
CODING AGENTS AREN'T ENOUGH!

EVALUATING AN ENTERPRISE SECURITY BRAIN FOR AGENTIC CLOUD INVESTIGATIONS

Leon Goldberg, Gal Engelberg, Eden Yavin, Elad Elouz, Ariel Zadok, Konstantin Koutsi

Sola Security, Tel Aviv, Israel

Corresponding author: gal.e@sola.security

ABSTRACT

Cloud-security investigation is dominated by *population* tasks: which identities can read a data store, how many resources fail a control, which assets are reachable from another account. These resolve against a complete inventory, not a named object. A partial answer to one is not a partial result. It is a different result. General-purpose coding agents can now be given read-only cloud credentials and asked to investigate directly, which raises the question of what a purpose-built security context layer still contributes. We evaluate the Sola Security Brain, a security intelligence layer whose relational substrate is resolved offline and whose security logic is evaluated against it at query time, against Claude Code operating the same live AWS environment through a read-only CLI, over 28 cloud-security investigation tasks. Answers are scored by a blinded, tier-weighted, grounding-gated *relative recall* over the joint claim pool, averaged across three independent grading draws. The Sola Security Brain reaches 0.693 coverage against 0.387, a gap of 0.306 that varied by ± 0.018 across three grading draws, or a relative gain of 79.2%. It leads on 25 of 28 tasks from the weaker model tier, at $17.7\times$ lower reasoning cost per task and $31.6\times$ lower cost per unit of coverage. Beyond the aggregate, we describe an answer-level pattern we term *sample-and-generalise*: under a turn budget the live agent enumerates a fraction of a large population, asserts an unhedged universal negative, and discloses the sample size only in answer metadata rather than in the answer. In one task it reported that no bucket policies exist after checking four bucket families, in a sweep that sampled 40 of roughly 5,000 buckets, in an account where 65 buckets carry a wildcard-principal read grant.

Keywords Cloud Security Posture · Agentic AI · Context Engineering · Benchmarking · Security Knowledge Graphs

1 Introduction

A security analyst asking “which roles can read my storage buckets” is not asking for a lookup. The answer is a property of a population: every identity-based grant, every resource policy, every wildcard or administrative entitlement that subsumes the specific one, resolved across an estate that may contain thousands of buckets and tens of thousands of roles. The same is true of most of the working questions in cloud security. *What should I fix first. Which identities give the widest reach if compromised. Which resources are reachable from another account.* Each is an aggregate over a complete set.

Two developments have changed how these questions get answered. First, general-purpose coding agents have become competent operators of cloud APIs: given read-only credentials and a command-line interface, such an agent will plan an investigation, issue calls, follow what it finds, and write up a result. Second, purpose-built security platforms increasingly maintain a *context layer* in which the substrate those relationships depend on, a normalised inventory with entities reconciled and relations validated, is resolved offline into a queryable representation, so that effective access, reachability, blast radius, failing controls and cross-account trust can be evaluated against it when a question arrives. The two approaches answer the same question from different positions: one reconstructs the relevant structure at question time from raw APIs, the other queries a structure that already exists.

That structured context improves LLM performance on security tasks is established, including by our own prior work. The Cross-Vendor Sola ISPM Benchmark held models and tasks fixed and varied only the injected context, and found that structured relational context raised answer correctness by roughly 34% relative while cutting exploration queries by roughly 70% [1]. Independent work reports the same direction in other domains. Supplying runtime topology to a patch-generating model raises correctness on topology-dependent Kubernetes findings from 11.1% to 78.0%, with no movement on a topology-independent control arm [2]. Replacing a flat document store with a typed knowledge graph, holding the model constant, moves an industrial asset-operations agent from 65% to 82–83% [3]. We therefore treat the context effect as established and ask a different question of it.

The open question is *where* that effect comes from on open-ended investigation, and what a system gains from the context it has. Existing agentic security benchmarks put the exploring agent under test rather than against a context layer [4–6]. Coverage deficits in broad information-seeking are documented in general settings [7], and premature stopping is a named failure mode in multi-agent systems [8]. Neither literature examines what an agent *asserts* when its budget runs out with a population only partly enumerated.

In this work we ask how far a context layer closes that gap. We evaluate the Sola Security Brain, in which a security intelligence layer is built offline and evaluated at query time, against Claude Code operating the same live AWS environment through a read-only CLI. The two answer 28 investigation tasks, authored from that environment and validated by a security practitioner. We compare the two on *coverage*: the share of the findings a complete answer would contain that each system actually surfaced.

We score coverage because these tasks are aggregates. What an analyst needs from the answer is how many of the identities holding read access it named, how many failing resources it listed, how many exposed paths it traced. An answer that is correct about the fifth of the population it examined is still wrong about the estate, and a correctness-style measure would score it well; coverage does not.

We measure coverage with pooled relative recall [9], the standard construction for scoring systems against a merged judged set. Each answer is reduced to the atomic factual claims it makes about the environment, and the two systems' claims for a task are merged into one pool, relabelled and shuffled so the judge cannot tell which system produced which. Every claim in the pool is then rated for security consequence and weighted accordingly, so that the score tracks the share of security value surfaced rather than the number of statements made. Claims are validated before they count towards a system's score. Coverage is then a system's tier-weighted, validated share of the pool, and because the pipeline is itself model-driven, every figure we report is the mean of three independent grading draws. Section 4 gives the measure in full and states its limits.

The rest of the paper is organised as follows. Section 2 situates the work against agentic security benchmarks and documented agent coverage and budget failures. Section 3 describes the environment, the task set and the two architectures under comparison. Section 4 specifies the scoring protocol, its provenance in the measurement literature, and the efficiency measures. Section 5 reports coverage, duration and cost, with per-theme and per-task decomposition. Section 6 interprets the result, describes the answer-level pattern we observed behind it, and states the study's limitations. Section 7 concludes.

2 Related Work

Two areas bear directly on this work: benchmarks that place LLM agents in security investigation settings, and documented coverage and budget failures of tool-using agents. The measurement literature our protocol draws on is discussed with the protocol, in Section 4.

ExCyTIn-Bench is the closest structural analogue to our baseline arm: investigation questions derived from graphs over 57 log tables of a controlled Azure tenant, with the agent discovering the schema and investigating through SQL for a best reward of 0.606 [4]. Its graph authors and scores the questions but is never given to the agent, leaving the graph-as-context condition untested there. Others vary the setting: a forensic disk image with product alerts [5], an unhinted event database [6], triage separated from novel-finding discovery [10], a replayed cluster snapshot [11], host-telemetry incident response [12], attack-chain reconstruction scored per reference step, where the strongest configuration recovers 39.6% [13], and threat hunting decomposed into a dependency graph of analytical tasks [14]. All report low absolute performance on open-ended discovery, all put the exploring agent under test rather than against a context layer, and all score against a reference label set, not a pooled finding set.

ITBench instantiates incidents in live Kubernetes clusters, and its compliance and security track is the closest prior case of an agent given live infrastructure for posture work; agents resolve 25.2% of those scenarios [15]. Cost-aware evaluation finds defensive investigation scales poorly with budget: doubling an agent's spend cap moved its defensive score by less than a point while tool calls rose by a tenth [16]. That is our boundary from another direction, the constraint

being what an agent can reach, not how long it thinks. Cloud posture and effective-permissions analysis is otherwise thin, and the substantial coverage-oriented treatment of cloud IAM is not an agent at all: whitebox and greybox analysis over permission-flow templates, benchmarked on whether known privilege-escalation paths are found [17]. Our own prior work is the immediate context here, spanning identity-posture visibility on a live multi-vendor estate [18], a cross-vendor benchmark varying injected context with models and tasks fixed [1], and a shared environment exposing enterprise security state through both a relational snapshot and live vendor APIs [19]. This paper differs in what it compares: not an agent against itself under varying context, but a context layer against a general coding agent on the same environment.

On the second area, coverage failures are documented. More than ten agentic systems score near zero on exhaustive collection over a large item set where humans approach completeness [7]; premature termination and unawareness of stopping conditions are named failure modes [8]; raising a tool-call cap alone does not raise accuracy, since the agent cannot see how much allowance remains [20]; and more searching makes abstention on unanswerable queries worse [21]. Where the answer is a complete set, recall-oriented suites show the gap directly [22], and exhaustive-list benchmarks name both premature stopping and the hedging that inflates recall with low-confidence items [23].

Nearer our observation is work on what an agent *asserts* when its evidence does not cover the question. Frontier agents asked to review a file set fail to read all of it in 67.9% of runs and, in 80.4% of those, either claim a complete review or leave the gap undisclosed, missing planted defects at roughly 1.8 times the rate of agents that read everything [24]. The epistemic error is named and measured in static settings as *over-closure*, answering a universal query negatively on evidence that does not cover its scope [25]. False absence claims are a category in hallucination detection for tool-using agents [26], search agents terminate with constraints unverified [27], agents read the lack of an immediate match as proof of non-existence [28], and frontier models reason poorly about when to decline [29].

Our contribution on this point is specific to security. None of that work is set in a domain where the population is an organisation's live resource inventory and the universal negative states that a class of exposure does not exist, and the asymmetry matters: a file missed in code review is caught on the next pass, whereas an exposure reported as absent is not revisited. Nor does any of it examine where the disclosure goes. Overclaiming is measured as whether the gap was disclosed at all; we observe agents that do disclose the sample size, into answer metadata, not into the sentence carrying the claim.

3 Experimental Setup

This section describes the two systems under comparison and the conditions they were run under. Section 3.1 describes the enterprise security brain and what its representation makes available at inference time, Section 3.2 the coding-agent baseline, and Section 3.3 the environment, the task set and the run configuration. Neither architecture is described here as a systems specification; nothing in the evaluation depends on implementation detail.

3.1 The enterprise security brain

A security brain is a context layer over an organisation's security-relevant state. It is the representation findings are derived from, built so that any question about the environment can be answered from it, including questions no policy anticipated. The conceptual architecture is set out in [30, 31]; we summarise the parts that bear on this evaluation. It has two cooperating layers, one that builds the representation and one that scores it.

The *modelling* layer builds the representation. Provider-specific extractors normalise heterogeneous API responses into a vendor-agnostic tabular form held in an immutable, versioned store, so that any downstream computation is reproducible against an explicit snapshot. Those records are then lifted into a typed property graph. Its nodes are assets: identities, workloads, data stores, network endpoints, code artefacts. Its edges are the relations between them, including ownership, identity assumption, data flow, network reachability, configuration attachment and control attestation. Edges arise three ways: deterministic schema-driven rules encoding known intra-vendor semantics, joins on normalised identifiers that bridge otherwise disconnected vendor silos, and model-assisted synthesis in which candidate relations are proposed from the joint schema and then validated against actual data before acceptance. Assets from different vendors are connected through typed relations rather than isolated in per-vendor silos.

The graph carries more than inventory and topology. Findings from scanners and third-party security products are attached to the assets they concern, so that a finding is a property of a modelled asset rather than a row in a separate queue, and its vendor-assigned severity can be refined against the context the graph supplies. Semantic attributes describe what an asset is to the organisation rather than what it is technically: its business function, the criticality of the data it holds, the environment it belongs to. Configuration anomaly scores, measuring how far an asset's configuration departs from that of comparable assets, attach to the same assets. Above all of this sits derived security knowledge, the

control predicates, attack-vector traversals and blast-radius patterns, held as executable specifications, not materialised results and evaluated against the underlying data at inference time.

The *scoring* layer turns that representation into calibrated signals, and is deliberately conservative about where model judgement enters. Intrinsic exposure is measured from observed weaknesses and exploitability patterns, while contextual importance describes what the asset is to the organisation. The two combine multiplicatively. The model is not permitted to assign a final score; it adjusts intermediate semantic variables that a deterministic aggregation then consumes.

What is resolved before a question arrives is therefore the substrate rather than the answer. An agent querying this layer still investigates: it plans, queries, follows what it finds, and refines. What it does not have to do is discover a schema, reconcile the same principal appearing under four different identifiers in four systems, infer that one entitlement subsumes another, or invent a traversal for reachability. That work is done once, not per question.

3.2 The coding-agent architecture

Several general-purpose coding agents can be pointed at live infrastructure in the way this arm requires. Command-line agents such as Claude Code [32], Codex CLI [33] and Gemini CLI [34] run in a terminal with shell access. Editor-integrated agents such as Cursor [35] and GitHub Copilot's agent mode [36], and open-source agents such as Aider [37] and OpenHands [38], expose the same underlying loop of plan, act, observe and revise. They differ in interface, in the model behind them and in how tool use is mediated, but not in the architectural position they occupy: none maintains a model of the environment between questions, and each rebuilds whatever structure a task needs from the tools available to it at the time. We evaluate Claude Code [32], one of the most widely used agents of this kind, as a representative of the class.

The tested configuration is the agent given read-only credentials for the environment and the cloud provider's command-line interface as its tool. It receives the task text and nothing else: no schema, no inventory, no indication of where to look. From there it reconstructs whatever structure the question needs at the moment it is asked, discovering what exists by listing it, deciding which objects are worth inspecting, issuing the calls, and assembling the relationships itself, all within a finite budget of turns and time.

Two properties follow from that position. The agent reads current state directly rather than a representation resolved earlier, so what it returns reflects the environment as it stands at the moment of the call. And it is not restricted to relations someone anticipated modelling: any relationship derivable from the API surface is available to it, whether or not a schema was designed to hold it.

The budget is the binding constraint. Where a question ranges over a large population, the available turns and time bound how many members of that population can be inspected, so the agent selects a subset and decides where to spend its calls.

3.3 Environment and task set

The evaluation environment is a real-life enterprise AWS environment: a live, multi-account estate of the kind an analyst works in daily, not a synthetic or seeded lab. It contains no production customer data. Both systems were given read-only access to it and answered the same tasks against it.

We elicited the tasks in two stages. A task-authoring agent, separate from either system under evaluation, surveyed the environment's data and proposed candidate tasks grounded in what was actually present; a human security practitioner then reviewed each candidate for relevancy, discarding tasks that were artificial, redundant or unrepresentative of analyst work. The task set was fixed before any run and left unchanged throughout. Both systems received identical task text, with no hints as to schema, expected answer shape, or where to look. The phrasing is deliberately that of an analyst, not a benchmark: short, natural, and free of completeness-signal words that would cue either system to enumerate. Table 1 lists all 28 tasks across six themes.

The six themes cover the working surface of cloud-security analysis.

- **Risk triage.** Ranking problems: riskiest resources, identities, compute, storage, resource groups, secrets. Producing an ordering means scoring every candidate on the same basis, so a system that inspects a subset does not return a worse ranking. It returns a ranking of the wrong set. Risk is compositional too, depending on exposure, entitlement and what the asset is to the organisation, which means the answer draws on attributes that originate in different sources and have to be joined before anything can be sorted.
- **Secrets and keys.** Who can read a secret, how many principals can, which secrets a compromised workload exposes. Read access is rarely granted directly. It arrives through role assumption, through resource policies,

Table 1: The 28 investigation tasks.

Theme	<i>n</i>	Tasks
Risk triage	7	q01 What are my riskiest resources right now, and why? q02 Who are my top 10 riskiest IAM identities? q03 What are my riskiest compute assets? q04 What are my riskiest storage assets? q05 What should I fix first? q06 What are my highest-risk resource groups, and why? q07 What are my riskiest secrets and encryption keys?
Secrets and keys	4	q08 Which secrets can more than one identity read? q09 Rank my secrets by how many principals can read them. q10 If a workload is compromised, which secrets does it expose? q11 Which internet-facing workloads can reach my secrets?
Access and reachability	5	q12 Which IAM roles can read my S3 buckets? q13 Who can decrypt my KMS keys? q14 Which principals have wildcard access, and what can they reach? q15 Which Lambda functions can be invoked, and by whom? q16 For my riskiest role, what can it reach?
Blast radius	4	q17 Rank my resources by blast radius. q18 If my most exposed resource is compromised, how far does it reach? q19 Walk me through my riskiest cluster. q20 Which identities give the widest reach if compromised?
Control failures	4	q21 Which resources fail public-access controls, broken down by control? q22 Rank my resources by number of failing controls. q23 Which of my S3 buckets are publicly accessible? q24 Which resources fail encryption-at-rest?
Exposure and cross-account	4	q25 What are my most internet-exposed assets? q26 Which exposed resources can reach sensitive data? q27 What are the attack paths to my privileged roles? q28 Which resources are reachable from another account?

and through wildcards that subsume the specific grant, so counting readers means resolving entitlements to their effective population.

- **Access and reachability.** The effective-permissions problem. Access is the union of identity-based and resource-based grants, modified by conditions and subsumption, and that union is materialised nowhere; it has to be computed. An answer that checks one mechanism is not partial but incorrect.
- **Blast radius.** How far a compromise reaches, which is what makes something urgent. Reach is transitive, so the answer is a graph traversal, and error here is asymmetric: an underestimate de-prioritises exactly the asset that matters most.
- **Control failures.** Which resources fail public-access controls broken down by control, which rank highest by number of failing controls, which fail encryption at rest. These are explicitly population questions. The answer is a count or a complete list, so sampling changes the answer instead of degrading it. Each control is also a predicate over configuration that several settings can satisfy or defeat.
- **Exposure and cross-account.** The external attacker's view. Exposure is a property of a chain spanning network reachability, identity and data placement, and cross-account trust crosses the boundary of the account being inspected, so the relations that matter are the ones least likely to be visible from any single API.

4 Evaluation Protocol

Both arms answered the same 28 tasks against the same environment, read-only, and we scored final answers, not reasoning traces. Table 2 states the configuration of each.

The two arms therefore differ in model tier as well as in architecture, with the baseline on the stronger class. We return to this in the limitations.

Table 2: Configuration of the two arms.

Arm	Model	Access	Run conditions
Sola Security Brain	Claude Sonnet 4.6	The context layer, queried	One process per task, five concurrent, warmed service, five-task warmup
Claude Code	Claude Opus 4.8	Read-only credentials and the provider command-line interface	Prior run reused verbatim, so the same answers are scored across evaluations

Coverage is measured by pooled relative recall over the claims the two systems make. Each step below is carried out by a language-model judge, and the whole chain is run three times [39].

1. *Extraction*. Each answer is reduced to the set of atomic factual claims it makes about the environment [40, 41].
2. *Blind pooling*. The two claim sets for a task are merged into one joint pool [42, 43], labelled only SET-X and SET-Y in randomised order, so the judge cannot determine which system produced which set. A claim asserted by both systems appears once in the pool but in both arms’ claim sets, so $C_{\text{sola}} \cap C_{\text{cc}}$ is generally non-empty and the two coverages need not sum to one. Order is randomised against judge position bias [44, 45].
3. *Tier weighting*. Every claim in the pool is rated for security consequence and weighted on a three-level scale: 1.0 for a claim decisive to the answer, 0.5 for one that is secondary or supporting, and 0 for one that is immaterial. Claims the two answers contradict each other on are neutralised where the disagreement is a matter of analysis rather than of configuration. The score therefore measures the share of security *value* surfaced, not the number of claims counted [46, 47], and is graded rather than binary [48].
4. *Grounding*. A claim earns its weight only if the object it names is real rather than fabricated, and is actually present in that arm’s answer. This check runs on text scrubbed of arm fingerprints, not on relabelled sets alone [49].

For an arm $a \in \{\text{sola}, \text{cc}\}$, denoting the Sola Security Brain and Claude Code arms, with claim set C_a over the joint pool $C = C_{\text{sola}} \cup C_{\text{cc}}$, with tier weight $w(c)$ and grounding gate $g_a(c) \in \{0, 1\}$, coverage is

$$\text{Cov}(a) = \frac{\sum_{c \in C_a} w(c) g_a(c)}{\sum_{c \in C} w(c)}. \quad (1)$$

The measure is *relative recall* [9]: each system’s share of a merged judged set, which is the standard construction where no enumerable gold set exists. It is not recall against the environment, and we do not report it as such. A claim that both systems should have made, and neither did, is invisible to it [50, 51]; we take this up in the limitations. The four-step chain is itself model-driven, so every figure reported is the mean of three independent draws, each with its own blind assignment and seed. The between-arm gap varied by ± 0.018 across the three.

Alongside coverage we record two efficiency measures. Duration is wall-clock seconds per task, end to end. Cost prices each arm on effective reasoning tokens, the context it carries forward, at standard published rates. The Sola arm is priced on accumulated reasoning context at the Sonnet 4.6 rate of \$3/M, the baseline on output tokens at the Opus 4.8 rate of \$15/M. The $5\times$ rate ratio holds at both input and output tiers, so the pricing interpretation does not change the ratio between arms. The tokens priced are question-time inference tokens for both arms.

Both measures are also reported normalised by the coverage they bought, as total seconds $\div$ total coverage and total dollars $\div$ total coverage. We normalise on suite totals rather than averaging per-task ratios, because a ratio with coverage in the denominator is unstable on tasks where an arm scores low.

5 Results

Coverage. Headline results are given in Table 3. Mean coverage is **0.693** against **0.387**, a gap of +0.306 or +79.2% relative, with the gap varying by ± 0.018 across the three grading draws. The Sola Security Brain leads on 25 tasks, Claude Code on 3, with no ties. The three Claude Code leads are all narrow, none exceeding 0.05; the Sola Security Brain’s widest margins are considerably larger, reaching +0.76.

Coverage by theme. Coverage by theme is given in Table 4, as three-draw means. The Sola Security Brain leads all six themes, from +49% on secrets and keys to +117% on exposure and cross-account reach, and leads every task in

Table 3: Headline results.

#	Measure	Sola Security Brain	Claude Code	Result
1	Coverage (relative recall)	0.693	0.387	+79.2%, leads 25/28
2	Duration, mean	193.8s	172.2s	+13%
3	Duration per unit coverage	279s	445s	1.6× faster
4	Cost per task	\$0.0073	\$0.1297	17.7× cheaper
5	Cost per unit coverage	\$0.0106	\$0.3352	31.6× cheaper

Table 4: Coverage by theme.

Theme	<i>n</i>	Sola Security Brain	Claude Code	Gap	Tasks led
Risk triage	7	0.701	0.380	+0.321 (+84%)	7/7
Secrets and keys	4	0.641	0.431	+0.210 (+49%)	3/4
Access and reachability	5	0.693	0.389	+0.304 (+78%)	5/5
Blast radius	4	0.708	0.348	+0.360 (+103%)	3/4
Control failures	4	0.669	0.435	+0.234 (+54%)	3/4
Exposure and cross-account	4	0.743	0.343	+0.400 (+117%)	4/4
All tasks	28	0.693	0.387	+0.306 (+79.2%)	25/28

three of them. The theme ranges do not overlap: the Sola Security Brain’s lowest theme is 0.641 and Claude Code’s highest is 0.435. The three Claude Code leads fall in secrets and keys, control failures, and blast radius.

Per-task coverage. Per-task results are given in Table 5, as three-draw means with the task text verbatim, sorted by the Sola Security Brain’s advantage. Gaps range from +0.76 to −0.04. Twelve tasks show a gap of 0.40 or more, two between 0.30 and 0.39, eleven between 0.01 and 0.29, and three are negative. The Sola Security Brain scores between 0.461 and 0.880 and exceeds 0.70 on 15 of 28 tasks; Claude Code scores between 0.120 and 0.680 and reaches 0.70 on none. Claude Code falls below 0.40 on 15 of 28 tasks, the Sola Security Brain on none.

Duration. Durations are given in Table 6. On raw wall-clock the Sola Security Brain is modestly slower: +13% on the mean, with a median paired ratio of 1.22, and was faster on 8 of 28 tasks. It is the steadier arm: standard deviation 49.9s against 69.8s, roughly 30% lower, over a tighter range. We report arm-level means and the paired ratio, without a paired statistical test. Run conditions differ between the arms, since the Sola arm ran five processes concurrently against a warmed service while the reused baseline’s conditions are not recorded, so the duration comparison carries this caveat where coverage does not.

Raw wall-clock prices two different quantities of work at the same rate. Normalised by the coverage it bought, the ordering reverses. The Sola Security Brain spent 5,426 seconds to deliver 19.40 units of coverage, or **279s per unit**, against Claude Code’s 4,822 seconds for 10.84 units, or **445s per unit**: per unit of coverage the Sola Security Brain is 1.6× **faster**. A system that answers a population task by enumerating the population will spend longer than one that samples and stops, and the coverage column is where that time reappears.

The Sola arm completed every task: 28 of 28 finished, with no empty answers and no format or validation retries.

Cost and coverage-normalised efficiency. Costs are given in Table 7. The ratio decomposes cleanly: 3.5× fewer reasoning tokens, consistent with a context-graph loop reasoning over compact retrieved facts rather than a growing shell transcript, multiplied by the 5× lower model rate. Claude Code’s actual recorded output spend is \$0.5849 per task, \$16.38 across the suite; we report the conservative normalised figure above. Per task the Sola Security Brain is between 9× and 71× cheaper, and cheaper per unit of coverage on all 28 tasks including the three it lost.

6 Discussion

The widest margins cluster on one task shape: *which, how many, rank these*. Blast-radius ranking, remediation ordering, secrets by reader count, widest identity reach, internet exposure. These are properties of a population, not of a resource, and answering them means enumerating the complete set: unioning identity-based grants with resource policies,

Table 5: Per-task coverage, sorted by advantage.

#	Task	Sola Security Brain	Claude Code	Gap
q17	Rank my resources by blast radius.	0.880	0.120	+0.76
q05	What should I fix first?	0.809	0.191	+0.62
q09	Rank my secrets by how many principals can read them.	0.818	0.200	+0.62
q03	What are my riskiest compute assets?	0.818	0.232	+0.59
q20	Which identities give the widest reach if compromised?	0.878	0.351	+0.53
q25	What are my most internet-exposed assets?	0.798	0.276	+0.52
q14	Which principals have wildcard access, and what can they reach?	0.838	0.355	+0.48
q24	Which resources fail encryption-at-rest?	0.825	0.378	+0.45
q12	Which IAM roles can read my S3 buckets?	0.760	0.316	+0.44
q13	Who can decrypt my KMS keys?	0.788	0.358	+0.43
q02	Who are my top 10 riskiest IAM identities?	0.764	0.364	+0.40
q28	Which resources are reachable from another account?	0.728	0.329	+0.40
q27	What are the attack paths to my privileged roles?	0.804	0.443	+0.36
q26	Which exposed resources can reach sensitive data?	0.642	0.323	+0.32
q23	Which of my S3 buckets are publicly accessible?	0.725	0.430	+0.29
q01	What are my riskiest resources right now, and why?	0.643	0.415	+0.23
q11	Which internet-facing workloads can reach my secrets?	0.645	0.421	+0.22
q21	Which resources fail public-access controls, broken down by control?	0.666	0.442	+0.22
q04	What are my riskiest storage assets?	0.585	0.390	+0.19
q19	Walk me through my riskiest cluster.	0.597	0.403	+0.19
q06	What are my highest-risk resource groups, and why?	0.556	0.391	+0.17
q16	For my riskiest role, what can it reach?	0.550	0.451	+0.10
q15	Which Lambda functions can be invoked, and by whom?	0.529	0.467	+0.06
q07	What are my riskiest secrets and encryption keys?	0.733	0.680	+0.05
q08	Which secrets can more than one identity read?	0.528	0.519	+0.01
q10	If a workload is compromised, which secrets does it expose?	0.573	0.582	-0.01
q22	Rank my resources by number of failing controls.	0.461	0.490	-0.03
q18	If my most exposed resource is compromised, how far does it reach?	0.475	0.517	-0.04

Table 6: Duration per task.

	Sola Security Brain	Claude Code
Mean	193.8s	172.2s
Median	198.4s	145.0s
Std. dev.	49.9s	69.8s
Min / Max	81.8s / 303.4s	115.0s / 409.0s
Suite total	5,426s	4,822s
Per unit of coverage	279s	445s
Advantage	Sola 1.6× faster	

applying wildcard and administrative subsumption, resolving grants to their real target populations, sweeping every exposure family. The baseline recomputes fragments of that live from raw APIs and exhausts its budget on breadth.

One pattern recurs in those answers, and we call it **sample-and-generalise**. First, the agent inspects a fraction of a large population. A bounded budget makes this rational, and choosing what to look at is the task, so this part is not a defect. Second, it reports on the whole population rather than on the fraction: the answer states that a property holds of nothing in the environment, not that it held of nothing in the subset examined. The claim is universal, the evidence is partial, and nothing in the sentence marks the difference. Third, the sample size is disclosed, but in a scope-caveats field below the answer rather than in the sentence making the claim. Nothing is concealed, and a reader who looks will find it, but a caveat filed below a conclusion is not weighed against it.

Table 7: Reasoning cost and efficiency.

	Sola Security Brain	Claude Code
Mean reasoning tokens per task	2,448	8,646
Normalised cost per task	\$0.0073	\$0.1297
Normalised cost, 28 tasks	\$0.206	\$3.631
Ratio	-	17.7×
\$ per unit of coverage	\$0.0106	\$0.3352
Seconds per unit of coverage	279s	445s
Coverage per \$	94.4	3.0

Together these produce a failure the reader cannot catch. A fabricated fact can be checked against the environment and found false. A finding that was never surfaced leaves nothing to check: a universal negative reads identically whether the agent enumerated the population or a fraction of it, so a true negative and a false one arrive as the same sentence. We describe this pattern, we do not measure it: our tasks were not stratified by population size, and we report no incidence rate.

Task q12, *which IAM roles can read my S3 buckets*, is the clearest instance. The baseline stated that no bucket policies exist, having checked four bucket families that all returned `NoSuchBucketPolicy`, and concluded there were no public or cross-account grants. The scope-caveats field of the same answer records that 40 of roughly 5,000 buckets were sampled. The answer is internally honest and externally wrong.

S3 read access is granted through several mechanisms, so an answer that checks bucket policies alone misses most of the grants. The Sola arm answered in three tiers: the administrative and power-user tier, which subsumes all S3 read by construction; the managed read-only tier, across 189 matching rows; and the specific grants, an inline Lambda-role family across 1,410 matched rows plus the distinct principals named in bucket policies. Sixty-five buckets carry a `Principal: * read grant`. Five external accounts hold cross-account read, one of them across 53 buckets. Whether any given wildcard grant is reachable depends on block-public-access state and policy conditions, which is triage an analyst can perform. It is not triage they can perform on a category reported as absent.

The baseline leads three tasks, by 0.009, 0.029 and 0.042, and all three fall outside what the context layer models today: chaining a workload through its roles to the secrets it exposes, tracing how far the most exposed resource reaches, and ranking failing controls where the highest-consequence items are individual secrets rather than resource clusters. Where the context is rich, coverage is decisively higher; where it is thin, the two systems are close [52]. The advantage is a property of what has been modelled, not of the architecture in the abstract.

Everything above was measured on a single cloud vendor, which is the setting most favourable to the baseline. AWS is the provider a context-free model knows best, its API surface is uniform, and one set of credentials reaches the whole estate, so the agent can reconstruct structure from a single coherent schema. Real environments are not one vendor. Identity spans a directory, a cloud and a set of SaaS applications; a workload's data reaches a warehouse governed elsewhere; the same principal appears under different identifiers in each system, and no API joins them. Reconciling those entities and inferring the relations between them is the work the modelling layer does offline, and it is work a live agent must repeat per question against schemas it was not designed around. We did not test this, so we state it as an expectation rather than a result: the gap we report should be read as a lower bound, and we would expect it to widen on multi-vendor estates rather than narrow.

That is the case for the layer, and it is an argument about when the work is done rather than about how well either system reasons. Completeness cannot be added at inference time; the representation has to hold the population before the question arrives. Normalisation, entity reconciliation and relation inference are done once and amortised across everything later asked of the layer, where a live agent pays that cost on every task and pays it again on the next. The Sola arm carried 2,448 reasoning tokens per task against 8,646, a factor of 3.5. The remaining factor in the $17.7\times$ cost ratio follows from the model tiers the two arms run on rather than from the architecture.

Several constraints bound what this evaluation supports. Coverage is relative recall over a pool built from the two systems' own output rather than against an external oracle, so it cannot distinguish one arm finding more of what matters from both arms missing the same things, and the denominator is defined by the opponent rather than by the environment [50, 51, 53]. A claim that both systems should have made, and neither did, is invisible to the measure. The pool is also narrow: two systems, one environment. Widening it, with further coding agents and with sources of

claims independent of the systems under test, would make the denominator less dependent on the particular pair being compared and the measure correspondingly more portable.

The clearest next step is more coding agents. One agent is a thin basis for a claim about an architectural class, and adding Codex CLI [33], Gemini CLI [34], Aider [37] and OpenHands [38] would make the baseline a pool rather than a single point, separating what is a property of the position from what is a property of Claude Code [32]. Second, ablation. Holding the task set fixed and varying one thing at a time would attribute the gap to its sources: the model behind each arm, the components of the security brain, and the customisations available to a coding agent. The last matters most for fairness, since a coding agent is not fixed at its default. Skills supply domain procedure on demand [54], and MCP servers connect an agent to systems beyond one vendor's CLI [55]. An agent equipped with security skills and a set of vendor MCP servers is a stronger baseline than the one we ran, and the honest version of this comparison tests against it. Third, an independent reference set: an assessor, or an independent pass over the environment, would propose the claims a complete answer should contain, and coverage reported against that fixed set would convert our principal methodological threat into a measured quantity. Fourth, the graph-disabled ablation, which would isolate the layer's contribution within a single arm.

7 Conclusion

Over 28 cloud-security investigation tasks on one live AWS environment, an enterprise security brain reached coverage of 0.693 against a general coding agent's 0.387, a relative gain of 79.2%. It led on 25 of the 28 tasks, from the weaker model tier and at a seventeenth of the reasoning cost. Raw wall-clock runs 13% against it; per unit of coverage it is $1.6\times$ faster and $31.6\times$ cheaper.

The baseline reasons capably over cloud infrastructure and reports its own limits honestly. Its coverage of population tasks is nonetheless bounded by what it can enumerate within a turn budget, and the more capable model did not lift that bound. Beyond the aggregate, we described an answer-level pattern we call *sample-and-generalise*: partial enumeration, an unhedged universal negative, and a sample size disclosed only in metadata. We did not measure how often it occurs. A false negative and a true negative are the same sentence, so the failure is not detectable from the claim itself, and the disclosure that would expose it sits where a reader will not weigh it. That is what makes it consequential in a domain where the finding that was not surfaced is the one that gets exploited.

What would strengthen the claim is a wider basis: more coding agents in the baseline pool, an ablation that varies the model, the layer's components and the agent's own customisations one at a time, and a reference set of claims drawn independently of the systems under test.

References

- [1] Eden Yavin, Gal Engelberg, Konstantin Koutsyi, Leon Goldberg, and Gal Baron. Cross-vendor sola ISPM benchmark: Evaluating agentic AI for federated identity security reasoning. *arXiv preprint arXiv:2606.02674*, 2026. doi: 10.48550/arXiv.2606.02674.
- [2] Farooq Shaikh. Does runtime topology context improve LLM-generated Kubernetes security patches? *arXiv preprint arXiv:2607.25995*, 2026. doi: 10.48550/arXiv.2607.25995. Workshop on the Use of Large Language Models for Cybersecurity (LLMSec), ESORICS 2026.
- [3] Madhulatha Mandarapu and Sandeep Kunkunuru. Knowledge graphs as the missing data layer for LLM-based industrial asset operations. *arXiv preprint arXiv:2605.26874*, 2026. doi: 10.48550/arXiv.2605.26874. Agents+Graph (AG2026) @ VLDB 2026.
- [4] Yiran Wu, Mauricio Velazco, Andrew Zhao, Manuel Raúl Meléndez Luján, Srisuma Movva, Yogesh K. Roy, Quang Nguyen, Roberto Rodriguez, Qingyun Wu, Michael Albada, Julia Kiseleva, and Anand Mudgerikar. ExCyTIn-Bench: Evaluating LLM agents on cyber threat investigation. *arXiv preprint arXiv:2507.14201*, 2025. doi: 10.48550/arXiv.2507.14201. ICML 2026.
- [5] Lehan Wang, Boli Chen, Ruixue Ding, Pengjun Xie, Jinwei Huang, Zhendong Liu, Shuo Wang, Tao Lei, Xin Ouyang, and Xiaomeng Li. SecRespond: Benchmarking AI agents for real-world post-compromise incident response. *arXiv preprint arXiv:2607.26791*, 2026. doi: 10.48550/arXiv.2607.26791.
- [6] Alankrit Chona, Igor Kozlov, and Ambuj Kumar. Cyber defense benchmark: Agentic threat hunting evaluation for LLMs in SecOps. *arXiv preprint arXiv:2604.19533*, 2026. doi: 10.48550/arXiv.2604.19533.
- [7] Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. WideSearch: Benchmarking agentic broad info-seeking. *arXiv preprint arXiv:2508.07999*, 2025. doi: 10.48550/arXiv.2508.07999.

- [8] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? *arXiv preprint arXiv:2503.13657*, 2025. doi: 10.48550/arXiv.2503.13657.
- [9] Sarah J. Clarke and Peter Willett. Estimating the recall performance of web search engines. *Aslib Proceedings*, 49 (7):184–189, 1997. doi: 10.1108/eb051463.
- [10] Daniel Begimher, Cristian Leo, Jack Huang, Pat Gaw, and Bonan Zheng. SIR-Bench: Evaluating investigation depth in security incident response agents. *arXiv preprint arXiv:2604.12040*, 2026. doi: 10.48550/arXiv.2604.12040.
- [11] Yilun Wang, Guangba Yu, Haiyu Huang, Yujie Huang, Zirui Wang, Pengfei Chen, and Michael R. Lyu. Cloud-OpsBench: A reproducible benchmark for agentic root cause analysis in cloud systems. *arXiv preprint arXiv:2603.00468*, 2026. doi: 10.48550/arXiv.2603.00468.
- [12] Aniket Anand, Yiwei Hou, Daniel Fields, Alex Kantchelian, David Tao, Kurt Thomas, and Grant Ho. Benchmarking and exploring the capabilities of LLMs for attack investigations. *arXiv preprint arXiv:2606.10281*, 2026. doi: 10.48550/arXiv.2606.10281.
- [13] Xuyang Liu, Yibin Han, Zhenwei Zhang, Kai Chang, Zhiwei Xu, Tian Qiu, Weixian Deng, Jiabao Gao, Xiaolin Peng, Hai Wan, and Xibin Zhao. DiagChain: A diagnostic benchmark for evaluating LLM agents on evidence-grounded attack chain reconstruction. *arXiv preprint arXiv:2608.03591*, 2026. doi: 10.48550/arXiv.2608.03591.
- [14] Yuqiao Meng, Luoxi Tang, Feiyang Yu, Xi Li, Guanhua Yan, Ping Yang, and Zhaohan Xi. Benchmarking LLM-assisted blue teaming via standardized threat hunting. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. doi: 10.48550/arXiv.2509.23571.
- [15] Saurabh Jha, Rohan R. Arora, Yuji Watanabe, Takumi Yanagawa, Yinfang Chen, et al. ITBench: Evaluating AI agents across diverse real-world IT automation tasks. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *PMLR*, pages 27134–27197, 2025.
- [16] Paul Kassianik, Blaine Nelson, and Yaron Singer. Beyond success rate: Cost-aware evaluation of offensive and defensive security agents. *arXiv preprint arXiv:2607.15263*, 2026. doi: 10.48550/arXiv.2607.15263.
- [17] Yang Hu, Wenxi Wang, Sarfraz Khurshid, and Mohit Tiwari. TAC: Hybrid IAM privilege escalation detection. *arXiv preprint arXiv:2304.14540*, 2023. doi: 10.48550/arXiv.2304.14540. Revised July 2026.
- [18] Gal Engelberg, Konstantin Koutsyi, Leon Goldberg, Reuven Elezra, Idan Pinto, Tal Moalem, Shmuel Cohen, and Yoni Weintrob. Sola-Visibility-ISPM: Benchmarking agentic AI for identity security posture management visibility. *arXiv preprint arXiv:2601.07880*, 2026. doi: 10.48550/arXiv.2601.07880.
- [19] Gal Engelberg, Michael Arenzon, and Leon Goldberg. Open security benchmark: Towards autonomous enterprise cyber defense. *arXiv preprint arXiv:2607.27288*, 2026. doi: 10.48550/arXiv.2607.27288.
- [20] Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-aware tool use enables effective agent scaling. *arXiv preprint arXiv:2511.17006*, 2025. doi: 10.48550/arXiv.2511.17006. COLM 2026.
- [21] Roy Xie, Deepak Gopinath, David Qiu, Dong Lin, Haitian Sun, Saloni Potdar, and Bhuwan Dhingra. Over-searching in search-augmented large language models. In *Proceedings of the 2026 Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2026. doi: 10.48550/arXiv.2601.05503.
- [22] Mahta Rafiee, Heydar Soudani, Zahra Abbasiantaeb, Mohammad Aliannejadi, Faegheh Hasibi, and Hamed Zamani. Total recall QA: A verifiable evaluation suite for deep research agents. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2026. doi: 10.1145/3805712.3808629.
- [23] Nikita Gupta, Riju Chatterjee, Lukas Haas, Connie Tao, Andrew Wang, Chang Liu, Hidekazu Oiwa, Elena Gribovskaya, Jan Ackermann, John Blitzer, Sasha Goldshtein, and Dipanjan Das. DeepSearchQA: Bridging the comprehensiveness gap for deep research agents. *arXiv preprint arXiv:2601.20975*, 2026. doi: 10.48550/arXiv.2601.20975.
- [24] Nolan Smyth, Yorguin-Jose Mantilla-Ramos, Pascal Jr Tikeng Notsawo, Saskia Helbling, Alberto Tosato, Mohamed Amine Merzouk, Nouha Dziri, Gauthier Gidel, and Tommaso Tosato. Quantifying overclaiming propensity in frontier LLM agents. *arXiv preprint arXiv:2609.20812*, 2026. doi: 10.48550/arXiv.2609.20812.
- [25] Byoungjae Min, Kennedy Edemacu, Sae-Hong Cho, Yoonhyuk Choi, Beakcheol Jang, and Jong Wook Kim. When absence is evidence: Evaluating completeness-sensitive negative reasoning in large language models. *arXiv preprint arXiv:2608.04591*, 2026. doi: 10.48550/arXiv.2608.04591.

- [26] Abhinaba Basu. Tool receipts, not zero-knowledge proofs: Practical hallucination detection for AI agents. *arXiv preprint arXiv:2603.10060*, 2026. doi: 10.48550/arXiv.2603.10060.
- [27] Dayoon Ko, Jihyuk Kim, Sohyeon Kim, Haeju Park, Dahyun Lee, Gunhee Kim, Moontae Lee, and Kyungjae Lee. When is enough not enough? illusory completion in search agents. *arXiv preprint arXiv:2602.07549*, 2026. doi: 10.48550/arXiv.2602.07549.
- [28] Yin hao Tang, Youqing Fang, Yanan Sun, Wenran Liu, Weiming Zhang, Bin Liu, Kuikun Liu, Wenwei Zhang, and Kai Chen. SciExplore: Evaluating autonomous agents from scientific navigation to information integration. *arXiv preprint arXiv:2607.20926*, 2026. doi: 10.48550/arXiv.2607.20926.
- [29] Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J. Bell. AbstentionBench: Reasoning LLMs fail on unanswerable questions. *arXiv preprint arXiv:2506.09038*, 2025. doi: 10.48550/arXiv.2506.09038.
- [30] Gal Engelberg, Leon Goldberg, Konstantin Koutsyi, Boris Plotnikov, Tiltan Gilat, and Ben Benhemo. AI native asset intelligence. *arXiv preprint arXiv:2605.09115*, 2026.
- [31] Leon Goldberg. Context is the architecture: Rethinking security platforms in the LLM era. Sola Security company blog, July 2026. URL <https://medium.com/sola-security/context-is-the-architecture-rethinking-security-platforms-in-the-llm-era-a70e36636787>. Non-peer-reviewed vendor publication. Accessed 22 September 2026.
- [32] Anthropic. Claude code. Command-line coding agent. Documentation, 2026. URL <https://docs.claude.com/en/docs/claude-code/overview>. Accessed 23 September 2026.
- [33] OpenAI. Codex CLI: a lightweight coding agent that runs in your terminal. Software repository, 2025. URL <https://github.com/openai/codex>. Accessed 23 September 2026.
- [34] Google. Gemini CLI: an open-source AI agent that brings gemini into the terminal. Software repository, 2025. URL <https://github.com/google-gemini/gemini-cli>. Accessed 23 September 2026.
- [35] Anyosphere. Cursor: an AI coding agent and editor. Product documentation, 2026. URL <https://cursor.com/>. Accessed 23 September 2026.
- [36] GitHub. GitHub Copilot documentation. Product documentation, 2026. URL <https://docs.github.com/en/copilot>. Accessed 23 September 2026.
- [37] Paul Gauthier. Aider: AI pair programming in your terminal. Software repository, 2026. URL <https://github.com/Aider-AI/aider>. Accessed 23 September 2026.
- [38] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations (ICLR)*, 2025. doi: 10.48550/arXiv.2407.16741.
- [39] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, Ian Soboroff, Hoa Trang Dang, and Jimmy Lin. The great nugget recall: Automating fact extraction and RAG evaluation with large language models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–190, 2025. doi: 10.1145/3726302.3730090.
- [40] Sewon Min, Kalpesh Krishna, Xinxu Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, 2023. doi: 10.18653/v1/2023.emnlp-main.741.
- [41] Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. Long-form factuality in large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. doi: 10.48550/arXiv.2403.18802.
- [42] Karen Sparck Jones and C. J. van Rijsbergen. Report on the need for and provision of an ‘ideal’ information retrieval test collection. Technical Report British Library R&D Report 5266, Computer Laboratory, University of Cambridge, 1975.
- [43] Ellen M. Voorhees and Donna K. Harman, editors. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, Cambridge, MA, 2005. ISBN 0-262-22073-3.
- [44] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36, Datasets and Benchmarks Track*, 2023. doi: 10.48550/arXiv.2306.05685.

- [45] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, 2024. doi: 10.18653/v1/2024.acl-long.511.
- [46] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. doi: 10.1145/582415.582418.
- [47] Ani Nenkova and Rebecca Passonneau. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*, pages 145–152, 2004.
- [48] Jimmy Lin and Dina Demner-Fushman. Will pyramids built of nuggets topple over? In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 383–390, 2006.
- [49] Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*, 2024. doi: 10.48550/arXiv.2404.13076.
- [50] Justin Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, 1998. doi: 10.1145/290941.291014.
- [51] Chris Buckley, Darrin Dimmick, Ian Soboroff, and Ellen Voorhees. Bias and the limits of pooling for large collections. *Information Retrieval*, 10(6):491–508, 2007. doi: 10.1007/s10791-007-9032-x.
- [52] Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation. *arXiv preprint arXiv:2506.05690*, 2025. doi: 10.48550/arXiv.2506.05690.
- [53] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32, 2004. doi: 10.1145/1008992.1009000.
- [54] Anthropic. Agent skills. Claude Platform documentation, 2026. URL <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>. Accessed 24 September 2026.
- [55] Agentic AI Foundation. Model context protocol. Open specification, originally released by Anthropic in 2024, 2026. URL <https://modelcontextprotocol.io/>. Accessed 24 September 2026.