Agent Evaluation

Week 10: Benchmarks, Metrics, and Assessment

PhD Course in Agentic Artificial Intelligence

12-Week Research-Level Course

Learning Objectives

Bloom's Taxonomy Levels Covered

- **Remember**: Define AgentBench, SWE-bench (software engineering), GAIA, LLM-as-Judge
- **Understand**: Explain why agent evaluation differs from LLM evaluation
- **Apply**: Run agents against standard benchmarks and interpret results
- **Analyze**: Compare agent performance across different dimensions
- **Evaluate**: Assess reliability and validity of different evaluation methods
- **Create**: Design custom evaluation protocols for novel agent applications

---

By end of lecture, you will understand how to rigorously evaluate agent systems.

Why Agent Evaluation is Hard

Beyond Single-Turn Accuracy

- LLM evaluation: Measure output quality on single prompts
- Agent evaluation: Measure multi-step task completion in environments

Key Challenges

- **Trajectory dependence**: Many valid paths to same goal
- **Partial credit**: How to score incomplete solutions?
- **Environment variance**: Results depend on environment state
- **Cost**: Each evaluation run costs time and API calls

What to Measure

- Success rate, efficiency, safety, robustness, cost
- Single metrics hide important trade-offs

Agent evaluation requires thinking about process, not just outcomes.

Agent Benchmark Landscape

Agent Benchmark Landscape

AgentBench
8 environments
OS, DB, Web
Multi-step tasks

WebArena
Web browsing
E-commerce
Realistic sites

SWE-bench
Code tasks
GitHub issues
Real repos

GAIA
Real-world QA
Multi-modal
3 difficulty levels

ToolBench
16k+ APIs
Tool selection
Multi-tool

OSWorld
Desktop OS
GUI tasks
Screenshots

Each benchmark tests different agent capabilities and environments.

Major Agent Benchmarks

AgentBench (Liu et al., 2023)

- 8 environments: OS, database, knowledge graph, web, games, etc.
- Measures general-purpose agent capability across domains

SWE-bench (Jimenez et al., 2024)

- Real GitHub issues from popular Python repos
- Task: Generate code patch to resolve issue
- Gold standard for code agent evaluation

WebArena (Zhou et al., 2024)

- Realistic web environments (shopping, forums, maps)
- Tests navigation, form-filling, multi-step web tasks

GAIA (Mialon et al., 2024)

- General AI Assistant benchmark
- Multi-modal, multi-step reasoning tasks

Choose benchmarks based on your agent's intended domain.

Evaluation Dimensions

Primary Metrics

- **Success Rate**: Task completed correctly (binary or graded)
- **Pass@k**: Success rate with k attempts allowed
- **Efficiency**: Steps/tokens/time to complete task

Secondary Metrics

- **Safety**: Avoids harmful actions, respects constraints
- **Robustness**: Performance under perturbations
- **Cost**: API calls, compute, latency

Human-Centric Metrics

- User satisfaction, trust calibration, explainability
- Often require human evaluation studies

Multi-dimensional evaluation reveals trade-offs hidden by single metrics.

Pass@k and Statistical Validity

Pass@k Metric

- Pass@1: Success on first attempt
- Pass@k: Success within k attempts
- Captures both capability and consistency

Calculation (Unbiased Estimator)

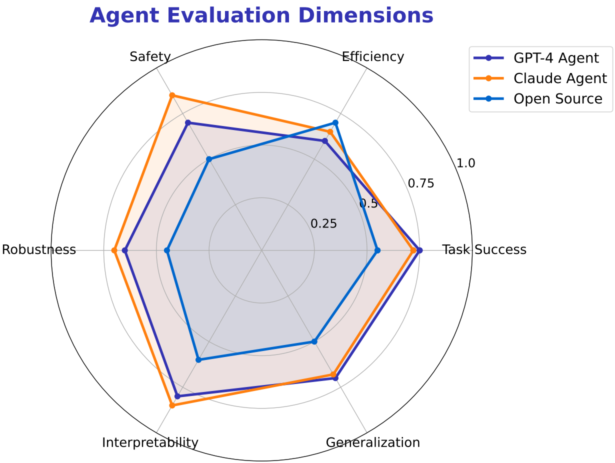
$$\text{Pass@k} = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

where n = samples, c = correct samples **Best Practices**

- Run at least 3-5 trials per task
- Use identical seeds for fair comparison
- Report both pass@1 and pass@5
- Include confidence intervals (variance can be 5-10%)

High pass@5 but low pass@1 indicates inconsistency – reliability matters.

Evaluation Dimensions



Comprehensive evaluation requires multiple dimensions beyond accuracy.

AgentBench Results Analysis

Key Findings (Liu et al., 2023)

- GPT-4 significantly outperforms other models (4.41/10 overall score)
- Open-source models struggle (0.5-2.0 on most environments)
- Performance varies dramatically across environments (5-78% by task)

Performance by Environment

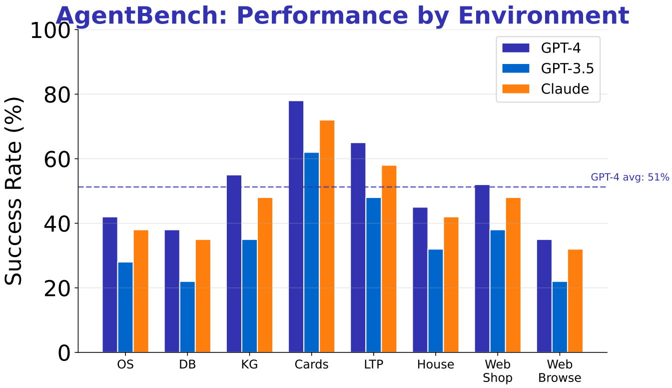
- Web browsing: 15-20% (relatively well-structured)
- Operating system: 5-10% (complex state management)
- Database: 8-12% (requires precise SQL)

Implications

- Current agents far from human-level on complex tasks
- Environment-specific optimization needed

Even best models fail on majority of complex multi-step tasks.

AgentBench Results



Performance varies significantly across environments and models.

Evaluation Methods

Automated Evaluation

- **Exact match**: Output matches expected answer
- **Unit tests**: Code passes test suite (SWE-bench)
- **Reward model**: Trained model scores outputs

LLM-as-Judge

- Use LLM to evaluate agent outputs (e.g., GPT-4 as grader)
- Flexible, handles natural language outputs
- Concerns: Bias, self-preference, cost

Human Evaluation

- Gold standard for subjective quality
- Expensive, slow, hard to scale
- Use for validation, not primary development loop

Automated metrics enable rapid iteration; human eval validates final quality.

Cost-Aware Evaluation

Why Cost Matters

- Higher accuracy often requires more tokens/API calls
- Production systems have budget constraints
- Fair comparison requires cost normalization

Cost Dimensions

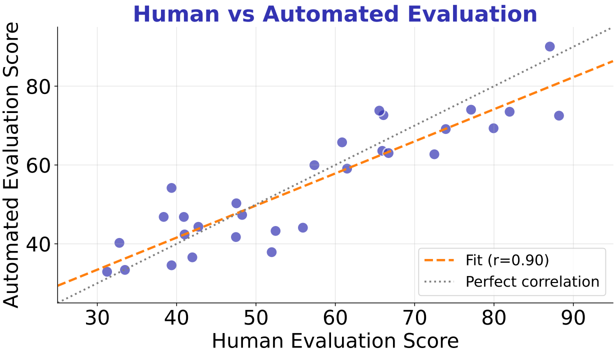
- Token count (input + output)
- Number of LLM calls
- Tool/API invocations
- Wall-clock time (latency)

Cost-Adjusted Metrics

- Accuracy per dollar spent
- Success rate at fixed budget
- Pareto frontier (accuracy vs. cost trade-off curve)

Best agent $\neq$ highest accuracy; best agent = best accuracy at acceptable cost.

Human vs Automated Evaluation



Automated metrics correlate with human judgment but not perfectly.

LLM-as-Judge: Practical Considerations

Benefits

- Scales to large evaluation sets
- Handles open-ended, natural language outputs
- No need for exact match or formal specification

Limitations

- **Position bias**: Prefers first option in comparisons
- **Self-preference**: GPT-4 rates GPT-4 outputs higher
- **Verbosity bias**: Longer responses rated higher
- **Cost**: Expensive for large-scale evaluation

Best Practices

- Use structured rubrics with explicit criteria
- Randomize order in pairwise comparisons
- Validate against human judgments on subset

LLM-as-Judge is powerful but requires careful protocol design.

Designing Custom Evaluations

Step 1: Define Success Criteria

- What does "task completed" mean?
- Binary success or graded scoring?

Step 2: Create Representative Tasks

- Cover range of difficulty and edge cases
- Include realistic failure modes

Step 3: Choose Evaluation Method

- Automated where possible, human for validation

Step 4: Establish Baselines

- Compare to: random baseline (lower bound), human baseline (upper bound), prior models (reference model)

Step 5: Report Confidence Intervals

- Multiple runs, statistical significance testing

Good evaluation = right benchmark + right metrics + right baseline.

Required Readings

This Week

- Liu et al. (2023). "AgentBench: Evaluating LLMs as Agents." arXiv:2308.03688
- Zhou et al. (2024). "WebArena: A Realistic Web Environment for Building Autonomous Agents." arXiv:2307.13854
- Mialon et al. (2024). "GAIA: A Benchmark for General AI Assistants." arXiv:2311.12983

Supplementary

- Jimenez et al. (2024). "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" arXiv:2310.06770
- Zheng et al. (2023). "Judging LLM-as-a-Judge with MT-Bench." arXiv:2306.05685

AgentBench provides the most comprehensive multi-environment evaluation.

Summary and Key Takeaways

Key Concepts

- **Benchmarks**: AgentBench, WebArena, SWE-bench, GAIA
- **Dimensions**: Success, efficiency, safety, robustness, cost
- **Methods**: Automated metrics, LLM-as-Judge, human evaluation

Design Principles

- Multi-dimensional evaluation reveals hidden trade-offs
- Combine automated and human evaluation
- Always compare against meaningful baselines

Next Week

- Domain Applications: Code and Finance Agents

Good evaluation = right benchmark + right metrics + right baseline.