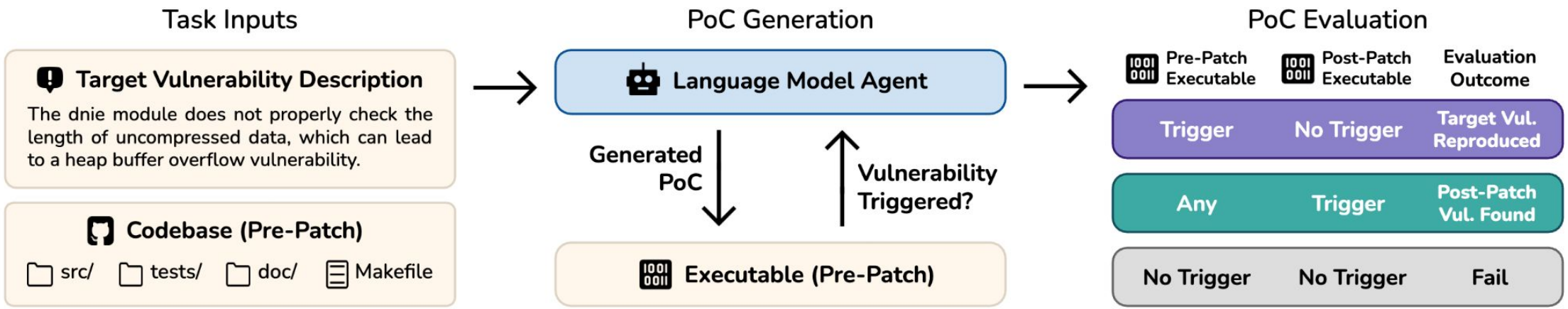
Ways That Eval Can Go Wrong

- Data is noisy or biased
  - Make sure the test data for evaluation is accurate and diverse enough!
- Not practical
  - Think about the practitioner's real needs!
- Shortcut - Eval can be gamed
  - Avoid any shortcut that your eval probably has!
- Not challenging enough
  - Design hard test cases to make sure your green agent is reliable!
- More info: https://arxiv.org/pdf/2502.06559v2

Evaluation Questions

- What is the goal / what to evaluate
- What is a task, what is an env to run the agent to achieve the goal, what's the size of the benchmark
- How to evaluate the agent (LLM Judge, etc.)
  - Need to generate data in the benchmark, talk about data pipeline
  - What metric to use to make benchmark high quality (contamination, bias)
- Principles: real-world, have different difficulty levels, not easy to get saturated.

CyberGym (https://www.cybergym.io/)



Goal: Evaluate an agent's cybersecurity capabilities by testing its ability to reproduce real-world vulnerabilities at a large scale

Task: Given a vulnerability description and the pre-patch codebase+executable, agents must generate a proof-of-concept (PoC) test that successfully triggers the vulnerability in the corresponding unpatched codebase

Env: a containerized sandbox to run programs

Zhun Wang*, Tianneng Shi* et al. CyberGym: Evaluating AI Agents' Cybersecurity Capabilities with Real-World Vulnerabilities at Scale. arXiv 2025

CyberGym (https://www.cybergym.io/)

Data Generation Pipeline:

- Built from ARVO dataset and historical, real-world vulnerabilities found by OSS-Fuzz, a continuous fuzzing project for open-source software
- reconstruct pre/post patch commits & executables and include the ground-truth PoC; rephrase into concise vuln descriptions with LLMs and manual inspection

How to Evaluate:

- Execute final PoC on pre-patch and post-patch builds. Count success if it (a) triggers the target vuln only **pre-patch (reproduction)**, or (b) triggers any vuln **post-patch (post-patch finding)**. Report overall success rate
- Detection is via runtime sanitizers (crash + stack trace), not subjective judging.
- A **data contamination analysis** is performed by evaluating vuln samples found after LLM knowledge cutoff dates

GDPEval (https://openai.com/index/gdpval/)

Goal: Measure LLM performance on economically valuable, real-world knowledge-work tasks, comparing AI deliverables to industry experts across diverse occupations

Task and Env: Each task is a realistic work assignment with reference files/context (docs, data, assets). Models produce a one-shot deliverable (e.g., doc, slide deck, spreadsheet, diagram, media)

GDPEval (https://openai.com/index/gdpval/)

Data Generation Pipeline: Tasks authored by vetted professionals (avg 14 yrs experience), pass a multi-step review (≈5 rounds) plus model-based validation; prompts mirror day-to-day work and include attachments; gold deliverables are experts' own solutions

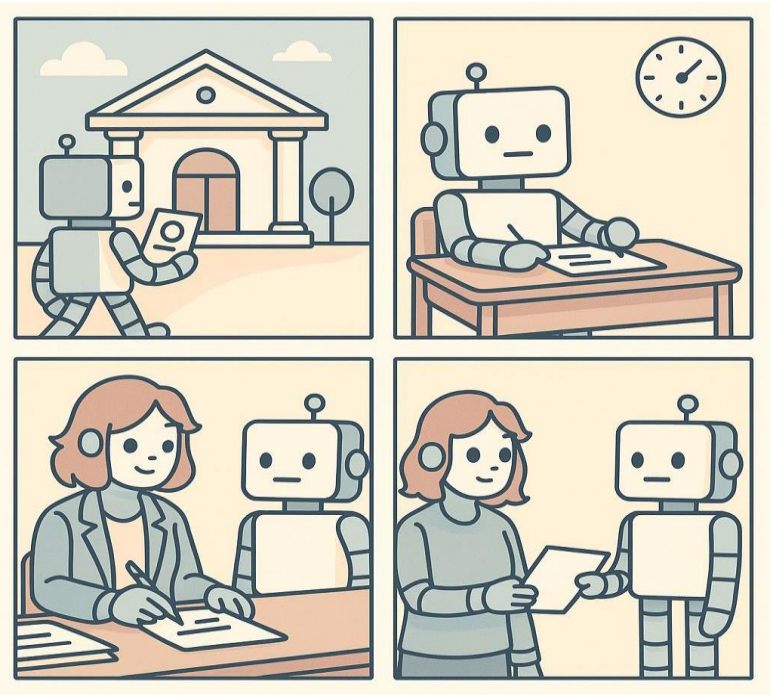
How to Evaluate:

- Blinded expert graders from the same occupations rank AI vs. human deliverables as better / as good as / worse
- Also compare time/cost
- Good example of **a benchmark with low contamination risk and hard to get saturated** as tasks require domain experts and tied to concrete work product

Testing My Agent

Goal: Measure LLM performance on economically valuable, real-world knowledge-work tasks, comparing AI deliverables to industry experts across diverse occupations

Task and Env: Each task is a realistic work assignment with reference files/context (docs, data, assets). Models produce a one-shot deliverable (e.g., doc, slide deck, spreadsheet, diagram, media)



Responsibilities of the Green Agent

- Green agents are specifically designed to serve as evaluators
- Green agents can interact with the platform through **MCP, A2A** or **APIs**, request permissions and resources, and submit results.
- The green agent carries multiple responsibilities:
  - Preparing the environment
  - Distributing test tasks to participant agents
  - Collecting their results
  - Verifying the environment
  - Reporting back to the platform

The Full Assessment Flow

1. The platform first confirms that all agents, including the green agent, are online and reset.
2. It then sends a task to the green agent, including the URLs of the participant agents to be tested.
3. The green agent orchestrates the interaction: assigning tasks, supervising execution, managing tools or environments, and continuously reporting updates back to the platform.
4. At the end, the green agent submits the metrics—which it defines through its implementation.

How Green Agents Are Built

- Typically, a green agent includes:
  - A dataset of test tasks
  - A predefined testing process (e.g., which agent to test first, in what order tasks are sent, and how tools are provided)
  - The environment where the tasks run: access to additional tools or MCP modules required for the tasks, along with instructions for the white agents on how to use them
- AgentBeats provides a **prompt-based toolkit** to help developers quickly spin up a prototype green agent, making it easy to get started.

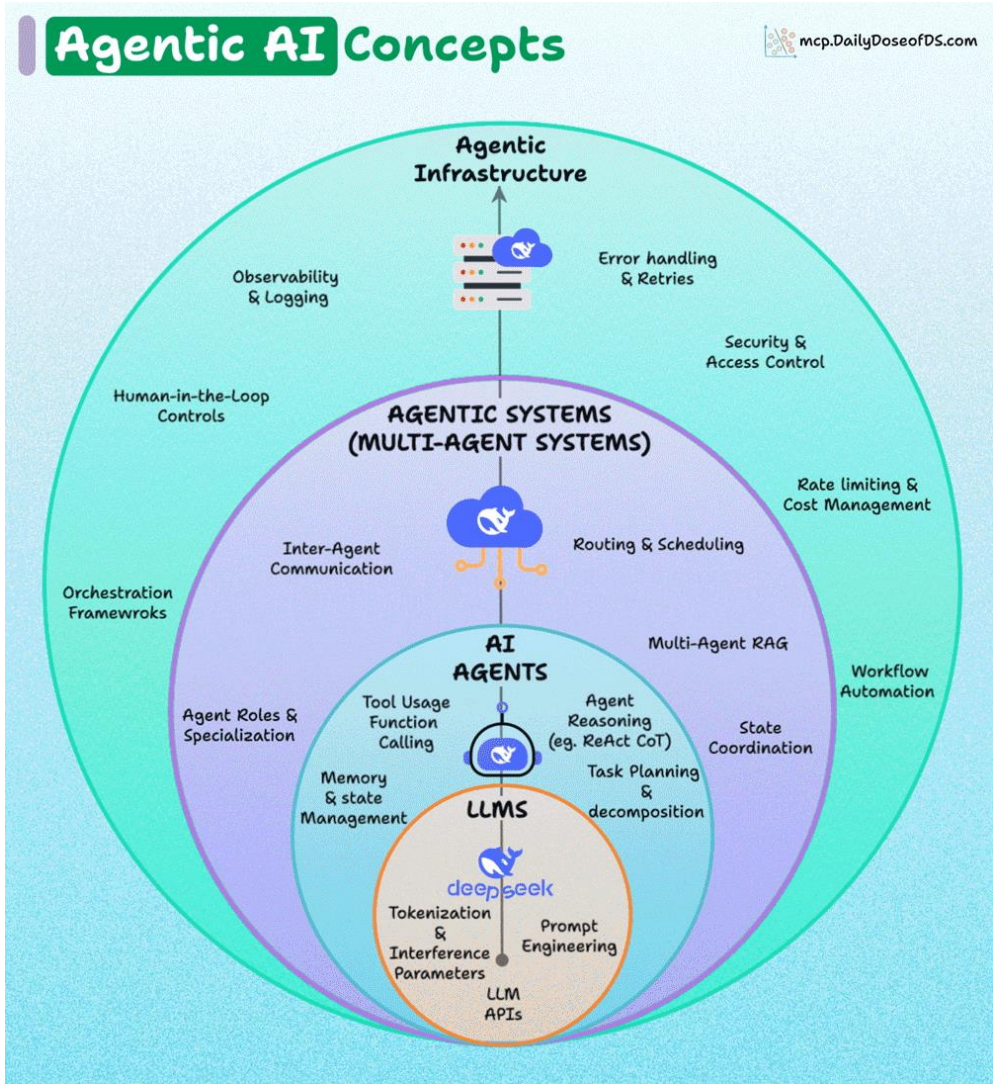
AgentBeats (https://agentbeats.dev/)

https://www.youtube.com/watch?v=QKpYqo3yDVs

Multimodal Autonomous AI Agent

Other Topics in Agentic AI

- MOOC:
  - https://rdi.berkeley.edu/agentic-ai/f25
  - https://agenticai-learning.org/sp25
  - https://agenticai-learning.org/f24



References

- https://rdi.berkeley.edu/agentic-ai/f25 (Oct. 6)
- https://arxiv.org/pdf/2503.16416