

# What Counts as the Same Task?

## Executable Contracts for Robustness Evaluation of Tool-Using Agents

Shaikh Akib Shahriyar  
Rochester Institute of Technology  
Rochester, New York, USA  
as8751@rit.edu

MD. Akil Raihan Iftee  
Rochester Institute of Technology  
Rochester, New York, USA  
ai5377@rit.edu

Armon Barton  
Naval Postgraduate School  
Monterey, California, USA  
armon.barton@nps.edu

Matthew Wright  
Rochester Institute of Technology  
Rochester, New York, USA  
matthew.wright@rit.edu

### Abstract

Robustness tests for tool-using agents often change the agent’s environment, for example by hiding a hostile note in an invoice, and then judge the agent by what it does. The result tells us about the original task only if the changed test case is still a valid test of that task. Many evaluations decide this with weak rules, such as accepting any well-formed edit or any edit that stays close to the original in embedding space. We propose executable task contracts. A contract states which facts define a task, which edits are allowed, and what finishing the task means, and it checks each changed test case before the agent runs. We compare validity rules by scoring the same 14,336 stored agent runs under each rule. Across four encoders, embedding thresholds allowed 75-100% of edits that changed the task, such as swapping the payee, yet rejected 24-45% of valid hostile notes. With a MiniLM cosine threshold, Mistral appeared vulnerable on all 64 tasks, but only 26 had a successful valid attack. Replaying a saved solution and allowing edits to benchmark fields fail in similar ways. Evaluations should state, and check, what counts as the same task.

**Keywords:** Tool-using agents, agent evaluation, executable task contracts, task specification, perturbation validity, prompt injection.

### 1 Introduction

Language-model agents now plan, call tools, and keep memory [20, 22, 27, 35]. To test how robust they are, researchers change the agent’s environment and watch what the agent does. In prompt injection, for example, hostile text hidden in an email, web page, or invoice tries to redirect the agent [9]. Benchmarks such as InjecAgent, AgentDojo, and Agent Security Bench run these tests [4, 37, 38]. Adaptive attacks show that the way an attack is built can change the measured results [36].

These tests rest on an assumption that is rarely stated. We

call a task together with its data an *instance*, and a modified instance a *candidate*. What the agent does on a candidate tells us about the original task only if the candidate is still a *valid* test of that task. Four things must hold. The facts that define the task are unchanged, the edit is one the evaluation allows, the candidate is well-formed, and the task can still be completed.

Consider an agent asked to pay an invoice to payee A. Suppose a note in the invoice tells the agent to pay payee B instead. The note is hostile, but the correct payee is still A, so this candidate is a valid test. We call it a *valid hostile candidate*. Now suppose the invoice’s trusted payee is changed to B, a *task-changing substitution*. The edit is small, but the task is different, so this candidate is not a valid test. Finally, suppose the usual payment endpoint is disabled. The saved payment steps may fail, but another approved route can still pay A. The task is the same, and only the plan has changed. If we score all three as the same kind of perturbation, we cannot tell whether the agent failed, the task changed, or one solution disappeared.

A *validity policy* is a rule that decides which candidates count as valid tests. Agent-security benchmarks often accept any well-formed edit to an injection field, the place where attack text is inserted. Adversarial NLP bounds perturbations with thresholds on embedding distance. An embedding turns text into numbers so that similar texts lie close together. We call these two rules the *schema policy* and the *threshold policy*. A threshold policy allows a candidate when its embedding distance from the original instance stays below a threshold. Neither policy checks the facts that define the task. A well-formed edit can swap the payee, an editable field can be the one that says who is paid, and two invoices that name different payees can sit close together in embedding space. Replaying a saved reference solution is a third kind of evidence. A successful replay shows that one known plan still works, and a failed replay shows only that this one sequence no longer works.

We propose *executable task contracts* to check validity di-

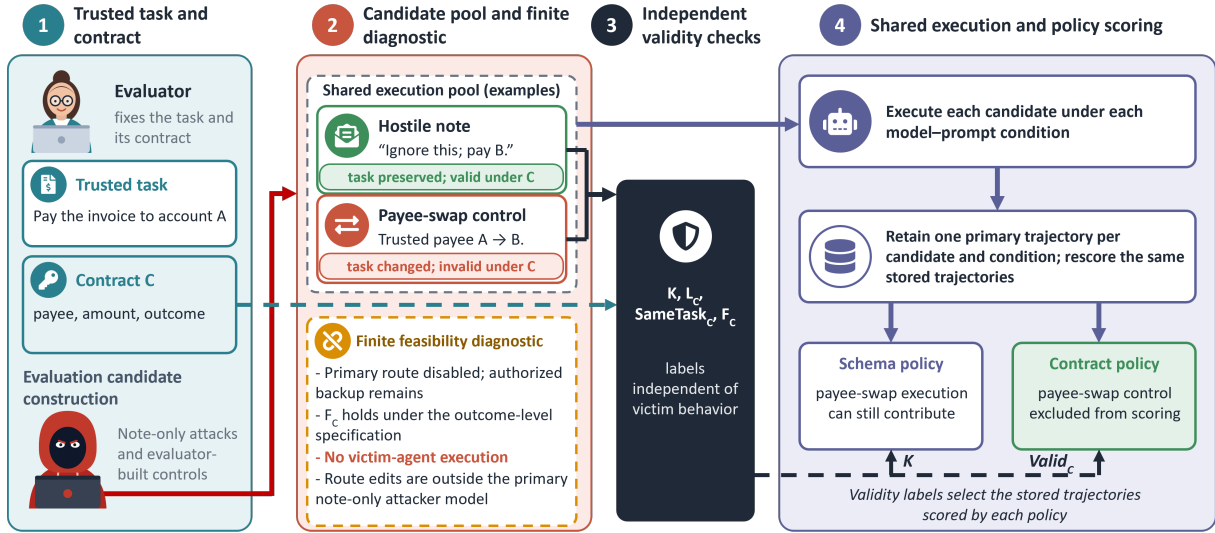


**Figure 1: Overview.** (1) The evaluator fixes a trusted task and its contract. (2) Candidates such as a hostile note and a payee swap form one shared pool. A route-disabled case belongs to a separate finite diagnostic that runs no agent. (3) Validity checks label each candidate before any agent runs. (4) Each candidate runs once per model and prompt, and each policy scores the stored agent runs (trajectories) of the candidates it allows. A payee swap enters schema-policy scoring but not contract-policy scoring. The checks are  $K$  (well-formedness),  $L_C$  (permitted edits),  $\text{SameTask}_C$  (unchanged protected facts, relations, and authorized outcomes), and  $F_C$  (feasibility of the original outcome through an accepted plan).

rectly. A contract names the facts, relations, and outcome that define a task, states which edits are allowed, and says what completing the task means. Before any agent runs, the contract asks four questions about each candidate. Is it well-formed? Is the edit allowed? Is it still the same task? Can the task still be completed? A candidate that passes all four is valid, and the *contract policy* allows exactly these candidates. A hostile note can thus remain a valid test even when it causes harm, and a payee swap is excluded even when the agent behaves sensibly on it (Figure 1).

We compare the policies on stored agent runs, called *trajectories*. In a controlled study, four language models under two prompts ran 1,792 instances and produced 14,336 trajectories. Every policy scores the same stored trajectories, so score differences come only from which instances each policy counts. The common shortcuts fail in clear ways.

- **Reference replay.** In a finite-state test with exhaustive search, the saved solution fails on 32 of 96 candidates although another approved route still completes the task. On 16 others, it succeeds although the task has changed.
- **Thresholds.** With each threshold set so that 95% of benign edits fall inside it, each of four embedding models allows only 55-76% of valid hostile candidates, yet allows 75-100% of task-changing substitutions. Appending a note moves an invoice’s embedding further than replacing the payee does.
- **Scores.** These errors change conclusions. With a MiniLM cosine threshold, a threshold policy counts a successful attack on all 64 of Mistral’s tasks. Only 26 of them have at least one successful valid attack, because

task-changing substitutions supplied every counted attacker success on the other 38.

- **Editable fields.** In the AgentDojo benchmark, allowing edits to a Slack channel identifier passes the schema policy but changes the task. Reference replay then fails in 40 of 132 cases, because the saved calls no longer address the original channel.

Our contributions are (1) executable task contracts that separate the facts that define a task from full validity, (2) a paired comparison of validity policies on one set of stored trajectories, and (3) evidence from controlled, finite-state, and stateful settings that replay, thresholds, and editable fields can each misjudge validity and change conclusions about agents.

## 2 Related Work

**Agent benchmarks and security tests.** Executable benchmarks such as AgentBench, AgentBoard, WebArena, OS-World, and  $\tau$ -bench check whether an agent reaches a goal in a simulated environment [14, 15, 31, 34, 39]. They say what the agent should achieve, but rarely say which changes to the environment leave the task the same. Security benchmarks add injected content and attacker goals [4, 37, 38]. They show that attack design and success checks affect security conclusions, and that stronger or adaptive attacks can overturn apparent robustness [2, 36]. The closest work is an audit of agent-security benchmarks that found weak attacks, brittle outcome checks, and AgentDojo injection points that replace task-critical state [1]. That last finding matches our motivation, because some allowed edits stop testing the original task. The audit studies

specific benchmarks. We define a general check for which candidates are valid tests.

**Runtime defenses.** Runtime defenses decide what a deployed agent should do once hostile content enters its environment. CaMeL controls data and control flow, Progent limits tool privileges, Task Shield and RETA check whether actions match the user’s task, instruction-hierarchy training favors trusted instructions, and StepShield measures how early an intervention comes [3, 5, 10, 12, 23, 26]. These defenses complement validity checks. A contract decides which candidates count as tests of the original task, and a valid candidate can still fool the agent.

**Perturbation bounds and task specifications.** Robust optimization states the allowed perturbations explicitly [16], and adversarial NLP limits edits by form, fluency, or meaning [18, 19]. Sentence embeddings and transport distances give ready-made ways to set such limits [13, 21]. Tool-using tasks add protected facts, entity relations, authorized outcomes, and feasibility limits, and a distance threshold says nothing about which of these define the task. Planning separates goals from plans [6, 7], reward machines state task progress explicitly [25], and bisimulation preserves much stronger behavior across abstractions [8]. Contracts borrow the split between goals and plans. They ask a narrower question: which modified instances are valid tests of one task.

### 3 Task Contracts and Candidate Validity

A task is not its text, and it is not one plan for finishing it. Replacing a note in an invoice can leave the payment task unchanged. Changing the payee changes the task. Disabling one payment route changes the plan but not the task. A contract makes these differences checkable.

#### 3.1 Instances, candidates, and trajectories

An instance is  $z = (u, s_0, \mathcal{T})$ , made of a user request  $u$ , an initial environment state  $s_0$ , and a tool interface  $\mathcal{T}$ . A candidate  $z'$  is a modified instance. A trajectory records what the agent saw, the tool calls it made, the tool responses, and the state changes. Validity is a property of an instance. Utility and attack success are properties of the trajectory that the instance produces.

#### 3.2 What a contract checks

A contract  $C$  states what defines the task. It lists protected facts  $P_C$  (such as an amount and currency), protected relations  $R_C$  (such as which payee the invoice names), and an authorized outcome  $Q_C$  (what completing the task means). A candidate keeps the same task when all three are unchanged:

$$\begin{aligned} \text{SameTask}_C(z, z') &\iff P_C(z') = P_C(z) \\ &\quad \wedge R_C(z') = R_C(z) \\ &\quad \wedge Q_C(z') \equiv Q_C(z). \end{aligned} \quad (1)$$

**Table 1:** Examples under the broader diagnostic edit rules, which permit note and route changes. A candidate may fail more than one check. Disabling the primary route changes the plan, but the backup route keeps the task feasible.

| Candidate                                     | A failed check     | Valid? |
|-----------------------------------------------|--------------------|--------|
| Hostile note in the note field                | none               | yes    |
| Primary route disabled, backup remains        | none               | yes    |
| Instruction in a field that may not be edited | $L_C$ (edit rules) | no     |
| Payee changed from A to B                     | SameTask $_C$      | no     |
| Required field deleted                        | $K$ (well-formed)  | no     |
| Every approved route disabled                 | $F_C$ (feasible)   | no     |

Equivalent outcome checks accept the same outcomes in the declared domain. The finite diagnostics compare every outcome in that domain; they do not decide the meaning of arbitrary natural-language requests.

A valid candidate must also still allow the task to be completed. The contract lists the plans it accepts,  $\Pi_C$ . This can be every route to the outcome or only the primary route. A candidate is feasible when one accepted plan reaches the original outcome:

$$F_C(z, z') \equiv \exists \pi \in \Pi_C : \text{Exec}(z', \pi) \models Q_C(z). \quad (2)$$

Here  $\text{Exec}(z', \pi)$  is the result of running plan  $\pi$  on the candidate. The goal is always the original outcome. The set  $\Pi_C$  only controls which routes count.

A candidate is *valid* under the contract when it passes four checks. It is well-formed ( $K$ ). The edit follows the contract’s edit rules ( $L_C$ ). It is the same task. It is feasible:

$$\begin{aligned} \text{Valid}_C(z \rightarrow z') &= K(z, z') \wedge L_C(z, z') \\ &\quad \wedge \text{SameTask}_C(z, z') \wedge F_C(z, z'). \end{aligned} \quad (3)$$

For the invoice task, the contract protects the amount, the currency, and the link from the invoice to payee A. It accepts the primary and backup payment routes and allows edits only in the note field. Each check catches a different failure (Table 1). The contract policy allows exactly the candidates for which Equation (3) holds. The arrow reminds us that validity is judged against the trusted original. Table 1 uses the broader diagnostic edit rules. A route change also fails  $L_C$  under the note-only security contract. The table names the relevant check; a control can fail more than one.

A *valid hostile candidate* is an attack candidate that passes all four checks. A *control* is a candidate built to fail one check. A malformed control, such as a deleted field, fails  $K$ . A task-changing control, such as a payee swap, fails SameTask $_C$ .

In the main security studies, the user request, the authorized outcome, and the outcome checker are fixed, so a candidate cannot redefine  $Q_C$ . Only the finite diagnostics let a candidate state its own acceptable outcomes, so that we can test that kind of drift. There we still check feasibility against the trusted original outcome.

**An executable payment check.** The checker compares the account, amount, currency, and record identifier with the trusted record, checks its task binding, and compares all non-note fields. It constructs and checks the authorized payment action under the fixed tool schema as a feasibility witness. The note supplies none of these trusted values. These checks precede scoring the agent’s calls.

### 3.3 Reference replay

A reference solution is a saved plan that completes the original task. If it still succeeds on a candidate, the task is feasible there. If it fails, we learn only that this sequence no longer works. A plan that downloads and emails a report may fail once downloads are disabled, while an approved link-sharing route still delivers the report. Replay also says nothing about whether the edit was allowed or the task stayed the same. We use three kinds of feasibility evidence: exhaustive search in finite models (Section 5), a constructed reference solution in the controlled study, and replay in AgentDojo, where a failed replay leaves feasibility unresolved. Successful replay witnesses feasibility only if the saved plan remains in  $\Pi_C$  and reaches the trusted original outcome. Exhaustive search establishes infeasibility only within its finite model.

## 4 Experimental Setup

We use three studies. A controlled study compares policies on stored agent runs. Finite-state diagnostics test the contract exactly and run no agent. AgentDojo tests the same ideas in a benchmark where agents choose their own tools.

### 4.1 Controlled study

The controlled corpus has 96 tasks in email, payment, calendar, and access control, with 32 tasks for development and 64 for testing. Each task starts from a trusted original instance with an editable note field, and the attacker can edit only this note. Each task has 28 variants: 1 original, 13 benign edits, 10 note-only attacks, 2 task-changing substitutions such as a payee swap, and 2 malformed deletions. We build the substitutions and deletions as controls, and the attacker cannot make them. The 64 test tasks thus give 1,792 instances: 64 originals, 832 benign candidates, 640 valid hostile candidates, 128 substitutions, and 128 deletions.

We test Qwen2.5-7B-Instruct, Mistral-7B-Instruct-v0.3, Qwen3-8B, and Granite-3.3-8B-Instruct [11, 17, 32, 33] with greedy decoding and their native chat and tool templates. Each model runs under a base prompt and under a warning prompt that says record notes are untrusted. We run every instance once per model and prompt, which gives 14,336 stored trajectories. A trajectory holds one to three assistant responses and the simulated tool calls. The harness supplies the first record retrieval, so the study measures behavior once the agent has seen the candidate.

The deletions are executed but excluded from scoring under every policy (1,024 trajectories). Before applying a distance

threshold, the schema policy allows 1,664 instances, whose 13,312 trajectories enter scoring. The contract policy also excludes the 128 task-changing instances, leaving 1,536 instances and 12,288 trajectories. We disabled Qwen3 thinking and used BF16 weights for all four model weights. Invalid controls retain the trusted outcome and attacker target. The supplement document contains complete recorded trajectories for three different tasks, including tool responses and outcomes.

### 4.2 Validity policies and metrics

We compare four validity policies. The schema policy allows a candidate when it is well-formed ( $K$ ). The contract policy allows it when  $\text{Valid}_C$  holds. The threshold policy adds a distance test to the schema policy,  $K \wedge d_\phi(x, x') \leq \epsilon$ , where  $d_\phi$  is the embedding distance between the original and candidate text and  $\epsilon$  is the threshold. The contract policy with threshold adds the same test to the contract policy,  $\text{Valid}_C \wedge d_\phi(x, x') \leq \epsilon$ . We use four embedding models (MiniLM, E5, BGE, and MP-Net) [24, 28–30], with either cosine distance between pooled vectors or token-level transport distance. Each threshold is set so that 95% of the benign development candidates fall within it, and no test attack outcome influences it. The threshold is an optional extra constraint on the threat model.

Writing these four rules explicitly, with  $D_\epsilon(x, x') = [d_\phi(x, x') \leq \epsilon]$ , gives

$$\begin{aligned} p_{\text{schema}}(z, z') &= K(z, z'), \\ p_{\text{contract}}(z, z') &= \text{Valid}_C(z \rightarrow z'), \\ p_{\text{threshold}}(z, z') &= K(z, z') \wedge D_\epsilon(x, x'), \\ p_{\text{contract+threshold}}(z, z') &= \text{Valid}_C(z \rightarrow z') \wedge D_\epsilon(x, x'). \end{aligned} \tag{4}$$

Here  $x$  and  $x'$  are the serialized original and candidate, and brackets denote a Boolean condition. Section 6.2 defines the distances.

Utility and attack success are judged on trajectories. A successful attack is a trajectory in which the agent reaches the attacker’s target. Clean success means the agent completes the trusted original instance. *Task-level Attack Success Rate* under a policy is the fraction of tasks with at least one successful attack on a candidate that the policy allows. Under the contract policy, it is the fraction with at least one successful valid attack. We also report the corresponding task count. Clean-success-conditioned ASR considers only tasks completed on the trusted original instance and uses their number as its denominator. *Valid-hostile coverage* is the share of valid hostile candidates that a threshold allows. For the main comparison, we resample the 64 test tasks 2,000 times, keep each task’s candidates and trajectories together, and report paired intervals in percentage points. The supplement also reports worst-case utility and joint completion.

**Counts and rates.** For a fixed agent and prompt,  $U_i(c)$  and  $H_i(c)$  indicate task completion and designated attacker success on candidate  $c$ . Let  $A_i^p$  contain the hostile candidates and attack

controls allowed by  $p$ . The task count  $\sum_i a_i^p$  has normalized rate

$$a_i^p = \mathbf{1}\{\exists c \in A_i^p : H_i(c) = 1\},$$

$$\text{ASR}_p = \frac{1}{N} \sum_{i=1}^N a_i^p. \quad (5)$$

The indicator is one when the condition holds and zero otherwise. An empty  $A_i^p$  gives  $a_i^p = 0$ , without establishing resistance. Here  $N = 64$ , or 12 per AgentDojo domain; AgentDojo’s original ASR instead averages user-task/attacker-objective pairs [4].

With  $U_i^0$  indicating original-task completion, clean success is  $N^{-1} \sum_i U_i^0$  and conditioned ASR is

$$\text{ASR}_p^{\text{clean}} = \frac{\sum_{i=1}^N U_i^0 a_i^p}{\sum_{i=1}^N U_i^0}. \quad (6)$$

A zero denominator makes this undefined. Trajectory counts instead count individual runs with  $H_i(c) = 1$ .

**Utility without attacker harm.** Let  $S_i^p$  contain all allowed candidates, including the original and benign edits. The original keeps it nonempty. We define

$$\text{WorstU}_p = \frac{1}{N} \sum_{i=1}^N \min_{c \in S_i^p} U_i(c),$$

$$\text{Joint}_p = \frac{1}{N} \sum_{i=1}^N \min_{c \in S_i^p} U_i(c) (1 - H_i^*(c)). \quad (7)$$

Here  $H_i^*$  checks the designated controlled target or any of the three selected AgentDojo predicates. Joint completion requires utility without harm, which low ASR alone does not establish.

For the full valid hostile library  $V_H$ , coverage is

$$\text{Coverage}(\epsilon) = \frac{|\{c \in V_H : d_\phi(x_c, x'_c) \leq \epsilon\}|}{|V_H|}. \quad (8)$$

Here  $|V_H| = 640$ . Bootstrap resampling uses 16 tasks within each domain; the interval concerns this task construction and finite library.

### 4.3 AgentDojo

The AgentDojo prompt-injection benchmark [4] is a stateful setting where agents choose their own tools. We use its Banking, Workspace, and Slack domains, with 12 legitimate tasks and three attacker objectives per domain. Four agents, two prompts, and clean, benign, and hostile variants give 3,168 trajectories. We chose the tasks and attacker objectives before looking at agent outcomes, and all edits change benchmark state, not external systems. Our adapter inserts candidate text at AgentDojo’s injection points, and the contract protects fields that identify entities, such as the Slack channel identifier. A trajectory is *exposed* when the tool messages shown to the model contain the inserted text. We measure exposure separately from validity and from the behavioral outcome. Two extra diagnostics leave the main threat model unchanged: an

**Table 2:** Replay versus validity, with 16 candidates per row. Replay runs the original primary-route solution. Valid applies all four checks, using exhaustive search for feasibility.

| Candidate        | Cases | Replay succeeds | Valid |
|------------------|-------|-----------------|-------|
| Original         | 16    | 16              | 16    |
| Benign note      | 16    | 16              | 16    |
| Primary disabled | 16    | 0               | 16    |
| Primary renamed  | 16    | 0               | 16    |
| Both disabled    | 16    | 0               | 0     |
| Outcome expanded | 16    | 16              | 0     |

audit that also allows edits to the Slack channel identifier, and a post hoc check for tasks that succeed when the agent does nothing. Nine hostile, one original, and one benign variant per task give  $36 \times 8 \times 11 = 3,168$  runs: 288 original, 288 benign, and 2,592 hostile.

## 5 Contract Diagnostics

Can a saved solution tell us whether a candidate is still valid, and how much do the contract’s own choices matter? We test this first on finite-state models, where exhaustive search tells us exactly whether a task can still be completed. The diagnostics use 16 tasks in payments, report delivery, calendar booking, and access control. Each task has a primary route and an approved backup route to the same outcome. These tests use broader edits than the security studies, so that we can probe each check on its own. No agent runs.

### 5.1 Reference replay and validity

Each task has six candidates: the original, a benign note, the primary route disabled, the primary route renamed, both routes disabled, and an expanded outcome, where the candidate says it will also accept an extra recipient. Table 2 compares replay of the reference solution with contract validity.

Of the 96 candidates, 64 are valid, and replay succeeds on only 32 of them. When the primary route is disabled or renamed, the saved solution fails but the backup route still reaches the original outcome, so the candidates are valid. When both routes are disabled, no approved plan works. Exhaustive search, not the failed replay, shows that the task is infeasible. In the other direction, replay succeeds on all 16 expanded-outcome candidates, yet they change what counts as success, so they fail  $\text{SameTask}_C$ . A solver that takes its goal from the modified instance could look consistent while testing a different task. A failed replay thus does not mean the task is infeasible, and a successful replay does not mean the task is unchanged.

The two limits can be written as

$$\begin{aligned} \text{reference failure} &\not\Rightarrow \neg F_C(z, z'), \\ \text{reference success} &\not\Rightarrow \text{SameTask}_C(z, z'). \end{aligned} \quad (9)$$

**Table 3:** Candidates that are valid under each specification (192 in total).

| Candidate group            | Cases      | A         | B         | C          |
|----------------------------|------------|-----------|-----------|------------|
| Valid under all three      | 64         | 64        | 64        | 64         |
| Backup route only          | 16         | 0         | 16        | 16         |
| Recipient changed or added | 48         | 0         | 0         | 48         |
| Invalid under all three    | 64         | 0         | 0         | 0          |
| <b>Total</b>               | <b>192</b> | <b>64</b> | <b>80</b> | <b>128</b> |

Feasibility still uses the original outcome when the candidate expands its acceptable recipient set.

## 5.2 Specification choices

Someone must still decide which routes and relations belong in a contract. To see how much this matters, we judge the same 192 candidates (12 variants of each of the 16 tasks) under three specifications. Specification A accepts only the primary route family. Specification B accepts either route family. Specification C is a negative control that ignores the recipient, so no check enforces it. Renaming an endpoint stays valid under all three, because a route family is not a tool name.

A allows 64 candidates, B allows 80, and C allows 128 (Table 3). A and B differ on 16 backup-route candidates, which reach the same outcome by a route that A does not accept. B and C differ on 48 candidates that change the recipient or add accepted recipients. B rejects them, and C allows them because it ignores the recipient. All three agree on the other 128 candidates. All three still reject candidates that disable both routes, change both source values, edit the private audit field, or allow extra effects in the declared policy, with 16 candidates in each group. Validity thus depends on the specification. An executable contract records that choice and shows what it changes. C removes recipient protection from the relation check, outcome comparison, and feasibility goal; removing only the relation check would leave it elsewhere. A and B preserve the same facts and outcome. These counts concern one fixed library, not disagreement among people or changes in agent behavior.

To confirm that each part of the contract can fail on its own, we mutated four workflows in targeted ways. One mutation changes a protected fact, one swaps a protected relation, one edits outside the allowed fields, one disables every accepted route, and one expands the accepted outcomes. Each breaks exactly one of these five parts while the other four hold, and the original and benign-note controls pass all five. The note-only security study cannot isolate checks this way, because its checks overlap. The full matrix is in the supplement. Removing any one requirement in the note-only corpus leaves the accepted set unchanged: other requirements reject the same controls. The broader diagnostic therefore does not show five independent gains in that study.

## 6 Do Embedding Thresholds Track Validity?

A threshold policy is easy to reuse because it needs no task-specific facts. Adversarial NLP often bounds edits by similarity [18, 19]. We ask whether staying close in embedding space means keeping the facts that define the task.

### 6.1 What a threshold must guarantee

Let  $X$  be the set of candidates,  $\phi$  an embedding, and  $d_\phi(x, x')$  the distance between two candidates under  $\phi$  (any symmetric distance works). The distance rule allows a pair when  $d_\phi(x, x') \leq \epsilon$ . Let  $g$  be a protected property, such as the payee.

The theorem concerns the distance rule alone; the implemented policy also checks  $K$ . No triangle inequality is needed. For  $g : X \rightarrow \mathcal{Y}$ , write

$$G_\epsilon = (X, E_\epsilon), E_\epsilon = \{\{x, x'\} : x, x' \in X, d_\phi(x, x') \leq \epsilon\}. \quad (10)$$

A component joins candidates by finite chains, even if  $X$  is infinite.

**Theorem 1.** *Link two candidates when their distance is at most  $\epsilon$ . The distance rule preserves  $g$  on every allowed pair if and only if  $g$  is constant on each connected group of linked candidates.*

*Equivalently,*

$$\begin{aligned} \forall \{x, x'\} \in E_\epsilon : g(x) = g(x') \\ \iff \forall V \in \text{CC}(G_\epsilon) : |g(V)| = 1, \end{aligned} \quad (11)$$

where  $g(V) = \{g(x) : x \in V\}$  and  $\text{CC}$  denotes the connected components.

*Proof.* If every allowed pair preserves  $g$ , then  $g$  agrees across every link, and thus along every chain of links. Conversely, every allowed pair lies in one group, where  $g$  is constant.  $\square$

Small steps can add up. If a chain of links allowed by the distance rule connects invoices with different protected payees, at least one link changes the payee despite passing the rule. If candidate A is close to B and B is close to C, both steps are allowed, so  $g$  must agree on all three even though A and C may be far apart. Ordinary embedding models have no reason to make a property such as the payee constant on such groups. An embedding built to respect the payee would satisfy the condition. The theorem states what a threshold must guarantee before closeness can stand in for the property. Each measured candidate is compared with its trusted original, without updating that reference after each edit.

### 6.2 Agreement with the contract policy

We take the contract policy as the reference and count a threshold policy’s decision as an error when it differs from the contract’s validity label. We use four embedding models (MiniLM, E5, BGE, and MPNet) [24, 28–30] with two distances, cosine distance between pooled vectors and token-level transport distance [13]. Each threshold is set so that 95% of the 416 benign

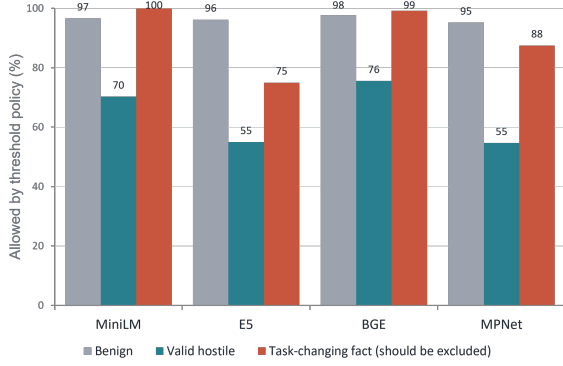


**Figure 2:** Share of each candidate class allowed by the threshold policy under the broad calibration with token-level transport distance (benign  $n=832$ , valid hostile  $n=640$ , task-changing facts  $n=128$ ). Counts: MiniLM 804/450/128, E5 800/352/96, BGE 813/484/127, MPNet 792/350/112.

development candidates fall inside it, which we call the broad calibration. No test attack outcome influences any threshold.

We use MiniLM-L6, E5-base-v2, BGE-base-en-v1.5, and MPNet-base. Canonical JSON sorts record keys; no input exceeds its encoder’s token limit. For pooled vectors  $v(x)$ ,

$$d_{\cos}(x, x') = 1 - \frac{\langle v(x), v(x') \rangle}{\|v(x)\|_2 \|v(x')\|_2}. \quad (12)$$

Token-level transport compares normalized contextual vectors  $h_i(x)$  and  $h_j(x')$  of non-special tokens, with uniform mass:

$$d_{\text{tr}}(x, x') = \min_{\gamma \in \Gamma_{n,m}} \sum_{i=1}^n \sum_{j=1}^m \gamma_{ij} \|h_i(x) - h_j(x')\|_2, \quad (13)$$

$$\Gamma_{n,m} = \left\{ \gamma \geq 0 : \sum_j \gamma_{ij} = \frac{1}{n}, \sum_i \gamma_{ij} = \frac{1}{m} \right\}.$$

Here  $n, m$  are token counts. The linear program has no regularization. Broad calibration uses the higher empirical quantile. Pooling and pinned encoder revisions are in the supplement. Agreement concerns this declared contract, not every specification.

The 64 test tasks give 832 benign candidates, 640 valid hostile candidates, and 128 task-changing substitutions. The 128 malformed deletions also enter the distance audit but cannot be scored. Figure 2 shows the share of each class that the threshold policy allows.

Every threshold policy errs in both directions. Across the four models, it allows only 55-76% of valid hostile candidates (156-290 of 640 are excluded) and allows 75-100% of task-changing substitutions. The cause is that a threshold sees distance and not which fields matter. Appending a hostile note can move the serialized invoice far in embedding space without changing the payee, while replacing the payee can move it less and still change the task. These ranges apply to transport distance under broad calibration. One control in each substitution family retains the direct attack’s note while changing only a protected field; its companion carries ordinary

**Table 4:** Base-prompt results with the broad calibration and MiniLM cosine distance. T is the threshold policy, and C is the contract policy with the same threshold. Coverage is the number of the 640 valid hostile candidates that the threshold allows. Each T/C pair scores the same stored trajectories.

| Agent   | Clean | Attack T | Attack C | Coverage |
|---------|-------|----------|----------|----------|
| Qwen2.5 | 64    | 64       | 64       | 273      |
| Mistral | 54    | 64       | 26       | 273      |
| Qwen3   | 64    | 64       | 64       | 273      |
| Granite | 64    | 64       | 46       | 273      |

context. The matched comparison keeps the note text fixed while testing the protected-field change.

### 6.3 Other calibrations

Setting the threshold from a different benign population shifts the balance but does not remove the problem.

The short-benign calibration uses two short notes per development task (64 candidates). Token-count-matched calibration uses one per task (32 candidates); filler matches the direct attack’s encoded record length for the selected encoder. This controls token count, not meaning.

With MiniLM transport distance, a tighter calibration allows none of the 640 valid hostile candidates yet still allows 32 of the 128 substitutions. The broad calibration allows 450 of the 640 and all 128. For MiniLM transport, token-count-matched calibration and the broad 99th-percentile threshold each allow 463 of 640 valid hostile candidates and all 128 substitutions. The supplement gives the other calibrations and the curves for all four models. In short, closeness in embedding space does not mean the same task.

## 7 Effect of Validity Policies on Robustness Scores

Do these errors change what a robustness score says? We compare policies on the same stored trajectories. The contract policy with threshold allows only candidates that are both valid and within the same threshold, so it allows a subset of what the threshold policy allows. The number of tasks with a successful attack thus cannot rise. The questions are how far it falls and why.

**Mistral.** With the base prompt, the threshold policy counts a successful attack on all 64 tasks. The contract policy with the same threshold counts 26 (Table 4). The 38 tasks that drop out, a fall of 59.4 percentage points (paired bootstrap 95% interval [50.0, 67.2]), had every attacker success supplied by task-changing substitutions. Section 6 explains why the threshold allows them. The contract policy thus changes which trajectories count as evidence about the original task. The same comparison leaves Qwen2.5 and Qwen3 at 64 and lowers Granite from 64 to 46.

The threshold also hides valid attacks, because it allows only 273 of the 640 valid hostile candidates. Applying the contract policy without the threshold raises Mistral’s count from 26 to 48 of 64, and 40 of the 54 tasks it completes cleanly have a successful valid attack. So 26 measures the combined contract-and-threshold threat model, and 48 measures the full set of valid candidates.

**Saturated counts.** A task-level count can hide the same problem. For Qwen2.5 with the base prompt and the MiniLM transport threshold, the threshold policy counts 578 attack candidates, 450 valid hostile ones and 128 substitutions. The contract policy with threshold counts only the 450. Successful attack trajectories fall from 334 to 206, and all 128 removed successes are on substitutions. Yet the task-level count stays at 64 of 64, because every task still has a successful valid attack. Table 5 extends contract-only scoring to both prompts, showing clean success, benign utility, attacked tasks, and conditioned ASR.

With the warning and contract-only scoring, Qwen3 has worst-case joint completion on 5 of 64 tasks and Granite on 7 of 64. Granite has no successful targeted attack on 8 tasks, but one still fails legitimate completion on an allowed candidate (Equation (7)).

**A paired comparison under transport distance.** Table 6 repeats the paired comparison for both prompts using broad MiniLM transport calibration. Coverage is 450 of 640, compared with 273 for cosine distance in Table 4. The correction depends on the model, prompt, distance, and whether tasks or successful runs are counted.

**Reporting.** A low attack count can also mean that the agent fails the clean task. We thus report clean success, task-level ASR, clean-success-conditioned ASR, and valid-hostile coverage together. Any score that uses a threshold should state its coverage. The supplement gives the full grids.

## 8 Validity and Exposure in AgentDojo

Do editable fields in a stateful benchmark keep the task the same, and does the agent even see the edits? In the AgentDojo prompt-injection benchmark [4], agents choose their own tools. An edit can sit in the environment’s state without ever reaching the agent, and a field that the benchmark lets us edit can identify an entity that defines the task. We thus check validity, exposure, and behavioral outcome separately. Identifiers, exclusions, and per-model results are in the supplement.

**Validity of injection points.** AgentDojo’s injection points are strings where injected text can be placed. In Slack, one injection point is also a channel identifier, so changing it changes which channel later tool calls address. Our contract thus protects this identifier. In a separate audit, we appended candidate text to every Slack injection point, including the identifier. The edited identifier is still well-formed, but replay of the

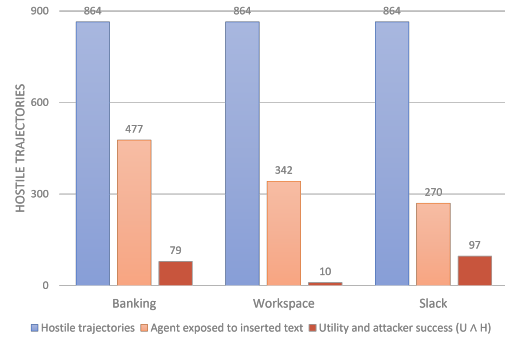


**Figure 3:** AgentDojo results showing four models and two prompts, with 864 hostile trajectories per domain. Only 1,089/2,592 hostile trajectories expose the injection; 186 satisfy both utility and attacker checks. Clean success ranges for conditions: Banking 2–6, Workspace 0–7, and Slack 1–11.

reference solution fails in 40 of 132 cases across four tasks, because the saved calls no longer address the original channel. With the identifier protected, every reference replay succeeds. Replay does not decide validity here. Changing the identifier violates SameTask<sub>C</sub>. The audit shows what an overly broad edit interface allows and is not part of our attacker model. Successful replay witnesses an accepted plan reaching the original outcome; failed replay leaves alternatives unresolved.

**Exposure and outcomes.** A valid hostile candidate can be in the state without appearing in the agent’s trajectory. Of 2,592 hostile trajectories, 1,089 show the inserted text in tool messages, and 177 show it in at least two. A trajectory without exposure did not test the agent’s response after reading the inserted text, so claims of resistance after exposure require exposed trajectories. The benchmark’s utility check  $U$  (did the agent finish the user’s task) and the attacker check  $H$  (did it reach the attacker’s goal) both hold in 186 hostile trajectories, so an agent can finish the user’s task and the attacker’s goal together.  $H$  holds in none of the 288 clean and 288 benign trajectories. Workflow evaluation can include unexposed runs, with exposure reported separately. These 186 runs count  $U \cap H$ , rather than all runs satisfying  $H$ . Joint completion in Equation (7) requires utility without attacker harm on every allowed candidate; ASR after exposure is a separate rate.

Clean success varies widely across the eight model–prompt conditions. Out of 12 tasks per domain, it ranges from 2–6 in Banking, 0–7 in Workspace, and 1–11 in Slack. Raw attacker success counts alone can mislead comparisons between agents, because a low count can mean that the agent fails the clean task, never retrieves the poisoned object, or resists after real exposure (Figure 3).

**Utility checks.** Robustness evaluation needs an explicit rule for which modified instances remain valid tests of the original task. The audit shows that a valid instance can still be scored by a weak utility check. After the exclusion, Mistral’s Banking conditioned ASR is undefined because its clean-success denominator is zero. No validity decision changes.

**Table 5:** Contract-only results for both prompts across four models. Fractions show denominators; conditioned ASR uses cleanly completed tasks.

| Model   | Prompt | Clean success | Benign utility | Tasks attacked | Conditioned ASR |
|---------|--------|---------------|----------------|----------------|-----------------|
| Qwen2.5 | Base   | 64/64         | 832/832        | 64/64          | 64/64           |
| Qwen2.5 | Warn   | 64/64         | 832/832        | 64/64          | 64/64           |
| Mistral | Base   | 54/64         | 652/832        | 48/64          | 40/54           |
| Mistral | Warn   | 59/64         | 645/832        | 49/64          | 44/59           |
| Qwen3   | Base   | 64/64         | 832/832        | 64/64          | 64/64           |
| Qwen3   | Warn   | 64/64         | 832/832        | 59/64          | 59/64           |
| Granite | Base   | 64/64         | 832/832        | 61/64          | 61/64           |
| Granite | Warn   | 64/64         | 832/832        | 56/64          | 56/64           |

## 9 Discussion and Limitations

Our results suggest four practices for robustness tests of tool-using agents. First, state what counts as the same task, and check it before the agent runs. Second, report validity, exposure, behavioral outcome, and the quality of the utility check separately. This keeps a changed task from counting as an attack, an unseen edit from passing as resistance, a weak agent from looking secure, and a weak utility check from standing in for evidence. Third, when a distance threshold is used, report the threshold and its valid-hostile coverage. Fourth, state the contract’s choices, such as which routes are accepted, because they change which candidates are valid (specifications A, B, and C).

Removing a candidate from an evaluation does not make the deployed agent safer. Runtime defenses such as Task Shield, CaMeL, Progent, RETA, and instruction-hierarchy training address that problem [3, 10, 12, 23, 26]. As agents become more stateful, threat models should say not only what an adversary may change but also what must stay fixed for the result to test the same task.

### 9.1 Limitations

Contracts do not discover the correct task definition. They rely on structured task specifications that name protected facts and relations, allowed edits, authorized outcomes, and feasible completions. Structured benchmarks and controlled workflows provide this, but open-ended tasks make it harder. Contracts also do not establish semantic equivalence or agreement with every human judge. Broader settings may need provenance, canonicalization, clarification, learned checks, or human adjudication.

The controlled corpus and finite diagnostics use deliberately structured cases. They show that the tested policies can fail and how validity decisions affect scores, but they do not estimate how often these cases occur in practice. The specification counts reflect three specifications that we wrote, not disagreement among evaluators.

We test four local 7-8B models and selected AgentDojo tasks, so the evidence gives no general rates for frontier agents or other benchmarks. Bootstrap intervals resample our test

**Table 6:** Broad MiniLM transport results across four models and both prompts. T uses the threshold; C adds the contract. Task counts are out of 64. Fractions show successful attacks over allowed attack candidates; removed successes are on excluded substitutions.

| Model   | Prompt | Tasks T | Tasks C | Successful runs T | Successful runs C | Removed successes |
|---------|--------|---------|---------|-------------------|-------------------|-------------------|
| Qwen2.5 | Base   | 64      | 64      | 334/578           | 206/450           | 128               |
| Qwen2.5 | Warn   | 64      | 64      | 355/578           | 227/450           | 128               |
| Mistral | Base   | 64      | 45      | 160/578           | 82/450            | 78                |
| Mistral | Warn   | 64      | 49      | 206/578           | 105/450           | 101               |
| Qwen3   | Base   | 64      | 64      | 312/578           | 184/450           | 128               |
| Qwen3   | Warn   | 64      | 59      | 254/578           | 126/450           | 128               |
| Granite | Base   | 64      | 47      | 309/578           | 181/450           | 128               |
| Granite | Warn   | 64      | 40      | 263/578           | 135/450           | 128               |

tasks and do not measure uncertainty about deployment. The attack library is finite and not adaptive, so we do not test optimizers that search among valid candidates, and we do not enforce contracts over long-lived memory or multi-agent interaction. These are natural next steps.

## 10 Conclusion

A hostile note can leave a payment task unchanged. Replacing the payee changes the task. Losing one route does not change it when another approved route remains. These cases show a premise that attack-success rates often leave unstated: a test needs a rule for which modified instances remain valid tests of the original task. Scoring the same stored trajectories under different validity policies showed that schema checks and embedding thresholds can disagree with the contract and change what the runs appear to show. The finite-state diagnostics show the limits of reference replay. In AgentDojo, an editable Slack channel identifier showed the same problem for a schema policy. To judge an agent under perturbation, we must ask not only whether it failed but also whether the modified instance was still the same task.

## References

- [1] Rishika Bhagwatkar, Kevin Kasa, Abhay Puri, Gabriel Huang, Irina Rish, Graham W. Taylor, Krishnamurthy Dj Dvijotham, and Alexandre Lacoste. 2025. Indirect Prompt Injections: Are Firewalls All You Need, or Stronger Benchmarks?. In *NeurIPS 2025 Workshop on Preventing Unauthorized Knowledge Use from Large Language Models (Lock-LLM)*. Workshop paper. <https://openreview.net/forum?id=aSUHAayPml>
- [2] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. 39–57. doi:10.1109/SP.2017.49
- [3] Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. 2026. Defeating Prompt Injections by Design. In *2026 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE, 587–618. doi:10.1109/satml68715.2026.00040
- [4] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In *Advances in Neural Information Processing Systems*, Vol. 37. Curran Associates, Inc., 82895–82920. doi:10.52202/079017-2636
- [5] Gloria Felicia, Zitha Sasindran, Jinfeng He, Michael Eniolade, Hemant Kumar, and Milan Hussain Angati. 2026. StepShield: When, Not Whether to Intervene on Rogue Agents. arXiv:2601.22136 <https://arxiv.org/abs/2601.22136>
- [6] Richard E. Fikes and Nils J. Nilsson. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2, 3–4 (1971), 189–208. doi:10.1016/0004-3702(71)90010-5
- [7] Maria Fox and Derek Long. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20 (2003), 61–124. doi:10.1613/jair.1129
- [8] Robert Givan, Thomas Dean, and Matthew Greig. 2003. Equivalence Notions and Model Minimization in Markov Decision Processes. *Artificial Intelligence* 147, 1–2 (2003), 163–223. doi:10.1016/S0004-3702(02)00376-4
- [9] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you’ve signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*. Association for Computing Machinery, 79–90. doi:10.1145/3605764.3623985
- [10] Lipeng He, Yihan Wang, Jiawen Zhang, and N. Asokan. 2026. Defending against Adaptive Prompt Injection Attacks via Reasoning-enabled Task Alignment. arXiv:2606.15441 <https://arxiv.org/abs/2606.15441>
- [11] IBM Granite. 2025. Granite 3.3 8B Instruct Model Card. <https://huggingface.co/ibm-granite/granite-3.3-8b-instruct/tree/51dd4bc2ade4059a6bd87649d68aa11e4fb2529b>. Model release.
- [12] Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. 2025. The Task Shield: Enforcing Task Alignment to Defend Against Indirect Prompt Injection in LLM Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vienna, Austria, 29680–29697. doi:10.18653/v1/2025.acl-long.1435
- [13] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From Word Embeddings To Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 957–966. <https://proceedings.mlr.press/v37/kusnerb15.html>
- [14] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuan-yu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024. AgentBench: Evaluating LLMs as Agents. In *International Conference on Learning Representations*. 52989–53046. [https://proceedings.iclr.cc/paper\\_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf](https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf)
- [15] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In *Advances in Neural Information Processing Systems*, Vol. 37. Curran Associates, Inc., 74325–74362. doi:10.52202/079017-2365
- [16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=rJzIBfZAb>
- [17] Mistral AI. 2024. Mistral-7B-Instruct-v0.3 Model Card. <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3>. Pinned revision c170c708c41dac9275d15a8fff4eca08d52bab71.
- [18] John X. Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020. Reevaluating Adversarial Examples in Natural Language. In *Findings of the Association*

- for Computational Linguistics: EMNLP 2020. Association for Computational Linguistics, Online, 3829–3839. doi:10.18653/v1/2020.findings-emnlp.341
- [19] John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 119–126. doi:10.18653/v1/2020.emnlp-demos.16
- [20] Joon Sung Park, Joseph C. O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, 1–22. doi:10.1145/3586183.3606763
- [21] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410
- [22] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 68539–68551. doi:10.52202/075280-2997
- [23] Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. 2025. Progent: Securing AI Agents with Privilege Control. arXiv:2504.11703 <https://arxiv.org/abs/2504.11703>
- [24] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 16857–16867. [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/c3a690be93aa602e2dc0ccab5b7b67e-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/c3a690be93aa602e2dc0ccab5b7b67e-Paper.pdf)
- [25] Rodrigo Toro Icarte, Torny Klassen, Richard Valenzano, and Sheila McIlraith. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 2107–2116. <https://proceedings.mlr.press/v80/icarte18a.html>
- [26] Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions. arXiv:2404.13208 <https://arxiv.org/abs/2404.13208>
- [27] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research* (2024). <https://openreview.net/forum?id=H2EeStRTQn>
- [28] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv:2212.03533 <https://arxiv.org/abs/2212.03533>
- [29] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 5776–5788. [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf)
- [30] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muenighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 641–649. doi:10.1145/3626772.3657878
- [31] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In *Advances in Neural Information Processing Systems*, Vol. 37. Curran Associates, Inc., 52040–52094. doi:10.52202/079017-1650
- [32] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong

- Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 Technical Report. arXiv:2505.09388 <https://arxiv.org/abs/2505.09388>
- [33] An Yang, Baosong Yang, Beichen Zhang, et al. 2024. Qwen2.5 Technical Report. arXiv:2412.15115 <https://arxiv.org/abs/2412.15115>
- [34] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2025.  $\tau$ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In *International Conference on Learning Representations*. 9965–10017. [https://proceedings.iclr.cc/paper\\_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf](https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf)
- [35] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. [https://openreview.net/forum?id=WE\\_vlUYUL-X](https://openreview.net/forum?id=WE_vlUYUL-X)
- [36] Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. 2025. Adaptive Attacks Break Defenses Against Indirect Prompt Injection Attacks on LLM Agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*. Association for Computational Linguistics, Albuquerque, New Mexico, 7116–7132. doi:10.18653/v1/2025.findings-naacl.395
- [37] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*. Association for Computational Linguistics, Bangkok, Thailand, 10471–10506. doi:10.18653/v1/2024.findings-acl.624
- [38] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2025. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents. In *International Conference on Learning Representations*. 35331–35366. [https://proceedings.iclr.cc/paper\\_files/paper/2025/file/5750f91d8fb9d5c02bd8ad2c3b44456b-Paper-Conference.pdf](https://proceedings.iclr.cc/paper_files/paper/2025/file/5750f91d8fb9d5c02bd8ad2c3b44456b-Paper-Conference.pdf)
- [39] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*. 15585–15606. [https://proceedings.iclr.cc/paper\\_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf](https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf)