Macrodeep Research

GTM-Bench — Designing an Evaluation for Go-to-Market–Competent Language Models

Evaluation · Jul 2026 · 18 min read

Macrodeep Research Team

Abstract

A benchmark-design note. This piece describes the task taxonomy, grading methodology, and instance format of GTM-Bench. It does not report model scores — the leaderboard is a separate release once runs are complete and reproducible.

Contributions

- Seven task categories — five deterministically graded against a hidden oracle, two rubric-graded
- Execution-based ORCHESTRATE grading inside a deterministic mock-MCP sandbox
- A separate objective-only score that ignores every LLM-judge category
- A containerized, reproducible harness built to accept outside submissions

Why a new benchmark

The language-model field measures what it can grade. MMLU rewards factual recall, GSM8K rewards arithmetic reasoning, SWE-bench rewards producing a patch that makes a hidden test suite pass, and tau-bench rewards multi-turn tool use inside a simulated environment. Each became load-bearing because it converted a fuzzy capability into a signal a machine could score without a human in the loop.

None of them measure whether a model is good at go-to-market work: resolving a half-specified account to the right company record, deciding whether a lead fits an ideal-customer profile, expanding a handful of seed accounts into a correct total-addressable set, choosing the next action in a sequence, or driving a multi-step task across a CRM and a sequencing tool to the correct end state. These are the operations that define GTM competence, and today they are evaluated — when they are evaluated at all — by vibes and cherry-picked demos.

GTM-Bench exists to put an axis under that capability. The design goal is not to produce impressive numbers; it is to build a benchmark that is hard to game, which means the grading has to be as close to execution-based as the domain allows.

Where GTM-Bench sits relative to prior work

It is worth being direct about the landscape, including the parts that are uncomfortable. Almost every

serious benchmark in circulation was authored by a party with an interest in the results. SWE-bench and its curated successor were shaped, in part, by the same labs that ship the coding models measured on them. GAIA, tau-bench, MLE-bench, GPQA — each came out of an organization that also builds models. The “we wrote the eval and we do well on it” structure is not an anomaly to apologize for; it is the default condition of the field. Pretending otherwise would be dishonest.

The distinction that actually matters is not authorship — it is reproducibility and external participation. SWE-bench earned its authority not because its origin was neutral but because anyone could clone the harness, run their own model, and watch third-party systems climb the same board under the same rules. GPQA earned it by being genuinely hard and openly graded. The benchmarks that collapsed under scrutiny were the ones nobody outside the author could reproduce. That is the bar GTM-Bench is built to clear: deterministic grading where possible, a containerized harness anyone can run, and a leaderboard that accepts outside submissions.

So GTM-Bench does not claim to be the first “clean” benchmark. It claims to be a checkable one, in a domain that currently has no rigorous eval at all. Positioned against the closest neighbors: SWE-bench measures code-patch correctness against a repository's own tests; tau-bench measures multi-turn tool use in a simulated customer-service environment; GAIA measures general assistant reasoning across tools and web. GTM-Bench borrows SWE-bench's execution-grading discipline and tau-bench's sandboxed-tool structure, and points both at the operations of go-to-market work — enrichment, qualification, market construction, routing, and orchestration — that none of the existing benchmarks touch.

The core design problem: GTM has no test suite

SWE-bench works because software has an oracle built in. A patch is correct if the repository's own tests pass. There is no argument to have with the grader.

Most GTM tasks have no such oracle. “Write a good cold email” has no test suite; two experienced reps will disagree about the same draft. If a benchmark leans on an LLM judge for the bulk of its score, it inherits the judge's biases and becomes trivially gameable by writing to the judge's preferences rather than to the real objective.

The design response is to engineer verifiability into task selection — weighting the benchmark toward tasks with a deterministic ground truth, and treating judge-graded tasks as a small, clearly separated minority.

Task taxonomy

GTM-Bench splits into seven categories. Five are objectively gradable against a hidden oracle; two are

generative and graded separately under a rubric.

ENRICH — entity resolution and enrichment. Given a partial record, return the correct canonical fields, graded field-by-field. The closest analogue to a unit test in the GTM domain. QUALIFY — ICP fit and tiering, scored as a classification problem (F1) against labels drawn from real closed-won / closed-lost outcomes. TAM — lookalike and total-addressable-market construction, scored as precision and recall at k against a curated target set. ROUTE — next-action and sequencing logic, graded against a defined-correct branch rather than a free-form judgment. ORCHESTRATE — the flagship: multi-step tool use graded on whether the correct end state was reached, in the spirit of SWE-bench's “did the suite go green.”

The two generative categories carry minority weight. COMPOSE — outbound copy, graded by rubric-based pairwise comparison against a held-out reference. SUMMARIZE — call and thread summarization, graded on faithfulness and coverage. Both use a judge model kept separate from any model under test, and both are excluded from the objective-only score.

Worked examples

The taxonomy is only meaningful if each category resolves to a concrete, gradable instance. Each task is a single JSON instance; the `oracle` block shown here for illustration is withheld from the model in the live set and lives only in the harness. Values are illustrative and anonymized.

ENRICH — the model receives a fragment and must return canonical fields; the grader compares field-by-field.

QUALIFY — a classification task graded on label match, with ground truth from real closed-won / closed-lost outcomes.

ORCHESTRATE — the flagship. The model acts across sandboxed tools; the grader checks the resulting end state, not the wording of the response.

Scoring

The score is weighted so the deterministically gradable categories dominate. Of a 100-point total, 75 points come from the objective categories (ENRICH, QUALIFY, TAM, ROUTE, ORCHESTRATE), and the generative categories carry the remainder.

Crucially, GTM-Bench reports two numbers: the composite score and a separate objective-only score that ignores the judge-graded categories entirely. A reader who distrusts LLM-judge methodology — a reasonable position — can look only at the objective-only number and still have a meaningful, un-gameable comparison. The rubric-graded categories are there to say something about generative

quality, not to prop up the headline.

The grading harness

Grading runs in a container, following the SWE-bench discipline of a reproducible environment rather than an ad-hoc script. A model adapter presents each instance, collects the response, and hands it to a category-specific grader.

ORCHESTRATE needs more than a string comparison, so it runs against a mock-MCP sandbox — a controlled environment that exposes CRM-like and sequencing-like tools over the same interface a production agent would see, but every tool is backed by an in-memory fixture. A `crm.update` call mutates sandbox state; a `sequencer.enroll` call appends to a sandbox enrollment list. Nothing touches a real system, and every call is logged with its arguments and its effect.

[Figure: FIG 1] Figure 1. An ORCHESTRATE trajectory in the mock-MCP sandbox. The grader asserts on the resulting state (record stage

Three properties fall out of this design. Determinism — the same sequence of tool calls produces the same state every time, so a model's score does not drift with external conditions. Observability — the full trajectory is recorded, so the harness can grade end state and separately report how the model got there. Isolation — no instance can contaminate another, and no run can have side effects outside its container, which is what lets external parties run the harness safely against their own models.

One open design decision sits here, stated plainly rather than hidden: whether ORCHESTRATE should grade end-state only or also penalize trajectory inefficiency. A model that reaches the right state after twelve redundant tool calls is not equivalent to one that does it in three. End-state-only grading is cleaner and harder to argue with; trajectory-aware grading is more faithful to production. The current position is to grade end-state for the headline and report trajectory efficiency as a separate diagnostic, so the primary score stays un-arguable.

Dataset construction and the contamination question

Task instances are harvested and anonymized from real GTM operations, then scrubbed for PII and brought into GDPR compliance before anything enters the set. The task distribution — what real enrichment, qualification, and orchestration problems look like at scale — is the part that is genuinely hard for others to replicate, and it is what keeps the benchmark grounded in real work rather than synthetic toy tasks.

That same grounding raises the honest hazard: if a model is trained on data that overlaps the source of the instances, it may have seen the answers. GTM-Bench therefore treats contamination policy as a first-class part of the spec, with held-out instance construction and a disclosed policy on train/test overlap. Any category where contamination cannot be ruled out is reported with that caveat attached,

not laundered into the headline number.

On authoring both the model and the benchmark

There is an obvious critique of any lab that publishes a benchmark and also builds a model to run on it: of course your model tops your test. Naming it is the only honest way to handle it. The design answers the critique structurally rather than rhetorically. The score is dominated by categories with an external ground truth (real outcomes, canonical records) the author does not get to define arbitrarily. The objective-only score removes the judge-graded surface where authoring bias would most easily creep in. The harness is reproducible so external parties can verify the numbers themselves. And the leaderboard, when it opens, accepts outside submissions.

A benchmark you can only run yourself is a marketing asset. A benchmark others can run, reproduce, and submit to is research. GTM-Bench is being built to be the second kind.

Known limitations

Stating the weaknesses is part of the design, not a disclaimer bolted on at the end. The sandbox is not the world — a mock-MCP environment is deterministic precisely because it is simplified; a high ORCHESTRATE score is necessary but not sufficient for production readiness. Ground truth in QUALIFY is outcome-derived, and outcomes are noisy — closed-won / closed-lost labels encode luck, timing, and rep quality alongside real signal. The generative categories remain judge-dependent — held to minority weight and excluded from the objective-only score for exactly this reason. A domain benchmark rewards domain-shaped answers — a strong score reflects fit to this distribution of work, not general capability. Static benchmarks decay — the mitigation is a versioned, rotating instance set and a held-out private split used for the authoritative leaderboard.

What this note is and isn't

This is the design. It defines the axis: what GTM competence means, operationally, and how to grade it without a human in the loop for the parts that matter most. It intentionally reports no scores. The leaderboard — frontier models run under this harness, reproducibly — is a separate release, and it will only go out when the numbers are real and someone outside can check them.

References

1. Jimenez et al. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? 2024.

2. Yao et al. tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. 2024.

3. Mialon et al. GAIA: A Benchmark for General AI Assistants. 2023.

4. Rein et al. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. 2023.