

Beyond Task Success: An Eight-Metric Tiered Evaluation Protocol for LLM Agents over Fragmented Operational Data

Neeraj Baji

Amazon

Seattle, WA

nbaji@amazon.com

Abstract

Single-metric task-success scores systematically misrank LLM agents in domain-specific deployments: two configurations can achieve identical end-to-end accuracy while exhibiting structurally different production behaviors — silent argument hallucination, infinite tool loops, redundant retries, semantically-equivalent-but-syntactically-divergent queries. We propose a behavioral evaluation protocol that decomposes agent quality into an aligned eight-dimensional score vector across three layers: *answer* (Accuracy, Semantic F1, Cosine Sim), *trajectory* (Set Match, Order Match, Args Match, Tool Coverage), and *execution* (Avg Failures/Traj). The protocol’s diagnostic value comes not from any single metric but from the *joint reading* of metrics across layers — specific gap patterns localize to specific behavioral failure modes (resilient self-correction, compliant-but-fragile planning, formatting-only failures, tool-misrouting, execution-environment failures). We instantiate the protocol on five reasoning configurations (zero-shot and DSPy-Simba-optimized variants of Chain-of-Thought and ReAct, plus a CodeAgent baseline) over the ToolHop multi-hop tool-use benchmark and over a domain-expert-curated production set of 50 operational queries from a real enterprise deployment. Optimized ReAct attains 65% on ToolHop and 86% on the production set; the eight-metric breakdown reveals that DSPy optimization patches CoT’s syntactic compliance without unlocking adaptive reasoning, exposes a structural “glass ceiling” for static planners, and surfaces a *Resilient* behavioral signature (high Set Match, low Order Match, non-zero declining Avg Failures/Traj) that transfers from synthetic benchmarks to domain-specific queries. The protocol is implemented as a stateful metric callable plus a trace-extraction harness that decouples measurement from deployment, enabling retroactive re-scoring of logged traffic against evolving metric definitions; the present paper describes both in sufficient detail for independent reimplementations.

1 Introduction

Standard practice in LLM-agent research reports a single end-to-end task-success score (often Exact-Match accuracy on a held-out set), supplemented by anecdotal qualitative analysis. We argue this is not just incomplete but *misleading* for domain-specific deployments where agents must operate over fragmented data with non-deterministic intermediate states. A scalar success score conflates several distinct behavioral failure modes — planning errors, tool-argument hallucination, syntactic non-compliance with tool schemas, intermediate-state blindness, formatting drift in final answers — and is therefore a poor optimization target and a poor production-readiness signal.

This paper makes three contributions:

1. A **tiered behavioral evaluation protocol** that decomposes agent quality into an aligned eight-dimensional score vector across answer, trajectory, and execution layers (Section 4).

2. A **library of canonical behavioral patterns** (Table 1) — *Resilient, Compliant-but-fragile, Typically right, formally wrong, Tool-misrouting, Execution-environment failure* — each defined by a specific gap pattern across the eight metrics. Single-metric protocols obscure all five.
3. **Empirical validation** on (a) the ToolHop multi-hop tool-use benchmark and (b) a domain-expert-curated production set of 50 operational queries from a real enterprise deployment whose users span the full operational organization at the deployment site — an organization responsible for a global multi-site fulfillment operation. We show that DSPy-Simba-optimized ReAct attains 65% on ToolHop and 86% on the production set; that DSPy optimization patches Chain-of-Thought’s syntactic compliance without unlocking adaptive reasoning (a finding invisible under accuracy-only evaluation); and that the *Resilient* behavioral signature transfers from the synthetic benchmark to domain-specific queries. The metric tracker and the trace-extraction harness that decouples measurement from deployment are described in detail; both are not publicly released due to the proprietary nature of the deployment data, but are specified with enough detail for independent reimplementations.

2 Related work

Tool-use and agent benchmarks. Recent work has produced a wave of agent benchmarks emphasizing realistic tool use and end-to-end task completion: τ -bench (Yao et al., 2024) for tool-augmented dialogue, AgentBench (Liu et al., 2023) as a multi-task suite, SWE-bench (Jimenez et al., 2024) for software engineering, and WebArena (Zhou et al., 2024) for web navigation. These benchmarks have been instrumental in driving capability progress, but each ultimately collapses agent quality to a single end-task success rate (or a small set of binary task indicators) — the failure mode our protocol is designed to address. ToolHop (Ye et al., 2025), which we use as our synthetic substrate, is multi-hop and tool-rich but similarly reports task success as the primary number.

Production-evaluation methodology. Pan et al. (2025) argue that non-deterministic production agent tasks fundamentally require human-in-the-loop evaluation and that synthetic scoring under-predicts deployment quality. Our work agrees with this premise but takes a complementary stance: rather than relying on HITL grading alone, we show that *multi-dimensional automated scoring* — when paired with expert-curated reference trajectories — can recover much of the diagnostic signal HITL provides while remaining cheap to apply at every release. AgentBoard (Ma et al., 2024) proposed multi-dimensional evaluation across capabilities, the closest methodological precedent; we differ by (i) coupling answer-, trajectory-, and execution-layer signals into a unified eight-metric report, (ii) defining a behavioral pattern library that maps gap structures to behaviors (Table 1), and (iii) validating in production rather than benchmark-only.

Process-vs-outcome scoring. Process reward models (Lightman et al., 2024) supervise intermediate reasoning steps rather than only final answers. Our trajectory-layer metrics (Set Match, Order Match, Args Match, Tool Coverage) extend this idea from chain-of-thought reasoning to tool-using agents, where the process is observable as a structured sequence of (tool, args, observation) tuples rather than free-form text.

Tool-use lineage. Our protocol applies to any agent whose action space is a tool inventory; the underlying reasoning paradigm is ReAct (Yao et al., 2023) augmented with prompt optimization (DSPy Simba (Khatab et al., 2024)). Toolformer (Schick et al., 2023) and Reflexion (Shinn et al., 2023) are foundational antecedents; Reflexion in particular relates to our *Resilient* behavioral pattern, where self-correction over failed tool calls is a feature rather than a bug.

3 Background and motivating deployment

3.1 The deployment context, in generic terms

The motivating deployment is an internal natural-language analytics agent serving an enterprise organization whose operational evidence is distributed across heterogeneous, semantically-overlapping data stores: relational tables (incident tickets, planned-change windows), document indices (runbooks, postmortems, case histories), and time-series stores (site telemetry). No unified data lake exists; data dictionaries are sparse beyond raw table schemas. Recurring high-context questions surfaced in operational triage workflows — “which sites are repeat offenders for this failure mode?”, “did last quarter’s planned-change windows correlate with the spike in high-severity tickets?”, “what historical postmortems share root causes with the current outage?” — cannot be answered by static dashboards.

Prior to this work, the team’s deployed assistant was a Retrieval-Augmented Generation (RAG) chatbot (Lewis et al., 2020) scoped to wiki retrieval and onboarding lookups: it could answer *what is X* but not *why is X happening across our infrastructure*. The work reported here re-architected that assistant as a tool-using agent that performs multi-hop reasoning over the fragmented data layer. We omit organization-specific framing because the contribution is methodological — the behavioral evaluation protocol applies to any agent over comparable data fragmentation.

3.2 Why static planning fails on this class of task

Before adopting fully agentic reasoning, the team evaluated a deterministic control flow driven by semantic routing: a router classified user intents and triggered hardcoded tool sequences. This approach failed to scale — as query diversity grew, the required trajectories exploded combinatorially and the maintenance burden of hardcoding arguments and managing tool sprawl became intractable. Deterministic routing suits narrow, low-variance domains; open-ended analytics over fragmented data necessitates dynamic reasoning. Our hypothesis is that the gap between static planning (Chain-of-Thought (Wei et al., 2022)) and dynamic reasoning (ReAct (Yao et al., 2023)) in such settings is *structural*: it stems from the agent’s inability to observe intermediate data states under CoT, and persists even when the static planner is aggressively optimized.

3.3 Agent and reasoning formalization

We formalize the task as sequential decision-making over a semi-structured environment. Given a natural language query x and a database schema $\mathcal{D}$, the agent generates a final answer y via an adaptive trajectory $\tau = (z_1, a_1, o_1, \dots, a_T, o_T)$ of interleaved thoughts z_t , tool actions a_t , and observations o_t , where each a_t is either a tool invocation or a Finish(y) action. At each step, the policy π_θ conditions on the user query and execution history: $a_t \sim \pi_\theta(a_t \mid x, h_{t-1})$. DSPy optimization keeps model weights θ frozen and instead optimizes the contextual prefix ϕ (system instructions and few-shot exemplars) to maximize the expected metric over the query distribution.

Tools are exposed via a standardized invocation protocol and split by data modality: SQL tools auto-inject DDL context to ground generation; document-index tools expose a clean abstraction over verbose JSON DSLs (Xue et al., 2025). A semantic router classifies incoming queries into pass-through chat, documentation RAG, or complex analytical reasoning, dispatching the latter to the dynamic ReAct loop. Every production query is logged as a structured trace — the substrate for the trace harness in Section 4.3.

4 Tiered behavioral evaluation framework

Thesis. For a domain-specific agent operating over fragmented data, a single end-to-end correctness metric is necessary but not sufficient. The diagnostic value of an evaluation protocol comes from its ability to localize *which* aspect of agent behavior is responsible for an observed failure — planning, tool selection, argument binding, output formatting, or

execution-environment instability — so that the next remediation investment is correctly targeted.

4.1 Eight aligned metrics across three layers

We instantiate the protocol as a stateful metric tracker that, after evaluating an agent on a test set, emits an eight-dimensional score vector aligned across the same prediction-vs-gold pairs (different facets of the *same* trajectory, not different runs). Diagnostic value comes from the *gap* between metrics in different layers.

Answer Layer (3 metrics). Accuracy is strict exact-match after normalization (lower-cased, comma-stripped, trailing-zero-stripped) — the metric most agent papers report. Semantic F1 is `dspy.evaluate.SemanticF1(decompositional=True)` (Khattab et al., 2024), which prompts an LLM-based ChainOfThought module to enumerate key ideas in gold and prediction, then computes precision/recall over their overlap. Cosine Sim is sentence-transformer (all-mpnet-base-v2) cosine similarity (Reimers and Gurevych, 2019; Song et al., 2020), capturing topical alignment even when no claim is exactly preserved. The progression Accuracy $\rightarrow$ Semantic F1 $\rightarrow$ Cosine Sim localizes failure mode: high Cosine Sim with low Accuracy indicates a formatting/normalization failure; low Cosine Sim indicates the agent answered a different question entirely.

Trajectory Layer (4 metrics). Given a reference trajectory τ^* authored by a domain expert, `Traj. Set Match %` is the fraction of items where the set of tools invoked equals the gold set (“did the agent touch the right data sources, regardless of order?”); `Traj. Order Match %` is the fraction where the predicted tool sequence equals the gold sequence (plan compliance); `Traj. Args Match %` is the fraction where the full (*tool, args*) sequence matches gold (planning plus argument-binding fidelity); and `Tool Coverage %` is the recall of gold tools in the predicted trajectory ($|gold \cap pred|/|gold|$), awarding partial credit when the gold trajectory is non-unique. The progression Set $\rightarrow$ Order $\rightarrow$ Args represents tightening compliance: a sharp drop from Set to Order indicates an order-flexible agent; a sharp drop from Order to Args indicates argument-layer failures even when the agent picked the right tools in the right order.

Execution Layer (1 metric). `Avg Failures/Traj` is the mean count of tool-invocation errors per trajectory (timeouts, raised exceptions, unknown-tool calls). It is distinct from outcome metrics: a trajectory can complete with the right answer despite many transient tool failures (the hallmark of a *resilient* agent) or fail outright with zero invocation errors (*competent-but-misdirected* planning). Plotting `Avg Failures/Traj` against Accuracy separates planning failure (low/low) from execution-environment failure (low/high); they demand different remediation paths.

4.2 Behavioral patterns surfaced by the report

The eight-metric report is interpretable through a small library of canonical behavioral patterns (Table 1).

Collapsing the eight metrics to a scalar obscures all five patterns; the protocol’s diagnostic value lies in the *joint reading* across layers, not in any single metric.

4.3 Format-agnostic extraction and trace harness

A single tracker instance scores trajectories from structurally different agents. A `trajectory_format` flag selects between (i) list-of-dicts trajectories — used by the custom ReAct agent and the ToolHop reference format — where each step has a `selected_fn` or `tool_name`, an `args` dict, and an optional error field; and (ii) DSPy ReAct’s flat-dict trajectories, where steps are encoded as numbered keys (`tool_name_0`, `tool_args_0`, `error_0`, ...). This format-agnostic design lets the same eight-metric report compare a CodeAgent baseline, ChainOfThought-based planners (list format), and DSPy ReAct (flat-dict format) on equal footing.

Table 1: Behavioral patterns surfaced by joint reading of the eight metrics.

Pattern	Signature	Behavioral interpretation
<i>Resilient</i>	high Accuracy, high Set Match, low Order Match, non-zero Avg Failures	Agent self-corrects through trial and error (e.g. ReAct). Failures are diagnostic, not blocking.
<i>Compliant-but-fragile</i>	low Accuracy, high Order Match, low Args Match, low Avg Failures	Static plan executes cleanly but binds wrong arguments (e.g. optimized CoT). Optimization patches arguments without unlocking adaptive reasoning.
<i>Topically right, formally wrong</i>	low Accuracy, high Cosine Sim, high Semantic F1	Normalization/formatting issue; underlying reasoning is correct.
<i>Tool-misrouting</i>	low Set Match, low Tool Coverage	Wrong tool entirely; semantic-router or tool-description failure rather than reasoning failure.
<i>Execution-environment</i>	low Accuracy, high Avg Failures, normal Set Match	Right plan, broken tools — usually credentials, timeouts, or DSL non-determinism.

A complementary trace-extraction harness pulls live trajectories from an experiment-tracking server filtered by run tag, normalizes relative date references in production questions (“tickets in the last 7 days”) to absolute ranges grounded on the trace’s request timestamp, and emits flat JSON records suitable for re-scoring. Together they decouple measurement from deployment: any logged production trace can be re-scored against new metric definitions, gold-trajectory revisions, or refined test sets *without re-running the agent*. This supports per-metric regression detection on every release: a drop in Traj. Order Match at constant Accuracy is a different alarm than a drop in Accuracy itself; a rise in Avg Failures/Traj at constant Accuracy flags execution-environment instability rather than reasoning regression.

5 Experimental design

5.1 Configurations

We compare five configurations to isolate the contribution of reasoning strategy from optimization framework: an external `smolagents.CodeAgent` (Wang et al., 2024) baseline; **Config A**, zero-shot `dspy.ChainOfThought` (static planning floor); **Config B**, `dspy.ChainOfThought` optimized via `DSPy Simba`; **Config C**, zero-shot `dspy.ReAct`; and **Config D** (proposed), `dspy.ReAct` optimized via `DSPy Simba`.

5.2 Datasets and protocol

The ToolHop dataset (Ye et al., 2025) serves as the synthetic proxy for multi-hop tool reasoning, with a 200/50/100 train/val/test split; optimization uses `dspy.Simba` (Exact-Match metric, $k = 16$), temperature 0, three seeds. For domain-specific evaluation, we partnered with domain experts at the deployment site to curate **50 high-priority operational questions** drawn directly from real operational triage workflows — covering postmortem root-cause correlation, planned-change-window incident attribution, ticket-severity trend analysis, and site-specific repeat-offender identification. Ground-truth answers and reference tool trajectories were authored by domain experts through structured review.

6 Results

6.1 ToolHop benchmark

Table 2 reports the comparative results in the eight-metric format. Headline Accuracy is shown numerically; secondary metrics are summarized as directional patterns relative to

the same metric in the Zero-Shot CoT baseline (the precise numerical fill is reproducible by reimplementing the protocol per Section 4 against the public ToolHop split with the published seeds).

Table 2: ToolHop benchmark: eight-metric report. Headline Accuracy bolded. Trajectory- and execution-layer columns abbreviated (Set/Ord/Args/Cov for Trajectory Set/Order/Args Match %, Tool Coverage %; Fail for Avg Failures/Traj).

Configuration	Acc.	Sem. F1	Cos. Sim	Set %	Ord %	Args %	Cov. %	Fail
External Baseline (CodeAgent)	28.0%	—	—	—	—	—	—	—
Config A — Zero-Shot CoT	2.67%	low	mid	71.0%	67.3%	low	mid	low
Config B — Optimized CoT	23.0%	mid	mid-hi	72.0%	70.0%	low-mid	mid	low
Config C — Zero-Shot ReAct	35.0%	mid-hi	high	38.0%	22.7%	low	high	non-zero
Config D — Optimized ReAct	65.0%	high	high	high	low (adapt.)	mid	high	non-zero, ↓

Configs B and D additionally report standard deviation over three seeds (B: $\pm 0.5\%$; D: $\pm 0.9\%$; A: $\pm 1.2\%$; C: $\pm 1.7\%$). Qualitative descriptors (low / mid / high / declining) are read relative to the same metric in Config A; they make the behavioral patterns of Table 1 legible at a glance without substituting for the numerical report.

6.2 Findings on ToolHop

Optimization fixes syntax, not structure. The Simba optimizer improved CoT Accuracy nearly $10\times$ ($2.67\% \rightarrow 23.0\%$), but Traj. Set Match % ticked up only 1% ($71\% \rightarrow 72\%$). DSPy optimization taught the agent *syntactic compliance* (tool-argument formatting) without fixing its *semantic planning* flaws. **This insight is invisible under a single-metric protocol** — Accuracy alone would suggest optimization “fixed” CoT, when in fact it only patched the leaf-level argument layer (visible only when paired with the trajectory-layer metrics).

The glass ceiling of static planning. Optimized CoT (23.0%) was outperformed by completely unoptimized zero-shot ReAct (35.0%). Static planners are structurally blind to intermediate execution states and hallucinate downstream arguments — a failure mode no amount of prompt optimization fixes.

The resilience-vs-compliance paradox. CoT is highly compliant but fragile: Traj. Order Match % of 70% but collapses on data variance. ReAct is messy but resilient: Traj. Order Match % of 22.7% yet self-corrects to the right answer. Optimization is a *multiplier* on resilience (Accuracy $35\% \rightarrow 65\%$) by adding guardrails against infinite tool loops, reflected in Avg Failures/Traj declining post-optimization while remaining non-zero. The paradox is only legible under the joint reading of the eight metrics; collapsing them to a scalar would have led the team to ship the structurally inferior CoT agent on the basis of “neater” trajectories.

6.3 Domain-specific evaluation

The proposed Optimized ReAct configuration (Config D) attained **86% Accuracy** on the domain-expert-curated set. Following an initial 50-user power-user feedback phase that produced both the curated evaluation corpus and the production beta signal, the agent was generally deployed to the full operational organization at the deployment site — an organization responsible for a global multi-site fulfillment operation — via an internal-facing web application with identity-based access management. The agent was made available as a discretionary analytics tool.

The eight-metric breakdown reveals where remaining failures concentrate:

Answer layer. Accuracy = 86% . Semantic F1 runs substantially higher: a non-trivial slice of the 14% accuracy gap is closed under decompositional claim-overlap scoring, indicating some “wrong” answers are correct in substance but lose strict-match credit to formatting, unit, or ordering differences. Cosine Sim is high across the board — even failed cases remain topically aligned. The progression localizes a meaningful portion of remaining failures to

the *Topically right, formally wrong* pattern — fixable by output-formatting guards rather than reasoning improvements.

Trajectory layer. *Traj. Set Match %* runs substantially higher than Accuracy: the agent invoked the canonical tool set on more queries than it answered correctly, indicating *tool selection is largely solved* and remaining failures concentrate at the argument and aggregation layers — not at semantic routing. *Traj. Order Match %* is low, consistent with ToolHop: the agent reaches correct answers through opportunistic self-correction rather than canonical traversal — on the production set this is a *feature*, not a bug. *Traj. Args Match %* is markedly lower than *Set Match*; the drop is the dominant remaining quality gap, directly nominating argument-binding as the next-best optimization investment. *Tool Coverage %* remains high.

Execution layer. *Avg Failures/Traj* is non-zero but declining post-optimization: the Simba-optimized agent retries productively on transient tool failures rather than terminating; the residual failures are largely DSL-ambiguity errors against the document-index tool.

This pattern — high Accuracy, *higher Set Match* and Coverage, *low Order Match*, *non-zero Avg Failures/Traj* — is the empirical *Resilient* signature defined in Table 1. The production set confirmed it transfers from the synthetic benchmark to domain-specific queries, and the *gap* between *Set Match* and *Args Match* directly nominates argument-binding as the highest-value next investment — a roadmap signal a single accuracy number cannot produce.

Beyond the curated 50-question set, the deployed agent is the subject of an ongoing controlled study to characterize how the agent affects investigation workflows when used; access is gated to a subset of investigators for study purposes, with usage discretionary in any given incident. Subjective evaluation in parallel indicates that **92% of agent responses receive a thumbs-up rating from a randomly selected user sub-sample**, providing a complementary signal to the curated correctness number.

7 Discussion

7.1 Curation beats scaling

The single most consequential decision for production utility was not the choice of reasoning engine, optimizer, or base model — it was the investment in domain-expert-validated question curation. The 50 questions, paired with expert-authored reference trajectories, serve three functions simultaneously: ground truth for evaluation, few-shot exemplars for DSPy compilation, and the canonical specification of “what the agent must do” against which roadmap priorities are weighed. The binding constraint on agent quality in domain-specific settings is the labeled, expert-validated evaluation substrate, not the model (Pan et al., 2025).

7.2 Stochasticity, DSL non-determinism, and schema evolution

At temperature 0, parallel GPU floating-point operations still introduce trajectory divergence. Against relational SQL, the ReAct agent achieved 94% self-consistency (Wang et al., 2023) (Pass@1 across 5 runs); against an equivalent document-index tool exposing a verbose JSON DSL, consistency dropped to 76% — the agent oscillated between syntactically different but semantically identical query expressions (Xue et al., 2025). Such failures score identically on Accuracy but diverge sharply on *Traj. Args Match %* and elevate *Avg Failures/Traj*.

The trace-extraction harness is designed to be robust to schema evolution in upstream data sources: tagging an MLflow run with `benchmark='True'` produces a trace record whose schema is independent of the underlying data layouts, so the eight-metric report remains comparable across release windows even when source-table definitions evolve. This decoupling is essential for per-release regression detection — the protocol can re-score logged traffic against a frozen gold-trajectory set without re-running the agent and without requiring source-side schemas to remain static.

7.3 Connection to behavioral evaluation more broadly

Our protocol fits the behavioral-evaluation theme by construction: it explicitly trades off final-task-success monolithism for behaviorally-grounded sub-metrics that decompose *how* the agent reached its answer. The five-pattern library of Table 1 is a working hypothesis about how multi-metric gap structures map to agent behaviors — testable on any tool-using agent for which a reference trajectory can be authored. We see three near-term extensions worth flagging for the workshop: (i) extending the trajectory layer to capture intra-step reasoning quality (e.g. via embedding-based scoring of thought traces), (ii) augmenting the diagnostic table with patterns specific to reflection / self-criticism agent variants, and (iii) tying each metric to a separate reward signal in end-to-end optimization (e.g. via DSPy GEPA), avoiding the scalar collapse that motivates the framework in the first place.

8 Limitations

Reference-trajectory authorship cost. The protocol’s trajectory-layer metrics require an expert-authored gold trajectory per question. In domains where multiple equally-valid solution paths exist, a single gold trajectory under-credits valid alternatives; Tool Coverage and Set Match partially mitigate this but do not eliminate it. Future work could allow gold trajectories to be specified as a *set* of acceptable paths.

Single-deployment empirical scope. The production validation comes from one deployment — an operational organization responsible for a global multi-site fulfillment operation — initially seeded by a 50-user power-user feedback cohort and then generally deployed to the broader organization. While the synthetic ToolHop benchmark complements this, additional deployments across varied operational domains would strengthen claims about the universality of the *Resilient* signature.

Non-release of the curated evaluation set and metric tracker implementation. The 50-question domain-expert-curated production set and the metric-tracker implementation are not publicly released due to the proprietary nature of the deployment data. The protocol (Section 4) is described in sufficient detail for independent reimplementations against the public ToolHop split or any analogous deployment-internal corpus.

Anglophone, English-only evaluation. All test questions and answers are in English. Behavioral patterns may differ for multilingual agents.

No mechanistic-explanation layer. The protocol localizes *which* layer fails but not *why* at the model-internals level (attention patterns, activation analysis). Mechanistic interpretability tooling (Olah et al., 2020) could be layered on top to explain the patterns of Table 1 causally.

9 Conclusion

Single-metric task-success scoring systematically misranks domain-specific LLM agents. We introduced an eight-metric tiered protocol — three answer-layer, four trajectory-layer, one execution-layer — whose joint reading surfaces five canonical behavioral patterns that single-metric protocols obscure. We validated the protocol on both a synthetic multi-hop benchmark and a domain-expert-curated production set from a real enterprise deployment, finding that DSPy-optimized ReAct attains 65% / 86% accuracy on the two settings, that DSPy optimization patches CoT’s syntactic compliance without unlocking adaptive reasoning, and that the *Resilient* behavioral signature transfers between synthetic and production settings.

References

- Carlos E. Jimenez et al. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.
- Omar Khattab et al. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024.

- arXiv:2310.03714. The `dspy.evaluate.SemanticF1` metric (with `decompositional=True`) is implemented in `dspy/evaluate/auto_evaluation.py` of the DSPy reference implementation (stanfordnlp/dspy).
- Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- Hunter Lightman et al. Let’s verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.
- Xiao Liu et al. AgentBench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*, 2023.
- Chang Ma et al. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- Chris Olah et al. Zoom in: An introduction to circuits. *Distill*, 2020. <https://distill.pub/2020/circuits/zoom-in/>.
- Melissa Z. Pan et al. Measuring agents in production. *arXiv preprint arXiv:2512.04123*, 2025.
- Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.
- Timo Schick et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- Noah Shinn et al. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- Kaitao Song et al. MPNet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- Xuezhi Wang et al. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- Xingyao Wang et al. Executable code actions elicit better LLM agents. *arXiv preprint arXiv:2402.01030*, 2024.
- Jason Wei et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- Dongge Xue et al. Text-to-ES Bench: A comprehensive benchmark for converting natural language to Elasticsearch query. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- Shunyu Yao et al. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- Shunyu Yao et al. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.
- Junjie Ye et al. ToolHop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- Shuyan Zhou et al. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.