

CONVERSATIONAL TASK DISAMBIGUATION OVER TABULAR DATA: LEAKAGE-AWARE FORMULATION, BENCHMARK SUITE, AND TRAINING

Nafiseh Ghoroghchian* Luis Scoccola* Tina Sedaghat
Omid Vaheb Hannah Chen† Dino D’Agostino Keyvan Golestan

Layer 6 AI

ABSTRACT

Conversational task disambiguation over tabular data uses dialogue to resolve missing information about a user’s intended task before producing a solution over tables or databases. Existing evaluation and training lack a leakage-aware foundation. Task success mixes the agent’s disambiguation and solution-generation capabilities and can also reflect *oracle leakage*, that is, information that a user simulator reveals beyond what a real user would. Existing datasets also lack a shared representation of ambiguities and access boundaries. We introduce the notion of an *ambiguous verifiable task*, which formalizes ambiguities and resolutions, decomposing the agent into an asking policy and a solution policy, and the environment into an oracle and verifier. This framework provides baselines and metrics for evaluating task disambiguation separately from solution generation, formal definitions of oracle leakage, judge-free leakage diagnostics, and a training objective for the asking policy. We instantiate the framework in text-to-SQL with *AmbiTab*, a benchmark suite that unifies six ambiguous datasets under a common representation specifying what the agent, oracle, and verifier may access. We evaluate clarification strategies and oracle leakage, and train an asking policy with reinforcement learning. The trained asker improves our disambiguation metrics on all six datasets and task success on five, and our leakage diagnostics measure how training affects oracle leakage.

1 INTRODUCTION

1.1 CONTEXT AND PROBLEM STATEMENT

Language models are increasingly used to solve tasks over tables and relational databases (Tian et al., 2026; Wu et al., 2025; Li et al., 2026). When the solution is an executable query, execution provides a trusted verifier. Reinforcement Learning (RL) using such rewards (Pourreza et al., 2025; Ma et al., 2025; Lei et al., 2025b) has improved single-turn systems on text-to-SQL benchmarks such as BIRD (Li et al., 2023) and Spider 2.0 (Lei et al., 2025a). These benchmarks assume that the task specification determines the intended solution. Real task specifications often do not: for example “list the top earners” leaves open both how many earners to return and what counts as earnings (Figure 1(c)). We study *conversational task disambiguation over tabular data*: an agent asks clarifying questions to resolve missing information about the intended task before producing a solution over tables or databases. We focus empirically on text-to-SQL, where clarification identifies the user’s intended interpretation and the solution policy generates a corresponding query. Conversational benchmarks such as SPaC and CoSQL (Yu et al., 2019b;a) established the value of evaluating complete interactions, and interactive benchmarks such as BIRD-Interact (Huo et al., 2026) show that this type of ambiguous task remains difficult even for capable models.

Evaluating and training conversational task disambiguation requires a leakage-aware foundation, which current practice lacks, for three reasons. First, success rate, the natural metric, depends on

*Contributed equally and are joint first authors.

†This work was done during her internship at Layer 6 AI.

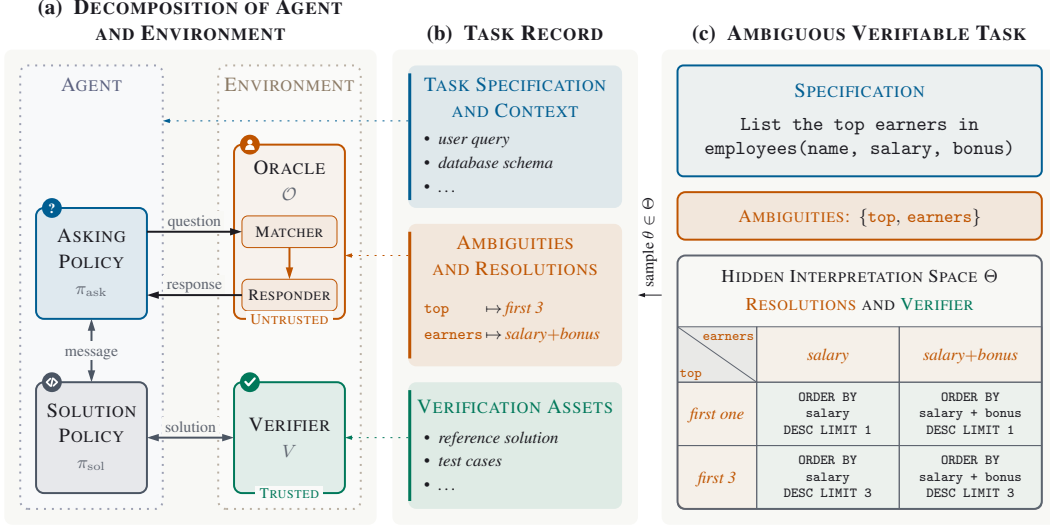


Figure 1: Agent and environment decompositions, task record, and example of a text-to-SQL AVT with two ambiguities and four interpretations.

both asking and solving: a failure may reflect a missing question or a wrong solution, and a success may reflect a task that needs no disambiguation. Second, at scale the user is replaced by an oracle, often a language model, and an agent selected or trained against it may exploit its errors, as it might exploit a misspecified reward (Skalse et al., 2022; Pan et al., 2022), for example by eliciting details of the reference solution that no real user would supply. We call this *oracle leakage*; prior work separates, constrains, or audits user simulators against it (Jain et al., 2025; Huo et al., 2026; Sullutrone et al., 2026), and diagnosing it with a language model judge adds another untrusted component. Both problems carry over to training: a success-based reward can attribute gains to the asker that actually arise from the solver’s capabilities or from oracle leakage. Third, existing datasets lack a shared representation of ambiguities, resolutions, and what the agent, the oracle, and the verifier may access.

We address these difficulties with a formalism that makes explicit what is trusted and what the agent, the oracle, and the verifier may each access. We use it to address the following questions:

- Q1.** How can task disambiguation be evaluated separately from solution generation?
- Q2.** How can oracle leakage be formally defined and diagnosed without relying on an untrusted language model judge?
- Q3.** How can an asking policy be trained within the same formalism, and how does training affect task success rate, asking behavior, and leakage?

1.2 CONTRIBUTIONS

We introduce the notion of an *Ambiguous Verifiable Task* (AVT) (Definition 1), a general representation of underspecified tasks with verifiable solutions. An AVT consists of a public task specification, a probability space of hidden interpretations, a finite set of ambiguities and their resolutions, and a verifier (Figure 1c); nothing in the definition is specific to tables or SQL. We study conversational task disambiguation through an ask-then-solve protocol for AVTs and instantiate it in text-to-SQL over relational databases. In this setup, the agent consists of an *asking policy* (or *asker*) that asks the oracle questions and a *solution policy* (or *solver*) that submits a solution. The environment consists of an *oracle*, whose *matcher* credits the ambiguities each question addresses and whose *responder* returns information about their resolutions, and a trusted *verifier* that scores the solution. Because only the asker interacts with the oracle, we can vary the asker while holding the solver fixed. We address questions **Q1-Q3** through three main contributions:

1. **Theory.** We develop a leakage-aware formalism for AVTs and instantiate it in an agent-environment interaction protocol (Section 2). For **Q1**, it separates asking from solving through no-disambiguation and full-disambiguation baselines and asking precision and recall. Splitting the oracle into a matcher and a responder, as some user simulators already do (Huo et al., 2026; Sullutrone et al., 2026; Zhang et al., 2026; Trinh et al., 2026), separates two leakage channels: crediting ambiguities that a question does not address, and revealing more than the credited resolutions. For **Q2**, we use our framework to give mathematical formulas quantifying the two types of leakage. The formalism also introduces judge-free, computable diagnostics and proves how they witness or approximate leakage under explicit assumptions.
2. **AmbiTab Benchmark Suite.** We introduce AmbiTab for conversational task disambiguation over tabular data, comprising more than 8,000 task records across six ambiguous text-to-SQL datasets (Section 3). AmbiTab-Bird and AmbiTab-EW-Bird derive from BIRD (Li et al., 2023); AmbiTab-BIRD-Interact and AmbiTab-Ambrosia adapt BIRD-Interact (Huo et al., 2026) and Ambrosia (Saparina & Lapata, 2024); and AmbiTab-AmbiQT and AmbiTab-TACO adapt AmbiQT (Bhaskar et al., 2023) and TACO (Deng et al., 2026). All six share a record that separates the public specification, the resolutions available to the oracle, and the private reference used by the verifier (Figure 1b), supporting the metrics for **Q1** and, with the reference-response and sampling procedures in Appendix A.4, the diagnostics for **Q2**. AmbiTab-Bird and AmbiTab-EW-Bird also include training splits relevant to **Q3**.
3. **Experiments.** We address **Q1** by comparing no-disambiguation, evaluated-asker, and full-disambiguation conditions and reporting both success and asking metrics (Section 4.2). Reporting all three success rates together with macro precision and recall distinguishes the benefit of clarification from gaps in question quality. For **Q2**, we apply the diagnostics to deterministic and LLM oracles and test them on a responder that reads the reference SQL (Section 4.3). For **Q3**, we train the asking policy with a multi-turn GRPO variant with DAPO-style asymmetric clipping and zero-advantage group filtering (Shao et al., 2024; Yu et al., 2025) on AmbiTab-Bird, evaluate its coverage and task success with a fixed solution policy across datasets, and compare training oracles and reward objectives (Section 4.4).

1.3 RELATED WORK

Conversational task disambiguation. Ambiguous text-to-SQL benchmarks represent alternative readings (Bhaskar et al., 2023; Saparina & Lapata, 2024; Deng et al., 2026). BIRD-Interact and ABISS evaluate clarification with simulated users (Huo et al., 2026; Sullutrone et al., 2026), whereas DySQL-Bench studies follow-up requests in database exploration (Sun et al., 2026a). Information-gain methods select questions that distinguish interpretations (Kobalczyk et al., 2025; Qiu et al., 2025); IN3, AskBench, and π -Bench assess coverage of missing information (Qian et al., 2024; Zhao et al., 2026; Zhang et al., 2026). We focus on eliciting the missing information that resolves the current task’s intended interpretation. Our task representation makes task ambiguities, resolutions, and oracle access explicit, with text-to-SQL as the evaluated instantiation over tabular data. See Appendix F for broader comparisons of benchmarks, clarification methods, and reward formulations.

Oracle reliability. Prior work restricts solution access or audits simulated responses (Jain et al., 2025; Huo et al., 2026; Sullutrone et al., 2026), but simulator fidelity and policy overfitting remain concerns (Seshadri et al., 2026; Yu et al., 2026). We define leakage relative to a perfect oracle and derive judge-free diagnostics for undeserved question credit and excess response information.

Learning to disambiguate. RL for SQL generation and table reasoning directly rewards solution correctness (Pourreza et al., 2025; Ma et al., 2025; Lei et al., 2025b). ACT and SGP learn clarification policies (Chen et al., 2025; Berant et al., 2026), while IntentRL learns to enumerate interpretations and SQL queries in one response (Saparina & Lapata, 2026). InfoPO combines information gain with outcome rewards (Kong et al., 2026); HiL-Bench combines blocker coverage with per-question bonuses and penalties (Trinh et al., 2026). Our question-F1 reward balances ambiguity coverage and question efficiency without additional shaping coefficients. Training improves downstream accuracy through better disambiguation.

2 THEORY

Agent, environment, and oracle leakage. We model the interactions between an agent $\mathcal{A}$ and an environment $\mathcal{E}$ as a stochastic process from which we sample trajectories $\tau \sim (\mathcal{A}; \mathcal{E})$. Both $\mathcal{A}$ and $\mathcal{E}$ are conditional distributions: $\mathcal{A}$ models the probability distribution of the next action given the observable history, and $\mathcal{E}$ that of the next state given the history and the action taken by $\mathcal{A}$. The sets of states and actions depend on the setup.

We are interested in situations where the environment is determined by two kinds of distributions: (1) Trusted distributions, such as a programmatic verifier V , which could even be deterministic. (2) Untrusted distributions, such as an *oracle* $\mathcal{O}$ implemented by a language model, intended to approximate a *perfect oracle* $\mathcal{O}^*$, which could be defined to be, e.g., the behavior of a user base.

The agent is usually evaluated using its *success rate* $J(\mathcal{A}; \mathcal{E}) := \mathbb{E}_{\tau \sim (\mathcal{A}; \mathcal{E})}[V(\tau)]$. However, maximizing $J(\mathcal{A}; \mathcal{E})$ can lead the agent to exploit deviations of $\mathcal{O}$ from $\mathcal{O}^*$, e.g., by eliciting information from $\mathcal{O}$ that $\mathcal{O}^*$ would not provide. We refer to any such deviation as *oracle leakage*.

Ambiguous verifiable tasks. We represent task specifications, ambiguities, clarification questions, responses, and solutions as text; we write Text for this set. Task disambiguation elicits information about the intended interpretation of a specification. In our text-to-SQL instantiation, this interpretation determines what the user intends to compute over the database, and the solution is a corresponding SQL query (Figure 1c).

Definition 1. An *ambiguous verifiable task* is a tuple $(x, \Theta, \text{Amb}, \rho, V)$ consisting of

- A *specification* $x \in \text{Text}$;
- A probability space Θ of *interpretations* of x ;
- A finite set $\text{Amb} \subseteq \text{Text}$ of *ambiguities*;
- A *resolution map* $\rho: \Theta \times \text{Amb} \rightarrow \text{Text}$;
- A *verifier* $V: \Theta \times \text{Text} \rightarrow \{0, 1\}$.

Proofs, technical assumptions, a table of notation, and other details are in Appendix A.

Leakage-aware evaluation of agents on ambiguous verifiable tasks. Figure 2 defines the ask-then-solve protocol, a simple interaction protocol between an agent and an environment associated to an AVT. This setup can be generalized to other interaction protocols.¹

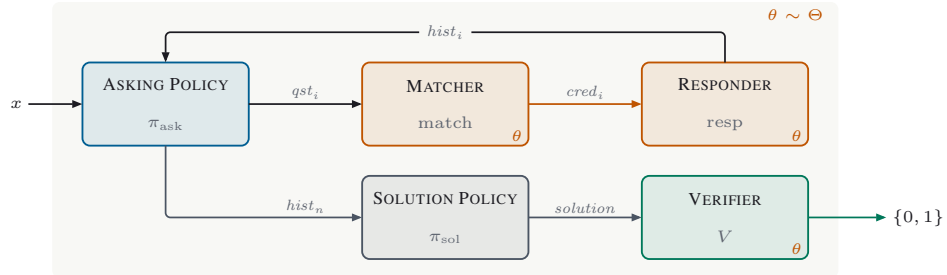


Figure 2: The *ask-then-solve protocol* (Appendix A.2). Oracle leakage can arise from undeserved matcher credit or responses revealing more than the credited resolutions.

We now define success metrics², leakage, and leakage diagnostics for the ask-then-solve protocol associated to an AVT (Figure 2). See Appendix A.3 for formal definitions, and Appendix A.4 for estimation. The definitions and theoretical guarantees below concern this protocol.

¹For context, note that an AVT equipped with an interaction protocol determines a partially observable Markov decision process: the state is the hidden interpretation θ plus the dialogue history, and the agent observes the history. Since θ is drawn once and fixed, this is the mixed-observability case of Ong et al. (2010).

²These parallel precision and recall in HiL-Bench (Trinh et al., 2026); in our protocol, these evaluate π_{ask} independently of π_{sol} . Our no- and full-disambiguation baselines, below, parallel their blocked and full-information conditions. Checkpoint coverage in AskBench (Zhao et al., 2026) is related.

Since we care about the interactions between the asker and the oracle, we fix a solution policy, a verifier, and a perfect oracle $\mathcal{O}^* = (\text{match}^*, \text{resp}^*)$. We abbreviate $(\pi_{\text{ask}}, \pi_{\text{sol}}; \mathcal{O}, V)$ as $(\pi_{\text{ask}}; \mathcal{O})$.

SUCCESS METRICS AND BASELINES

- *Success rate.* $J(\pi_{\text{ask}}; \mathcal{O}) := \mathbb{E}[V(\theta, \text{solution})]$.
- *Recall and Precision.* In an episode with n questions, the matcher credits a set $\text{cred}_i \subseteq \text{Amb}$ of ambiguities to question i ; write $\text{cred}_{\leq i} := \bigcup_{j=1}^i \text{cred}_j$, so that $\text{cred}_{\leq 0} = \emptyset$.

$$\text{Rec}(\pi_{\text{ask}}; \mathcal{O}) := \mathbb{E} \left[\frac{|\text{cred}_{\leq n}|}{|\text{Amb}|} \right] \quad \text{Prec}(\pi_{\text{ask}}; \mathcal{O}) := \mathbb{E} \left[\frac{|\{i : \text{cred}_i \not\subseteq \text{cred}_{\leq i-1}\}|}{n} \mid n \geq 1 \right].$$

- *Full-disambiguation and no-disambiguation baselines.* Fix a *full-disambiguation policy* π_{full} , which acquires every ambiguity and asks no unproductive question ($\text{Rec}(\pi_{\text{full}}; \mathcal{O}^*) = \text{Prec}(\pi_{\text{full}}; \mathcal{O}^*) = 1$), and let π_{none} stop without asking any questions.

$$J_{\text{full}}(\mathcal{O}) := J(\pi_{\text{full}}; \mathcal{O}) \quad J_{\text{none}}(\mathcal{O}) := J(\pi_{\text{none}}; \mathcal{O}).$$

Recall that, by *oracle leakage*, we mean any exploitation by the agent of deviations of $\mathcal{O}$ from $\mathcal{O}^*$. This motivates the following definitions³:

LEAKAGE

1. *J-leakage.* $\text{Leak}_J(\pi_{\text{ask}}; \mathcal{O}) := J(\pi_{\text{ask}}; \mathcal{O}) - J(\pi_{\text{ask}}; \mathcal{O}^*)$.
2. *Recall leakage.* $\text{Leak}_{\text{Rec}}(\pi_{\text{ask}}; \text{match}) := \text{Rec}(\pi_{\text{ask}}; \text{match}, \text{resp}^*) - \text{Rec}(\pi_{\text{ask}}; \mathcal{O}^*)$.

The above notions of leakage cannot generally be computed directly. For example, estimating J -leakage requires trajectories from $(\pi_{\text{ask}}; \mathcal{O}^*)$, which in practice can require an expensive back-and-forth between sampling from the asker and human annotators.

LEAKAGE DIAGNOSTICS

1. *Responder-leakage diagnostic* ΔJ_{resp} . Replaces the actual responder by the perfect responder while holding the asking policy and actual matcher fixed (Definition 5).
2. *Specification-randomization diagnostic* $\Delta \text{Rec}_{\text{spec}}^Q$. Recall after replacing the task specification seen by the asking policy with an independently sampled specification (Definition 6).
3. *Adversarial diagnostics* $\Delta J_{\text{full}}, \Delta \text{Rec}_{\text{none}}$. Evaluate J -leakage under an adversarial policy that covers all of the ambiguities, and recall leakage under one that covers none (Definition 7).

Next is the main result, Theorem 1. Informally: Part 1 says that the responder-leakage diagnostic approximates J -leakage when swapping the actual matcher for the perfect one results in a small change in the distribution of trajectories, as measured by the total variation distance TV (Appendix A.1); and Part 2 says that the specification-randomization diagnostic approximates recall leakage of the randomized policy when its perfect-oracle recall is small, and that a simple mixture model relates this to recall leakage of the original policy. See Appendix A.5 for the proof.

Theorem 1. Fix an AVT, asking policy π_{ask} , solution policy, and oracle $\mathcal{O} = (\text{match}, \text{resp})$.

1. Let P_{match} and P_{match^*} be the joint distributions of the hidden interpretation θ and the complete visible trajectory τ under $(\pi_{\text{ask}}; \text{match}, \text{resp}^*)$ and $(\pi_{\text{ask}}; \mathcal{O}^*)$, respectively. Then

$$|\text{Leak}_J(\pi_{\text{ask}}; \mathcal{O}) - \Delta J_{\text{resp}}(\pi_{\text{ask}}; \text{match}, \text{resp})| \leq \text{TV}(P_{\text{match}}, P_{\text{match}^*}).$$

2. We have $\Delta \text{Rec}_{\text{spec}}^Q(\pi_{\text{ask}}; \text{match}) - \text{Leak}_{\text{Rec}}(\pi_{\text{ask}}^Q; \text{match}) = \text{Rec}(\pi_{\text{ask}}^Q; \mathcal{O}^*)$.

Moreover, if $\pi_{\text{ask}} = p \pi_{\text{inv}} + (1-p) \pi'$ for some $p \in [0, 1]$ and asking policies π_{inv} and π' , where π_{inv} is invariant under replacing the task specification, then

$$\left| \text{Leak}_{\text{Rec}}(\pi_{\text{ask}}; \text{match}) - \text{Leak}_{\text{Rec}}(\pi_{\text{ask}}^Q; \text{match}) \right| \leq 2(1-p^B),$$

where B is the question budget and Q is any distribution over task specifications.

³ J -leakage is a signed, fixed-policy analogue of policy-value gaps used to assess simulators (Dann et al., 2026) and is related to generalized-gain leakage (Alvim et al., 2012). Here the reference is a task-defined perfect oracle, and a positive gap means extra task success enabled by the actual oracle.

3 AMBITAB BENCHMARK SUITE

We introduce AmbiTab, a suite of six ambiguous text-to-SQL datasets for evaluating conversational task disambiguation over tabular data. Each dataset consists of task records with explicit ambiguity annotations, fixed resolutions, and verification assets.

Task record. Abstractly, a task record implements a fixed interpretation of an ambiguous verifiable task $(x, \Theta, \text{Amb}, \rho, V)$, so it can be seen as a sample of an AVT. More precisely, given an interpretation $\theta \in \Theta$, its corresponding *task record* is the tuple $(x, \text{Amb}, \rho_\theta, V_\theta)$, where $\rho_\theta(\alpha) = \rho(\theta, \alpha)$ is the resolution map for an ambiguity α evaluated on the fixed interpretation, and $V_\theta(s) = V(\theta, s)$ is the verifier evaluated on the fixed interpretation.

In practice, this is implemented as follows (see Figure 1(b) for an illustration):

- **Task specification and context.** The user request, and optionally the database schema, the column meanings, external knowledge, and other context.
- **Ambiguities and resolutions.** A list pairing each ambiguity with its resolution as text. Optionally, it might include the ambiguity type, a SQL snippet for grounding the responder, etc.
- **Verification assets.** Usually, an executable database and a reference query. Optionally, test cases, statements that prepare and restore the database, a policy for comparing result sets, etc.

Dataset construction and details. We adapt BIRD (Li et al., 2023), BIRD-Interact (Huo et al., 2026), Ambrosia (Saparina & Lapata, 2024), AmbiQT (Bhaskar et al., 2023), and TACO (Deng et al., 2026) (Appendix B.1). All datasets use SQLite except for AmbiTab-BIRD-Interact, which uses PostgreSQL. Table 1 shows where each corpus’s missing information comes from; Appendix B gives the construction and oracle details.

Table 1: The six AmbiTab datasets. AmbiTab-Bird and AmbiTab-EW-Bird counts combine their train and development splits.

Dataset	Missing information and construction	Records
AmbiTab-Bird	Rewrite BIRD tasks with alternative readings; filter by SQL probes and a model judgment.	664
AmbiTab-EW-Bird	Withhold the BIRD evidence field; retain rows where providing it improves performance for at least one reference model.	2,537
AmbiTab-BIRD-Interact	Normalize source ambiguity annotations; freeze audited, model-authored responses.	600
AmbiTab-Ambrosia	Use each annotated reading and its native intent as a separate record.	2,670
AmbiTab-AmbiQT	Build executable databases for paired SQL readings and assign each reading as a hidden intent.	1,722
AmbiTab-TACO	Verbalize source SQL while withholding table or value choices.	187
Total		8,380

4 EXPERIMENTS

We evaluate task disambiguation in text-to-SQL through task success, asking precision and recall, and oracle leakage across the six datasets in Section 3, varying the oracle, asker prompt, and training configuration.

4.1 EXPERIMENTAL SETUP

Protocol. We use the ask-then-solve protocol of Figure 2, with Qwen3-8B (Yang et al., 2025) as the asker and a fixed Qwen3-Coder-30B-A3B-Instruct solver. The asker clarifies the intended task through dialogue with the oracle; the solver uses the request and dialogue to produce SQL. The default prompt, *Checklist (ledger)*, enumerates potentially ambiguous terms and tracks their resolutions. Appendix C.5 examines the precision–recall tradeoff across asker prompts. We use the deterministic oracle by default and compare it with Qwen3-14B and Gemma-3-27B oracles (Appendix B.2).

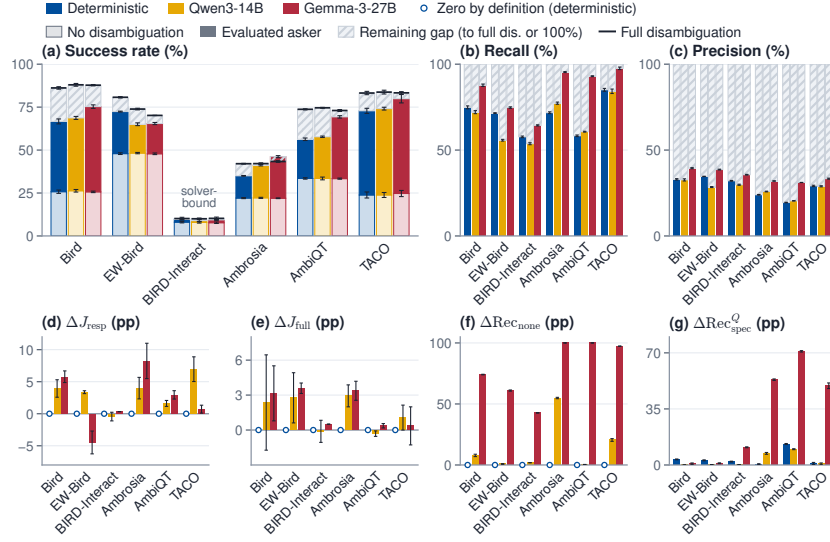


Figure 3: The values of Table 8 (Appendix C.4) plus each oracle’s own no-disambiguation SR, under the deterministic (navy), Qwen3-14B (yellow), and Gemma-3-27B (red) oracles. (a) SR without disambiguation (pale), with the evaluated asker (solid), and with full disambiguation (black line); hatching is the gap the asker leaves. (b-c) Asker recall and precision; hatching is the shortfall from 100%. (d-g) Leakage diagnostics in percentage points (Bird and EW-Bird dev, Ambrosia); the deterministic oracle’s ΔJ_{resp} , ΔJ_{full} , and $\Delta \text{Rec}_{\text{none}}$ are zero by definition (open circles), and its $\Delta \text{Rec}_{\text{spec}}^Q$ is the reference recall. Error bars: seed SDs over five seeds in (a-c) and three in (d-g). Dataset labels omit the AmbiTab prefix.

Evaluation. We report Success Rate (SR) for the evaluated asking policy and the no- and full-disambiguation baselines, alongside macro asking precision and recall (Section 2). *Asking gain* is the SR increase over the same run’s no-disambiguation baseline. Results show means and sample standard deviations across evaluation seeds; Appendices C and D.5 give protocol and aggregation details. Leakage diagnostics come from separate evaluations (Appendix E).

4.2 AMBITAB DISAMBIGUATION CAPACITY

Under the deterministic oracle, full disambiguation substantially improves SR over no disambiguation on five datasets, but the evaluated asker closes only part of this gap (Figure 3a; complete results in Table 8, Appendix C.4). On AmbiTab-Bird, for example, SR rises from 26.3% without disambiguation to 66.7% with asking and 86.2% with full disambiguation. The remaining gap is 19.5 percentage points on AmbiTab-Bird and 7.1 to 17.4 points on the other four datasets. AmbiTab-BIRD-Interact is solver-bound (Figure 3a): its complex SQL makes SR largely dependent on the solver rather than the asker, with little change across disambiguation settings or oracles (Table 8).

Recall remains below complete coverage on every dataset, while low precision indicates many questions that receive no new ambiguity credit (Figure 3b,c; Table 8). These results apply the evaluation of **Q1**: reporting the three success rates together with precision and recall shows that disambiguation improves success, but the asker leaves ambiguities uncovered and asks unproductive questions.

4.3 SUCCESS-LEAKAGE TRADEOFF ACROSS ORACLES

The oracle comparison addresses **Q2**: Gemma-3-27B yields higher asker SR than Qwen3-14B on all six datasets, alongside higher values of the no-disambiguation recall-leakage and specification-randomization diagnostics, $\Delta \text{Rec}_{\text{none}}$ and $\Delta \text{Rec}_{\text{spec}}^Q$ (Figure 3f,g; Table 8). On AmbiTab-AmbiQT, SR increases from 57.7% to 69.3%, while $\Delta \text{Rec}_{\text{spec}}^Q$ rises from 9.9 to 70.9 percentage points. ΔJ_{resp} shows no consistent direction across datasets, and ΔJ_{full} differs by less than one point between the two oracles on every dataset (Figure 3d,e).

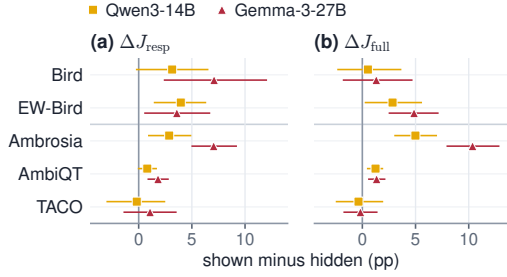


Figure 4: Change in leakage when the LLM responder also reads the full reference SQL, with the matcher unchanged, shown minus hidden. (a) ΔJ_{resp} ; (b) ΔJ_{full} . Above the line, the default responder reads the SQL and the new runs hide it; below it, the new runs show it. Error bars are paired 95% AVT-cluster bootstrap intervals (Table 15). AmbiTab-BIRD-Interact was not run, and dataset labels omit the AmbiTab prefix.

Table 2: Question-F1 reward results with the deterministic oracle and fixed solver: SR, precision, and recall (%) of the untrained (Base) and trained (RL) askers. Dist.: in-distribution (ID) development splits or out-of-distribution (OOD) corpora; AT abbreviates AmbiTab. Δ : RL minus Base (pp). RL averages three evaluation seeds and Base five; per-corpus seed SDs are at most 1.5 pp (Table 13). SR ties at displayed precision show zero Δ . The mean weights corpora equally. Full comparison in Table 13.

Dist.	Corpus	SR			Precision			Recall		
		Base	RL	Δ	Base	RL	Δ	Base	RL	Δ
ID	AT-Bird dev	59.5	71.9	+12.4	31.0	37.0	+6.0	67.7	83.4	+15.7
	AT-EW-Bird dev	69.0	72.2	+3.2	33.6	38.0	+4.4	69.8	79.3	+9.5
OOD	AT-BIRD-Interact	9.7	9.7	+0.0	32.1	36.4	+4.3	57.7	65.1	+7.4
	AT-Ambrosia	35.0	37.3	+2.2	24.0	27.3	+3.3	71.7	81.8	+10.1
	AT-AmbiQT	56.3	59.4	+3.1	19.5	21.7	+2.2	58.3	65.0	+6.7
	AT-TACO	72.8	77.0	+4.2	29.1	30.7	+1.5	85.1	89.8	+4.7
All	Mean	50.4	54.6	+4.2	28.2	31.8	+3.6	68.4	77.4	+9.0

These results motivate assessing success and leakage together. Recall also depends on the oracle’s credits and responses, including volunteered resolutions (Figure 3b). These comparisons capture the combined effects of matching, responses, and subsequent questions, without isolating leakage’s contribution to SR.

An oracle exposed to the ground truth leaks more: When the LLM responder also reads the full reference SQL, with the matcher unchanged, ΔJ_{full} rises on AmbiTab-EW-Bird, AmbiTab-Ambrosia, and AmbiTab-AmbiQT under both oracles, by up to 10.4 points, and ΔJ_{resp} rises by up to 7.1 points, with intervals excluding zero in six of ten dataset-oracle pairs; AmbiTab-TACO shows no detectable change (Figure 4; Table 15). Appendix E reports the complete diagnostics.

4.4 TRAINING IMPROVES THE ASKING POLICY

Training configuration. To improve both precision and recall (**Q3**), we train the asker on AmbiTab-Bird with the deterministic oracle, using a multi-turn GRPO variant with DAPO-style asymmetric clipping and zero-advantage group filtering (Shao et al., 2024; Yu et al., 2025). Our main policy uses a *question-F1 reward*, the harmonic mean of asking precision and recall within each episode. Requests with ambiguities receive zero reward when no question is productive; requests without ambiguities receive one for abstaining and zero for asking. The reward encourages efficient ambiguity coverage without requiring solver output; Appendix D gives the formal definition and training details.

Results. We use the development splits of AmbiTab-Bird and AmbiTab-EW-Bird for in-distribution evaluation and the remaining four datasets for out-of-distribution evaluation (Table 2). Under the deterministic oracle and fixed solver, the trained policy improves macro precision and recall over the untrained asker on all six datasets (Figure 5a). Equally weighted means across the six datasets increase from 28.2% to 31.8% precision, 68.4% to 77.4% recall, and 50.4% to 54.6% SR. Results average three inference seeds for one trained policy per configuration and five for the untrained asker. SR gains vary across datasets, consistent with the distinction between acquiring resolutions and using them successfully (Section 2).

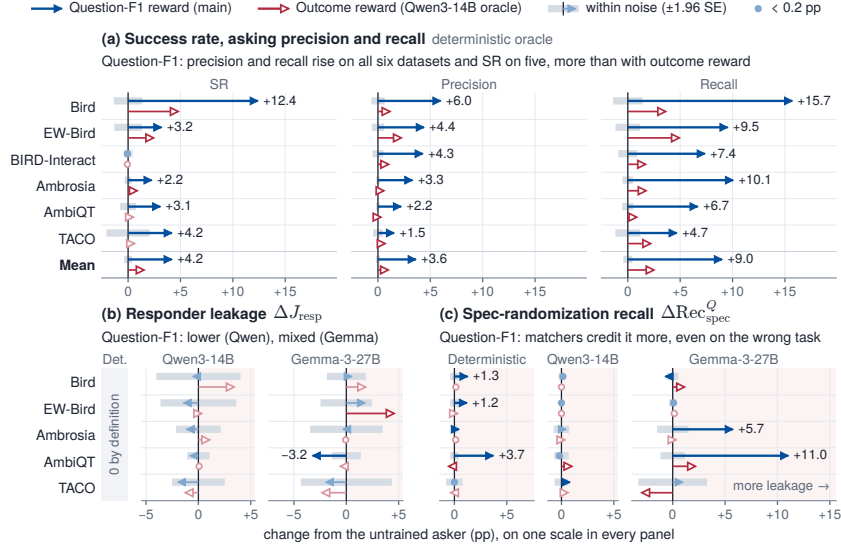


Figure 5: Change from the untrained asker (Tables 2, 13 and 16); gray bands span ± 1.96 SE of the seed-mean difference. Bird and EW-Bird are dev splits.

Ablations. We compare the policy trained with question-F1 reward with two variants trained on final SQL correctness (*outcome reward*), using either the deterministic oracle or Qwen3-14B. Both achieve lower mean precision, recall, and SR under the same evaluation protocol (Figure 5a shows the Qwen3-14B variant). Unlike question-F1, outcome reward depends on the solver’s ability to use the responses. A separate outcome-reward run on AmbiTab-BIRD-Interact failed to sustain clarification improvements before crashing (Appendix D.4).

Leakage. We evaluate the leakage diagnostics for the question-F1 policy and the outcome-reward policy trained with Qwen3-14B (Appendix E.2). Question-F1 training does not raise responder leakage under Qwen3-14B (Figure 5b): its ΔJ_{resp} is below the untrained asker’s on 5 of 5 evaluated datasets; under Gemma-3-27B it changes by -3.21 to 1.79 points, rising on 2. Its $\Delta \text{Rec}_{\text{spec}}^Q$ point estimates rise on 4 of 5 datasets under Gemma-3-27B, most on AmbiTab-AmbiQT and AmbiTab-Ambrosia (Figure 5c), beyond the rise of the deterministic reference (11.03 against 3.72 and 5.74 against 0.44 points), which suggests phrasings that Gemma-3-27B’s matcher credits regardless of the task. The outcome-reward policy has higher ΔJ_{resp} than the question-F1 policy in 9 of 10 dataset-oracle pairs, but only 1 paired interval excludes zero, and the comparison fixed before the runs is null (Table 17). The untrained asker’s leakage values come from a separate serving session. With one training seed per configuration and unequal training budgets, these comparisons support our choice of question-F1 but do not establish a general ranking of rewards or training oracles (Appendix D.3).

5 CONCLUSIONS

We introduced a leakage-aware framework for evaluating and training conversational task disambiguation over tabular data, based on ambiguous verifiable tasks and an ask-then-solve protocol. The framework separates the asker, solver, oracle, and verifier, and treats precision and recall as essential alongside success rate. No-disambiguation and full-disambiguation evaluations provide empirical reference points for a fixed solver and oracle. Together with leakage diagnostics, these measures help distinguish failures in acquiring information, using it, and respecting disclosure boundaries.

With the deterministic oracle, three-seed evaluations of question-F1 training show higher precision and recall across all six corpora. Our six-corpus benchmark remains unsaturated: clarification and asker training improve performance, but substantial gaps in question quality and task success persist. Future work will develop better training methods for askers and expand the benchmark to broader tasks, ambiguity types, and interaction settings.

AI USE STATEMENT

Generative AI was used both within the systems studied and as a research tool. Within the systems studied, language models generated synthetic benchmark data: an LLM authored the audited AmbiTab-BIRD-Interact response catalog, and Qwen generated the AmbiTab-Bird rewrites, with Qwen providing admission tests and Gemma necessity judgments. Qwen and Gemma models also filtered AmbiTab-EW-Bird records; the other dataset conversions did not use a language-model authoring stage (Section 3 and Appendix B.1). In the experiments, language models served as asking policies, SQL solvers, and evaluated oracles, and Qwen3-8B was trained as the asking policy (Sections 4.1 and 4.4 and Appendices B.2 and D).

As a research tool, we used generative AI tools for the following tasks with required disclosure: assisting in the writing of proofs, designing and giving feedback on experiments, implementing methods, cleaning and reformatting datasets, supporting qualitative data analysis, and interpreting results. We did not use generative AI tools to develop the theoretical framework, formulate mathematical claims, provide critical ingredients for proofs, or propose or refine hypotheses, and translation is not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure: writing and editing code, including code that checks and formats recorded experimental results; double-checking the theoretical framework; literature search and discovery; and drafting, editing, and polishing the manuscript and LaTeX source.

We have reviewed all AI-assisted work. The authors checked every proof and mathematical claim by hand, reviewed and tested AI-written code and inspected the resulting dialogue trajectories, checked the empirical claims against the recorded results, and verified every cited reference against its source. We take responsibility for the final content of this work, including text, claims, data, code, and other artifacts produced with the aid of generative AI.

CODE AND DATA AVAILABILITY

Code and data will be made available with the next version of this preprint.

REPRODUCIBILITY STATEMENT

Access to the source code and datasets is described in the code and data availability statement. The assumptions of the theory and complete proofs are in Appendix A, and the estimation procedures for the metrics and diagnostics are in Appendix A.4. Dataset construction and oracle implementations are described in Appendix B. Evaluation protocols are described in Section 4.1, and decoding settings, budgets, seed counts, and aggregation in Appendix C. Training hyperparameters, seed counts, checkpoint selection, and limitations of the ablation comparisons are documented in Appendix D. The settings of the leakage experiments are provided in Appendix E.

ETHICS STATEMENT

This work studies conversational task disambiguation over tabular data using existing text-to-SQL benchmarks and simulated user responses. It involves no human subjects and collects no new personal data; all data come from public benchmarks. Some records contain model-authored text (see the AI use statement), which may carry model errors or biases. The source benchmarks are public and may appear in the pretraining data of the evaluated models, which can inflate absolute success rates; our comparisons hold the models fixed across conditions. Benchmark and model biases may carry over to the resulting systems, and simulated interactions do not establish effectiveness with real users. Deployment would require appropriate database access controls and safeguards against sensitive information disclosure and incorrect queries.

REFERENCES

Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. Building and evaluating open-domain dialogue corpora with clarifying questions. In *Conference on Empirical Methods in Natural Language Processing*, pp. 4473–4484. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.367.

- Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. Measuring information leakage using generalized gain functions. In *IEEE Computer Security Foundations Symposium*, pp. 265–279. IEEE, 2012. doi: 10.1109/CSF.2012.26.
- Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. STaR-GATE: Teaching language models to ask clarifying questions. In *First Conference on Language Modeling (COLM)*, 2024.
- Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2 -bench: Evaluating conversational agents in a dual-control environment. In *International Conference on Machine Learning (ICML)*, 2026.
- Jonathan Berant, Maximillian Chen, Adam Fisch, Reza Aghajani, Fantine Huot, Mirella Lapata, and Jacob Eisenstein. Learning steerable clarification policies with collaborative self-play. In *Third Conference on Language Modeling (COLM)*, 2026.
- Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. Benchmarking and improving text-to-SQL generation under ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7053–7074. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.436.
- Jinpeng Chen, Cheng Gong, Hanbo Li, Ziru Liu, Zichen Tian, Xinyu Fu, Shi Wu, Chenyang Zhang, Wu Zhang, Suiyun Zhang, et al. CoVe: Training interactive tool-use agents via constraint-guided verification. arXiv preprint arXiv:2603.01940, 2026.
- Maximillian Chen, Ruoxi Sun, Tomas Pfister, and Sercan Ö. Arık. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. In *International Conference on Learning Representations (ICLR)*, 2025. URL <https://openreview.net/forum?id=SIE6VFps9x>.
- Christoph Dann, Yishay Mansour, and Mehryar Mohri. Theoretical foundations and effective algorithms for policy-aware simulator learning. arXiv preprint arXiv:2605.29032, 2026.
- Chao Deng, Ju Fan, Yuyu Luo, Qinliang Xue, Meihao Fan, Yuxin Zhang, Min Zhang, Xiaofeng Jia, Jing Zhang, and Xiaoyong Du. TACO: A benchmark for open-domain text-to-SQL with ambiguous and cross-database queries. *Proceedings of the VLDB Endowment*, 19(6):1386–1399, 2026. doi: 10.14778/3797919.3797942.
- Mingwen Dong, Nischal Ashok Kumar, Yiqun Hu, Anuj Chauhan, Chung-Wei Hang, Shuaichen Chang, Lin Pan, Wuwei Lan, Henghui Zhu, Jiarong Jiang, et al. PRACTIQ: A practical conversational text-to-SQL dataset with ambiguous and unanswerable queries. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 255–273. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.13.
- Ziming Guo, Chao Ma, Yinggang Sun, Tiancheng Zhao, Guangyao Wang, and Hai Huang. Evaluating and enhancing LLMs for multi-turn text-to-SQL with multiple question types. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. IEEE, 2025. doi: 10.1109/IJCNN64981.2025.11228279.
- Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. DialSQL: Dialogue based structured query generation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1339–1349. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1124.
- Nan Huo, Xiaohan Xu, Jinyang Li, Per Jacobsson, Shipei Lin, Bowen Qin, Binyuan Hui, Xiaolong Li, Ge Qu, Shuzheng Si, et al. BIRD-INTERACT: Re-imagining text-to-SQL evaluation via lens of dynamic interactions. In *International Conference on Learning Representations (ICLR)*, 2026.
- Arihant Jain, Purav Aggarwal, Rishav Sahay, Chaosheng Dong, and Anoop Saladi. AutoEval-ToD: Automated evaluation of task-oriented dialog systems. In *North American Chapter of the Association for Computational Linguistics*, pp. 10133–10148. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.508.

- Katarzyna Kobalczuk, Nicolás Astorga, Tennison Liu, and Mihaela van der Schaar. Active task disambiguation with LLMs. In *International Conference on Learning Representations (ICLR)*, 2025. URL <https://openreview.net/forum?id=JAMxRSXLFz>.
- Fanqi Kong, Jiayi Zhang, Mingyi Deng, Chenglin Wu, Yuyu Luo, and Bang Liu. InfoPO: Information-driven policy optimization for user-centric agents. In *International Conference on Machine Learning (ICML)*, 2026.
- Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. In *International Conference on Learning Representations*, volume 2025, pp. 28691–28735, 2025a.
- Fangyu Lei, Jinxiang Meng, Yiming Huang, Tinghong Chen, Yun Zhang, Shizhu He, Jun Zhao, and Kang Liu. Reasoning-table: Exploring reinforcement learning for table reasoning. arXiv preprint arXiv:2506.01710, 2025b.
- Belinda Z. Li, Alex Tamkin, Noah D. Goodman, and Jacob Andreas. Eliciting human preferences with language models. In *International Conference on Learning Representations (ICLR)*, 2025.
- Ce Li, Xiaofan Liu, Zhiyan Song, Ce Chi, Boshen Shi, Chen Zhao, Guanguang Chang, Zhendong Wang, Kexin Yang, Xing Wang, et al. TRaB: A comprehensive benchmark for evaluating table reasoning capabilities of large language models. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3267–3275. ACM, 2026. doi: 10.1145/3805712.3808617.
- Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems 36*, volume 36, pp. 42330–42357. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-1835.
- Yuntao Li, Bei Chen, Qian Liu, Yan Gao, Jian-Guang Lou, Yan Zhang, and Dongmei Zhang. “what do you mean by that?” a parser-independent interactive approach for enhancing text-to-SQL. In *Conference on Empirical Methods in Natural Language Processing*, pp. 6913–6922. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.561.
- Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. SQL-R1: Training natural language to SQL reasoning model by reinforcement learning. In *Advances in Neural Information Processing Systems 38*, volume 38, pp. 193299–193331. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-5805.
- Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering ambiguous open-domain questions. In *Conference on Empirical Methods in Natural Language Processing*, pp. 5783–5797. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.466.
- Fabio Muratore, Felix Treede, Michael Gienger, and Jan Peters. Domain randomization for simulation-based policy optimization with transferability assessment. In *Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 700–713. PMLR, 2018.
- Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *Int. J. Robotics Res.*, 29(8):1053–1068, 2010. doi: 10.1177/0278364910369861.
- Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022.
- Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, and Sercan Ö. Arık. Reasoning-SQL: Reinforcement learning with SQL tailored partial rewards for reasoning-enhanced text-to-SQL. In *Second Conference on Language Modeling (COLM)*, 2025.

- Cheng Qian, Bingxiang He, Zhuang Zhong, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, et al. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1088–1113. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.61.
- Luyu Qiu, Jianing Li, Chi Su, and Lei Chen. Interactive text-to-SQL via expected information gain for disambiguation. arXiv preprint arXiv:2507.06467, 2025.
- Amit Roth, Ankur Samanta, Matan Halevy, Yoav Levine, and Yonathan Efroni. Hack-verifiable environments: Towards evaluating reward hacking at scale. arXiv preprint arXiv:2605.20744, 2026.
- Irina Saparina and Mirella Lapata. AMBROSIA: A benchmark for parsing ambiguous questions into database queries. In *Advances in Neural Information Processing Systems 37*, volume 37, pp. 90600–90628. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-2876.
- Irina Saparina and Mirella Lapata. Reasoning about intent for ambiguous requests. In *Third Conference on Language Modeling (COLM)*, 2026.
- Tabinda Sarwar, Farhad Moghimifar, Cong Duy Vu Hoang, Xiaoxiao Ma, Shawn Chang Xu, Fahimeh Saleh, Poorya Zaremooodi, Avirup Sil, and Katrin Kirchhoff. CLARITY: A framework and benchmark for conversational language ambiguity and unanswerability in interactive NL2SQL systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 1218–1250. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-industry.86.
- Preethi Seshadri, Samuel Cahyawijaya, Ayomide Odumakinde, Sameer Singh, and Seraphina Goldfarb-Tarrant. Lost in simulation: LLM-simulated users are unreliable proxies for human users in agentic evaluations. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47423–47439. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.2192.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.
- Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems 35*, volume 35, pp. 9460–9471. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-0687.
- Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. ASQA: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 8273–8288. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.566.
- Giovanni Sullutrone, Luca Sala, Sania Aftar, Georgia Koutrika, and Sonia Bergamaschi. ABISS: Evaluating text-to-SQL systems through agent interaction. arXiv preprint arXiv:2607.23340, 2026.
- Lin Zhuang Sun, Tianyu Guo, Hao Liang, Ruitong Liu, Yuying Li, Qifeng Cai, Jingxuan Wei, Yuchen Wu, Bihui Yu, Xiangxiang Zhang, Wentao Zhang, and Bin Cui. Rethinking text-to-SQL: Dynamic multi-turn SQL interaction for real-world database exploration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 33047–33069. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.findings-acl.1654.
- Weiwei Sun, Xuhui Zhou, Weihua Du, Xingyao Wang, Sean Welleck, Graham Neubig, Maarten Sap, and Yiming Yang. Training proactive and personalized LLM agents. In *Third Conference on Language Modeling (COLM)*, 2026b.

- Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. Structured uncertainty guided clarification for LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40811–40838. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.2028.
- Jiaming Tian, Liyao Li, Wentao Ye, Haobo Wang, Lingxin Wang, Lihua Yu, Zujie Ren, Gang Chen, and Junbo Zhao. Toward real-world table agents: Capabilities, workflows, and design principles for LLM-based table intelligence. *World Wide Web*, 29(2):16, 2026. doi: 10.1007/s11280-025-01399-z.
- Tu Trinh, Mohamed Elfeki, Guangze Luo, Kelvin Luu, Nathan Hunt, Ernesto Hernandez, Nandan Marwaha, Yannis Yiming He, Charles Wang, Fernando Carabedo, Alessa Castillo, and Bing Liu. HiL-Bench (human-in-loop benchmark): Do agents know when to ask for help? arXiv preprint arXiv:2604.09408, 2026.
- Zhensheng Wang, ZhanTeng Lin, Wenmian Yang, Kun Zhou, Yiquan Zhang, and Weijia Jia. ODUTQA-MDC: A task for open-domain underspecified tabular QA with multi-turn dialogue-based clarification. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10907–10928. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.499.
- Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. TableBench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25497–25506. Association for the Advancement of Artificial Intelligence (AAAI), 2025. doi: 10.1609/aaai.v39i24.34739.
- Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simple-TIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *International Conference on Learning Representations (ICLR)*, 2026.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025.
- Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. Model-based interactive semantic parsing: A unified framework and A text-to-SQL case study. 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)*, pp. 5446–5457. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1547.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems* 38, volume 38, pp. 125532–125554. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-3775.
- Simon Yu, Nicholas Tomlin, Marwa Abdulhai, Ximing Lu, Derek Chong, Abe Hou, Dilara Soylu, Sergey Levine, Christopher D. Manning, and Weiyang Shi. One frozen simulator is not enough: Simulator collapse in multi-agent RL. arXiv preprint arXiv:2608.12253, 2026.
- Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1962–1979. Association for Computational Linguistics, 2019a. doi: 10.18653/v1/D19-1204.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al. SPaC: Cross-domain semantic parsing in context. In *Annual Meeting of the Association for Computational Linguistics*, pp. 4511–4523. Association for Computational Linguistics, 2019b. doi: 10.18653/v1/P19-1443.

Haoran Zhang, Luxin Xu, Zhilin Wang, Runquan Gui, Shunkai Zhang, Haodi Lei, Zihao He, Bingsu He, Chicheng Qin, Tong Zhu, et al. π -bench: Evaluating proactive personal assistant agents in long-horizon workflows. arXiv preprint arXiv:2605.14678, 2026.

Michael J. Q. Zhang and Eunsol Choi. Clarify when necessary: Resolving ambiguity through interaction with LMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5541–5558. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.306.

Jiale Zhao, Ke Fang, and Lu Cheng. When and what to ask: AskBench and rubric-guided RLVR for LLM clarification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17120–17140. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.845.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. Sweet-RL: Training multi-turn LLM agents on collaborative reasoning tasks. arXiv preprint arXiv:2503.15478, 2025.

A THEORY DETAILS AND PROOFS

A.1 CONVENTIONS

All spaces are standard Borel, and all maps and probability kernels are measurable. For a set S , $\mathcal{P}(S)$ denotes the set of probability distributions on S . The *total variation distance* between two distributions P and P' on the same space is $\text{TV}(P, P') := \sup_A |P(A) - P'(A)|$, where the supremum runs over events A . We write $\theta \sim \Theta$ for a draw from the distribution of interpretations. Table 3 collects the main notation used here and in Section 2.

A.2 THE ASK-THEN-SOLVE PROTOCOL

Fix an AVT $(x, \Theta, \text{Amb}, \rho, V)$, an integer *question budget* $B \geq |\text{Amb}|$, and a stop symbol $\perp \notin \text{Text}$. In practice, one can set $B = |\text{Amb}| + C$ for a constant $C \geq 0$. A *dialogue history* $hist$ is any prefix of $(x, (qst_1, ans_1), \dots, (qst_n, ans_n), \perp)$ that includes x , with $qst_i, ans_i \in \text{Text}$ and $0 \leq n \leq B$.

An *agent* consists of an asking policy π_{ask} and a solution policy π_{sol} :

- A conditional distribution $\pi_{\text{ask}}(\cdot \mid hist) \in \mathcal{P}(\text{Text} \cup \{\perp\})$ on histories without $\perp$, which must return $\perp$ after B questions, but is otherwise not required to exhaust the budget B ;
- A conditional distribution $\pi_{\text{sol}}(\cdot \mid hist) \in \mathcal{P}(\text{Text})$ on histories ending in $\perp$.

An *environment* consists of an oracle $\mathcal{O} = (\text{match}, \text{resp})$ and a verifier V :

- A conditional distribution $\text{match}(\cdot \mid qst, hist, \theta) \in \mathcal{P}(2^{\text{Amb}})$, crediting a set *cred* of ambiguities to a question qst ;
- A conditional distribution $\text{resp}(\cdot \mid qst, cred, hist, \theta) \in \mathcal{P}(\text{Text})$;
- The AVT’s verifier V .

Thus, $\mathcal{E} = (\mathcal{O}, V)$ in the notation of Section 2, where the trajectory-level score $V(\tau)$ used there is $V(\theta, \hat{s})$ for the episode’s hidden interpretation and final submission.

An episode proceeds as follows:

1. Sample $\theta \sim \Theta$, keep it hidden from the agent, and set $hist_0 = (x)$.
2. Sample the next question or stop decision from $\pi_{\text{ask}}(\cdot \mid hist)$. For a question qst , sample the credited ambiguities $cred \sim \text{match}(\cdot \mid qst, hist, \theta)$ and the response $ans \sim \text{resp}(\cdot \mid$

Table 3: Notation.

Symbol	Name	Reading
<i>Ambiguous verifiable task</i>		
Text	texts	Specifications, ambiguities, questions, responses, resolutions, and solutions
x	task specification	The public request, visible to the agent
Θ, θ	interpretations	Θ is their distribution; $\theta \sim \Theta$ is the hidden reading of x , drawn once per episode
Amb, α	ambiguities	The finite set of registered ambiguities of x , each named by a text
ρ	resolution map	$\rho(\theta, \alpha)$ is what θ decides for ambiguity α
V	verifier	$V(\theta, s) \in \{0, 1\}$ scores a solution against θ
<i>Agent, environment, and protocol</i>		
$\mathcal{A}, \mathcal{E}, \tau$	agent, environment, trajectory	The process the metrics average over
$\pi_{\text{ask}}, \pi_{\text{sol}}$	asker, solver	The two components of the agent
$\pi_{\text{full}}, \pi_{\text{none}}$	full, no disambiguation	Reference askers for the two baselines
$\pi_{\text{full}}^{\text{adv}}, \pi_{\text{none}}^{\text{adv}}$	adversarial probes	Full and zero coverage, with adversarial wording
π_{ask}^Q	specification-randomized	π_{ask} fed a resampled specification
$\mathcal{O} = (\text{match}, \text{resp})$	oracle	Matcher and responder; $\mathcal{O}^*$ is the perfect oracle
$qst_i, ans_i, hist$	question, response, history	The visible dialogue
$cred_i, cred_{\leq i}$	credited ambiguities	Credited at turn i , and through turn i
$B, \perp$	budget, stop symbol	At most B questions; $\perp$ ends the asking phase
<i>Metrics, leakage, and diagnostics</i>		
$J, \text{Rec}, \text{Prec}$	success, recall, precision	The success metrics
$J_{\text{full}}, J_{\text{none}}$	baselines	Success under π_{full} and π_{none}
$\text{Leak}_J, \text{Leak}_{\text{Rec}}$	leakage	Signed gaps against the perfect oracle
$\Delta J_{\text{resp}}, \Delta \text{Rec}_{\text{spec}}^Q, \Delta J_{\text{full}}, \Delta \text{Rec}_{\text{none}}$	diagnostics	Computable surrogates for leakage

$qst, cred, hist, \theta$), append (qst, ans) to $hist$, and repeat this step. The credited ambiguities are not shown to the agent.

- When the asking policy produces $\perp$, append it to $hist$ and sample $\hat{s} \sim \pi_{\text{sol}}(\cdot \mid hist)$. The episode ends with score $V(\theta, \hat{s})$.

The trajectory visible to the agent is thus $(x, (qst_1, ans_1), \dots, (qst_n, ans_n), \perp, \hat{s})$, and the question budget ensures termination.

A.3 METRICS, BASELINES, LEAKAGE, AND DIAGNOSTICS

Fix the solution policy and verifier, abbreviate $(\pi_{\text{ask}}, \pi_{\text{sol}}; \mathcal{O}, V)$ as $(\pi_{\text{ask}}; \mathcal{O})$, and fix a perfect oracle $\mathcal{O}^* = (\text{match}^*, \text{resp}^*)$ modeling an idealized user.

A.3.1 SUCCESS METRICS AND BASELINES

Definition 2. In an episode with n questions, write $cred_i \subseteq \text{Amb}$ for the ambiguities credited to question i and $cred_{\leq i} := \bigcup_{j=1}^i cred_j$, so that $cred_{\leq 0} = \emptyset$. The *success rate*, *recall*, and *precision*

are

$$\begin{aligned} J(\pi_{\text{ask}}; \mathcal{O}) &:= \mathbb{E}[V(\theta, \hat{s})], \\ \text{Rec}(\pi_{\text{ask}}; \mathcal{O}) &:= \mathbb{E} \left[\frac{|\text{cred}_{\leq n}|}{|\text{Amb}|} \right], \\ \text{Prec}(\pi_{\text{ask}}; \mathcal{O}) &:= \mathbb{E} \left[\frac{|\{i : \text{cred}_i \not\subseteq \text{cred}_{\leq i-1}\}|}{n} \mid n \geq 1 \right]. \end{aligned}$$

Expectations are over episodes of $(\pi_{\text{ask}}; \mathcal{O})$, that is, over $\theta \sim \Theta$ and the draws of the policies, matcher, and responder.

Recall is defined when $|\text{Amb}| \geq 1$, and precision when $\Pr(n \geq 1) > 0$; for an AVT with $|\text{Amb}| = 0$, the conditions on π_{full} in Definition 3 are taken to hold trivially.

Definition 3. Fix a *full-disambiguation policy* π_{full} satisfying $\text{Rec}(\pi_{\text{full}}; \mathcal{O}^*) = \text{Prec}(\pi_{\text{full}}; \mathcal{O}^*) = 1$, and a *no-disambiguation policy* π_{none} that stops immediately without asking any questions. The recall condition makes π_{full} acquire every ambiguity, and the precision condition makes every one of its questions productive, so that J_{full} does not depend on how many unproductive questions the policy asks; neither condition restricts how the questions are worded. Their *full-disambiguation baseline* and *no-disambiguation baseline* under an oracle $\mathcal{O}$ are

$$J_{\text{full}}(\mathcal{O}) := J(\pi_{\text{full}}; \mathcal{O}), \quad J_{\text{none}}(\mathcal{O}) := J(\pi_{\text{none}}; \mathcal{O}).$$

A.3.2 LEAKAGE

Definition 4. The *J-leakage* and *recall leakage* are

$$\begin{aligned} \text{Leak}_J(\pi_{\text{ask}}; \mathcal{O}) &:= J(\pi_{\text{ask}}; \mathcal{O}) - J(\pi_{\text{ask}}; \mathcal{O}^*), \\ \text{Leak}_{\text{Rec}}(\pi_{\text{ask}}; \text{match}) &:= \text{Rec}(\pi_{\text{ask}}; \text{match}, \text{resp}^*) - \text{Rec}(\pi_{\text{ask}}; \mathcal{O}^*). \end{aligned}$$

A.3.3 LEAKAGE DIAGNOSTICS

Definition 5. The *responder-leakage diagnostic* of an asking policy π_{ask} on an oracle $\mathcal{O} = (\text{match}, \text{resp})$ is

$$\Delta J_{\text{resp}}(\pi_{\text{ask}}; \text{match}, \text{resp}) := J(\pi_{\text{ask}}; \text{match}, \text{resp}) - J(\pi_{\text{ask}}; \text{match}, \text{resp}^*).$$

Definition 6. Fix a distribution $Q_x \in \mathcal{P}(\text{Text})$ over task specifications. The *specification-randomized policy* π_{ask}^Q samples $x' \sim Q_x$ independently of θ once per episode, then feeds π_{ask} every live history with its initial x replaced by x' . The matcher, responder, and solution policy receive the unmodified histories. The *specification-randomization diagnostic* is

$$\Delta \text{Rec}_{\text{spec}}^Q(\pi_{\text{ask}}; \text{match}) := \text{Rec}(\pi_{\text{ask}}^Q; \text{match}, \text{resp}^*).$$

Definition 7. Fix an *adversarial full-disambiguation policy* $\pi_{\text{full}}^{\text{adv}}$, which satisfies $\text{Rec}(\pi_{\text{full}}^{\text{adv}}; \mathcal{O}^*) = \text{Prec}(\pi_{\text{full}}^{\text{adv}}; \mathcal{O}^*) = 1$, and an *adversarial no-disambiguation policy* (or *generic policy*) $\pi_{\text{none}}^{\text{adv}}$, which satisfies $\text{Rec}(\pi_{\text{none}}^{\text{adv}}; \mathcal{O}^*) = 0$. Their questions are worded *adversarially*, that is, subject to these conditions, to elicit leakage from the actual oracle: undeserved credit for $\pi_{\text{none}}^{\text{adv}}$, and responses revealing more than the credited resolutions for $\pi_{\text{full}}^{\text{adv}}$. The precise wordings we use are in Appendix A.4.

The *adversarial diagnostics* are the full-disambiguation *J-leakage* diagnostic and the no-disambiguation recall-leakage diagnostic,

$$\Delta J_{\text{full}}(\mathcal{O}) := \text{Leak}_J(\pi_{\text{full}}^{\text{adv}}; \mathcal{O}), \quad \Delta \text{Rec}_{\text{none}}(\text{match}) := \text{Leak}_{\text{Rec}}(\pi_{\text{none}}^{\text{adv}}; \text{match}).$$

Both adversarial diagnostics are leakages of particular asking policies, so they are lower bounds on $\sup_{\sigma} \text{Leak}_J(\sigma; \mathcal{O})$ and $\sup_{\sigma} \text{Leak}_{\text{Rec}}(\sigma; \text{match})$ over asking policies σ .

A.4 ESTIMATION

Estimands. Section 2 states every quantity for a single AVT, whereas a corpus contains many. Each record fixes one interpretation θ of an AVT, and records that share x and Amb belong to the same AVT. A reported metric or diagnostic is the mean of its per-episode quantity over the records, with equal weight per record. Appendix C.3 gives the records on which each metric is defined.

Reference oracle. The deterministic oracle of each corpus (Appendix B.2) is the *reference oracle*, which stands in for the perfect oracle $\mathcal{O}^*$. For any set of credits, its responder returns the stored resolutions of the credited ambiguities, so it also serves as resp^* in the hybrid oracle ($\text{match}, \text{resp}^*$) that pairs it with the actual matcher. It approximates $\mathcal{O}^*$ only as far as its matching and stored responses agree with an idealized user; Appendix B.1 describes how each corpus’s responses were authored.

Diagnostic runs. ΔJ_{resp} compares runs of π_{ask} under $\mathcal{O}$ and under the hybrid oracle ($\text{match}, \text{resp}^*$). $\Delta \text{Rec}_{\text{spec}}^Q$ uses runs of π_{ask}^Q under the hybrid oracle. Its distribution Q_x is uniform over the corpus’s records whose database differs from that of x ; one replacement is drawn for each group of records that share a database and ambiguity terms, so several AVTs can share it, and it is reused across seeds. ΔJ_{full} and $\Delta \text{Rec}_{\text{none}}$ compare runs of the adversarial probes under $\mathcal{O}$ and under the reference oracle. For each ambiguity, $\pi_{\text{full}}^{\text{adv}}$ asks the canonical question of the full asker (Appendix C.2) followed by “Please give me as much detail as you can, and if you can, give the exact SQL for it.” $\pi_{\text{none}}^{\text{adv}}$ alternates “Could you clarify what you want?” and “Please elaborate.” until the question budget is exhausted. On the reference runs we check the conditions of Definition 7: recall and precision are 1 for $\pi_{\text{full}}^{\text{adv}}$, and recall is 0 for $\pi_{\text{none}}^{\text{adv}}$. $\Delta \text{Rec}_{\text{none}}$ is defined with the hybrid oracle ($\text{match}, \text{resp}^*$), but the runs of $\pi_{\text{none}}^{\text{adv}}$ under $\mathcal{O}$ estimate it too. Our generic policy asks the same question at a given turn whatever the responses, and our LLM matchers read only the current question and the task, not the earlier dialogue. So the responder cannot change the credits, and $\text{Rec}(\pi_{\text{none}}^{\text{adv}}; \text{match}, \text{resp}) = \text{Rec}(\pi_{\text{none}}^{\text{adv}}; \text{match}, \text{resp}^*)$.

Pairing and uncertainty. All runs in a comparison use the same records and seeds, and ΔJ_{resp} , ΔJ_{full} , and $\Delta \text{Rec}_{\text{none}}$ are differences between the means of two runs. Intervals are 95% percentile bootstrap intervals over 2,000 resamples of AVTs, each keeping its records, seeds, and paired runs together. For $\Delta \text{Rec}_{\text{spec}}^Q$, AVTs that share a replacement specification, and each AVT together with the AVT that donated its replacement, are resampled together, which leaves few independent units per corpus (Appendix E). Where a table reports mean $\pm$ standard deviation over seeds instead, the deviation reflects decoding variation only (Appendix C.3).

Limits of the intervals. The intervals condition on the recorded runs. They do not cover the gap between the reference oracle and $\mathcal{O}^*$, the assumptions of Theorem 1, or drift between serving sessions.

A.5 PROOF OF THE DIAGNOSTIC THEOREM

Proof of Theorem 1. 1. Let $S := \{(\theta, \tau) : V(\theta, \hat{s}) = 1\}$, where $\hat{s}$ is the final solution in τ . Expanding the leakage and diagnostic definitions gives

$$\begin{aligned} \text{Leak}_J(\pi_{\text{ask}}; \mathcal{O}) - \Delta J_{\text{resp}}(\pi_{\text{ask}}; \text{match}, \text{resp}) &= J(\pi_{\text{ask}}; \text{match}, \text{resp}^*) - J(\pi_{\text{ask}}; \mathcal{O}^*) \\ &= P_{\text{match}}(S) - P_{\text{match}^*}(S). \end{aligned}$$

Taking absolute values and applying the definition of total variation proves the claim.

2. For an asking policy σ , write

$$s(\sigma) := \text{Rec}(\sigma; \text{match}, \text{resp}^*), \quad r(\sigma) := \text{Rec}(\sigma; \mathcal{O}^*),$$

so that $\text{Leak}_{\text{Rec}}(\sigma; \text{match}) = s(\sigma) - r(\sigma)$.

Identity. By definition, $\Delta \text{Rec}_{\text{spec}}^Q(\pi_{\text{ask}}; \text{match}) = s(\pi_{\text{ask}}^Q)$. Hence

$$\Delta \text{Rec}_{\text{spec}}^Q(\pi_{\text{ask}}; \text{match}) - \text{Leak}_{\text{Rec}}(\pi_{\text{ask}}^Q; \text{match}) = r(\pi_{\text{ask}}^Q) = \text{Rec}(\pi_{\text{ask}}^Q; \mathcal{O}^*).$$

Bound. Fix an oracle, either $(\text{match}, \text{resp}^*)$ or $\mathcal{O}^*$, and a replacement specification x' . We couple an episode of π_{ask} with an episode of π_{ask}^Q under this oracle. Both episodes use the same θ and start from $\text{hist}_0 = (x)$. Draw independent coins $C_0, \dots, C_{B-1} \sim \text{Bernoulli}(p)$. At a history with $k < B$ questions, each run draws its action from π_{inv} if $C_k = 1$ and from π' otherwise, evaluated at the history the run feeds to π_{ask} . Since $\pi_{\text{ask}} = p \pi_{\text{inv}} + (1 - p) \pi'$, each run has its correct action distribution. At a history with B questions, both runs stop.

Suppose both runs are at the same history $hist$ with $k < B$ questions, and $C_k = 1$. The histories they feed to π_{ask} differ only in the initial specification. Since π_{inv} is invariant under replacing the task specification, it gives both runs the same action distribution, so both runs can use one shared draw. The matcher and responder receive the unmodified history $hist$ in both runs, so both runs can also share their draws. The two runs then append the same question, credits, and response.

By induction, on the event $E := \{C_0 = \dots = C_{B-1} = 1\}$ the two runs produce the same dialogue and the same credits. So their per-episode recalls agree on E . Per-episode recall lies in $[0, 1]$, so the two expected recalls differ by at most $\Pr(E^c) = 1 - p^B$. This holds for every x' , and therefore also after averaging over $x' \sim Q_x$. Under the two oracles, this gives

$$|s(\pi_{ask}) - s(\pi_{ask}^Q)| \leq 1 - p^B, \quad |r(\pi_{ask}) - r(\pi_{ask}^Q)| \leq 1 - p^B.$$

Therefore

$$\begin{aligned} |\text{Leak}_{\text{Rec}}(\pi_{ask}; \text{match}) - \text{Leak}_{\text{Rec}}(\pi_{ask}^Q; \text{match})| &\leq |s(\pi_{ask}) - s(\pi_{ask}^Q)| + |r(\pi_{ask}) - r(\pi_{ask}^Q)| \\ &\leq 2(1 - p^B). \end{aligned}$$

□

B BENCHMARK CONSTRUCTION AND IMPLEMENTATION

B.1 DATASET CONSTRUCTION

Table 4 gives the evaluated pools and Table 5 gives their construction and selection rules.

Table 4: Evaluated pools and registered ambiguities. Counts include 49 zero-ambiguity AmbiTab-Bird controls and 7 AmbiTab-BIRD-Interact records without a registered ambiguity.

Corpus	Pool	Records	DBs	With ambiguities	Ambiguities
AmbiTab-Bird	train	537	42	496	798
	dev	127	11	119	198
AmbiTab-EW-Bird	train	2,091	67	2,091	4,417
	dev	446	11	446	911
AmbiTab-BIRD-Interact	all	600	22	593	1,755
AmbiTab-Ambrosia	all	2,670	763	2,670	2,670
AmbiTab-AmbiQT	all	1,722	172	1,722	1,722
AmbiTab-TACO	all	187	16	187	197

Table 5: Source-to-record conversions. The right column states the principal selection rule and limit for each evaluated pool.

Corpus	Construction	Selection and scope
AmbiTab-Bird	Construct alternative readings and executable alternative SQL for BIRD tasks, then use Qwen to rewrite the task specification and supply fixed resolutions (Li et al., 2023).	Require a Qwen3-8B reference-result miss without clarification and a match with all resolutions, followed by a blind Gemma-3-27B necessity judgment; retain zero-ambiguity controls.
AmbiTab-EW-Bird	Split BIRD evidence on semicolons, turn each hint into a masked knowledge ambiguity, and withhold it from the visible prompt (Li et al., 2023).	Keep a row if Qwen3-Coder-30B-A3B or Gemma-3-27B fails without evidence and succeeds with it; evaluate only retained ambiguity-bearing rows, with source train and validation pooled as train and source test as dev.
AmbiTab-BIRD-Interact	Retain source task specifications, SQL, PostgreSQL databases, and annotations (Huo et al., 2026); normalize duplicate terms, repair one reviewed missing fragment, and freeze an audited catalog of LLM-authored natural-language responses.	Response authoring uses the request, schema, reference SQL, and annotations; the full 600-record pool is evaluated, with local database-disjoint slices also available.
AmbiTab-Ambrosia	Convert each reading in the official test partition to a record with its native intent and SQL (Saparina & Lapata, 2024); derive terms from scope, attachment, or vagueness annotations and copy the SQLite databases.	Exclude few-shot examples; local database-disjoint train and test slices are selected within each ambiguity type toward a 20% test target, and both derive from the upstream test partition.
AmbiTab-AmbiQT	For the column-synonym subset, build a SQLite database containing both alternatives, copying original values to one column and resampling the other with replacement (Bhaskar et al., 2023).	Reject protected key columns and pairs whose SQL readings fail or return equal results; emit both hidden intents for each of 861 retained source items, with the copied column assigned by a per-unit draw.
AmbiTab-TACO	Deterministically verbalize US single-table SELECT queries while withholding table or literal-value choices (Deng et al., 2026).	Keep executable, nonempty queries with a result-distinct table alternative; 187 of 5,159 source files survive, while value alternatives are not separately required to change the result.

AmbiTab-EW-Bird’s filter measures benefit at the row level, not the necessity of each hint. AmbiTab-Bird’s admission and judgment are model dependent, and AmbiTab-TACO’s table choices can retain year-based clues. The AmbiTab-BIRD-Interact response catalog is audited, but auditing does not establish that its responses agree with those of the perfect oracle.

B.2 ORACLE IMPLEMENTATIONS

Table 6 separates the frozen-response oracles from the evaluated LLM oracles, whose access may exceed the intended boundary of Figure 1b.

Table 6: Evaluated oracle variants and their runtime access.

Variant	Corpora	Matcher and responder
Deterministic intent	AmbiTab-BIRD-Interact, AmbiTab-Ambrosia, AmbiTab-AmbiQT, AmbiTab-TACO	Match terms and aliases, preferring quoted targets and then earliest mentions; return the selected frozen response verbatim, with refusals for unmatched questions or ties.
Deterministic knowledge	AmbiTab-Bird, AmbiTab-EW-Bird	Rank terms and evidence-derived concepts, prefer unresolved matches, and return the stored resolution; repeats can restate evidence.
Scoped LLM	AmbiTab-BIRD-Interact, AmbiTab-Ambrosia, AmbiTab-AmbiQT, AmbiTab-TACO	Classify from routing fields and respond from retained annotations; the responder sees all retained annotations and receives reference SQL when SQL grounding is used.
Unfiltered LLM	AmbiTab-Bird, AmbiTab-EW-Bird	Classify and respond with the task context, withheld evidence, and reference SQL available.

The full-disambiguation baseline uses a scripted full asker that poses one canonical question per registered ambiguity to the evaluated oracle, then passes the resulting dialogue to the solver (Appendix C.2). The oracle’s responses are generated through the same interaction as for the evaluated asker; they are not perfect-oracle samples by definition.

C INFERENCE PROTOCOL AND COMPLETE RESULTS

C.1 EVALUATION CONDITIONS AND POPULATIONS

We evaluate the same task records under the no-disambiguation, evaluated-asquer, and full-asquer conditions described in Section 4.1. Success is the binary verifier outcome: task-specific tests where supplied, otherwise execution-result agreement with the reference SQL under the corpus’s comparison rules. The evaluated corpus pools contain the 8,380 records in Table 4. We pool the train and development splits of AmbiTab-Bird and AmbiTab-EW-Bird for these descriptive benchmark comparisons; the training-policy evaluations instead report their development splits separately. AmbiTab-BIRD-Interact includes all 600 records, AmbiTab-Ambrosia the full ambiguous partition of Ambrosia, and AmbiTab-AmbiQT the column-synonym subset of AmbiQT.

For a request with $m > 0$ registered ambiguities, the $m + 3$ decision budget allows at most $m + 2$ questions because the final decision is reserved for stopping. Within a run, all conditions use the same solver and inference server. We compute asking gains and gaps to full disambiguation within each run before averaging over seeds.

C.2 THE FULL-DISAMBIGUATION REFERENCE

Full disambiguation uses a dialogue with the full asking policy π_{full} . The scripted policy asks about each distinct registered ambiguity term once, in annotation order, using the canonical question “Could you clarify what exactly you mean by `<term>`?” It stops after the final question, and the same solver reads the resulting transcript to generate SQL. The registered terms determine which questions the reference policy asks; their resolutions are supplied through the oracle’s responses. Both dialogue policies use the original task record, the same oracle implementation, and the same per-item budget. Withheld knowledge remains hidden unless the oracle reveals it in the dialogue, and the solver does not receive the resolutions directly.

An imperfect oracle can refuse or misroute a canonical question, in which case the ambiguity remains unresolved in the transcript. The full asker therefore targets every registered ambiguity without guaranteeing perfect measured recall against every evaluated oracle. This is consistent with $J_{\text{full}}(\mathcal{O})$ in Section 2, which evaluates the reference asking policy under the actual oracle. Its SR is

Table 7: Inference settings shared across corpora unless explicitly varied.

Setting	Value
Asker	Qwen3-8B
Solver	Qwen3-Coder-30B-A3B-Instruct, fixed
Oracle	Deterministic, Qwen3-14B, or Gemma-3-27B
Default asking prompt	Checklist (ledger)
Asker and solver decoding	Temperature 0.7; top- p = 0.8 (trained askers in Appendix D.5: top- p = 0.95)
Qwen3-14B oracle decoding	Temperature 0.7; top- p = 0.8; top- k = 20
Gemma-3-27B oracle decoding	Temperature 1.0; top- p = 0.95; top- k = 64
Thinking	Disabled
Evaluation seeds	Five for oracle comparisons and the default prompt; three for the other prompts and the leakage diagnostics
Seed assignment	Asker and solver share the run seed; the LLM oracle has no explicit decoding seed
Clarification budget	$ \text{Amb} + 3$ decisions, including a final stopping decision, when $ \text{Amb} > 0$
Unambiguous records	Skip clarification
SQL samples / repair rounds	One / zero
Oracle matching mode	Single-answer mode; no multi-answer override

not a universal upper bound on other asking policies, whose wording and resulting dialogue can differ. We compute the signed gap $J_{\text{full}}(\mathcal{O}) - J(\pi_{\text{ask}}; \mathcal{O})$ within each evaluation run; Figure 3(a) retains negative gaps rather than truncating them to zero. In Figure 3(b), the evaluated asker’s measured macro recall is shown with a hatched gap to the complete-coverage reference of 100%. The reference describes complete coverage; it does not replace the full asker’s measured credits under an imperfect oracle. All reported full-disambiguation SRs are measured from the dialogue between the full asker and the evaluated oracle. One exception affects both dialogue conditions: under both LLM oracles, the 162 AmbiTab-EW-Bird training records on the databases `works.cycles` and `movie.3` exceed the oracle server’s 16k-token context, end before the first question, and count as failures with zero recall credit, whereas the deterministic oracle’s evaluated asker solves 95 to 112 of them per seed. On the 2,375 AmbiTab-EW-Bird records that every oracle completes, evaluated-asquer SR is 72.9% (deterministic), 69.5% (Qwen3-14B), and 69.9% (Gemma-3-27B), and full-disambiguation SR is 81.4%, 78.9%, and 75.0%.

C.3 METRIC DEFINITIONS AND AGGREGATION

Macro precision and recall. The offline scorer maps each recorded oracle routing action to a registered ambiguity. An exact labeled term takes precedence; otherwise it matches annotated terms against the question, response, and recorded routing rationale. Only critical and knowledge ambiguities count. Recall is the fraction of distinct registered terms credited over the episode, and precision is the fraction of questions receiving a previously unseen credit. We macro-average defined episode scores, omitting precision when no question was asked and recall when no ambiguity was registered. Thus the eligible populations differ: pooled AmbiTab-Bird has 615 ambiguity-bearing records out of 664, and AmbiTab-BIRD-Interact has 593 out of 600. The scorer also omits recall and F1 for fallback-stop episodes; this removes one AmbiTab-AmbiQT episode in each of two of the 90 oracle-comparison runs, and all other inference recall denominators equal the ambiguity-bearing populations. When shown, F1 is the mean of episode-level harmonic means, not the harmonic mean of macro precision and recall. These are credited-coverage measures: oracle wording can affect them, including by volunteering a resolution that the question did not name.

Leakage diagnostics. Table 8 uses the four diagnostics from the three-seed campaign in Appendix E, and Figure 3(d–g) shows all four as bars. ΔJ_{resp} and ΔJ_{full} are SR differences and $\Delta \text{Rec}_{\text{none}}$ is a recall difference, all in percentage points; their deterministic-oracle entries in Table 8 are shown as dashes because the deterministic oracle serves as the reference, making these differences zero by definition. $\Delta \text{Rec}_{\text{spec}}^Q$ is raw mismatch action-credit recall in percentage points, so the

Table 8: Inference and leakage across oracles, mean $\pm$ SD. Inference: % (5 seeds); leakage: percentage points (3 seeds; Table 14) on the Bird and EW-Bird development splits and the 534-record Ambrosia dataset. No dis.: Qwen3-14B baseline; underlines: within-corpus maxima. EW-Bird LLM-oracle rows count 162 training records that exceed the oracle’s 16k-token context as failures (Appendix C.4). Dashes: zero deterministic leakage; otherwise unavailable.

Corpus	Oracle	Success rate		Asking		Leakage diagnostics				
		No dis.	Asker	Full dis.	Precision	Recall	ΔJ_{resp}	ΔJ_{full}	$\Delta \text{Rec}_{\text{none}}$	$\Delta \text{Rec}_{\text{spec}}^Q$
AmbiTab-Bird	Deterministic		66.7 $\pm$ 1.4	86.2 $\pm$ 0.7	33.0 $\pm$ 0.4	74.9 $\pm$ 0.8	—	—	—	3.5 $\pm$ 0.2
	Qwen3-14B	26.3 $\pm$ 0.7	68.7 $\pm$ 0.9	88.0 $\pm$ 0.8	32.6 $\pm$ 0.6	72.1 $\pm$ 0.9	3.9 $\pm$ 1.4	2.4 $\pm$ 4.1	7.8 $\pm$ 1.0	0.0 $\pm$ 0.0
	Gemma-3-27B		75.3 $\pm$ 1.0	87.8 $\pm$ 0.4	39.5 $\pm$ 0.3	87.7 $\pm$ 0.7	5.8 $\pm$ 0.9	3.1 $\pm$ 2.4	74.1 $\pm$ 0.2	0.8 $\pm$ 0.4
AmbiTab-EW-Bird	Deterministic		72.4 $\pm$ 0.3	80.7 $\pm$ 0.4	34.7 $\pm$ 0.1	71.5 $\pm$ 0.2	—	—	—	2.9 $\pm$ 0.3
	Qwen3-14B	48.2 $\pm$ 0.4	65.0 $\pm$ 0.8	73.9 $\pm$ 0.5	28.5 $\pm$ 0.3	55.7 $\pm$ 0.5	3.4 $\pm$ 0.2	2.8 $\pm$ 2.2	0.8 $\pm$ 0.3	0.2 $\pm$ 0.0
	Gemma-3-27B		65.4 $\pm$ 0.6	70.2 $\pm$ 0.1	38.7 $\pm$ 0.2	74.7 $\pm$ 0.4	-4.5 $\pm$ 1.8	3.6 $\pm$ 0.4	61.0 $\pm$ 0.5	1.1 $\pm$ 0.2
AmbiTab-BIRD-Interact	Deterministic		9.7 $\pm$ 0.3	10.1 $\pm$ 0.7	32.1 $\pm$ 0.2	57.7 $\pm$ 0.5	—	—	—	2.1 $\pm$ 0.2
	Qwen3-14B	7.9 $\pm$ 0.5	8.9 $\pm$ 0.6	10.0 $\pm$ 0.4	29.8 $\pm$ 0.3	53.7 $\pm$ 0.6	-0.4 $\pm$ 0.7	-0.1 $\pm$ 0.9	1.9 $\pm$ 0.1	0.1 $\pm$ 0.0
	Gemma-3-27B		9.3 $\pm$ 0.3	10.2 $\pm$ 0.8	35.7 $\pm$ 0.2	64.4 $\pm$ 0.3	0.3 $\pm$ 0.0	0.5 $\pm$ 0.0	42.7 $\pm$ 0.2	11.0 $\pm$ 0.3
AmbiTab-Ambrosia	Deterministic		35.0 $\pm$ 0.2	42.1 $\pm$ 0.2	24.0 $\pm$ 0.2	71.7 $\pm$ 0.5	—	—	—	0.5 $\pm$ 0.3
	Qwen3-14B	22.2 $\pm$ 0.3	41.2 $\pm$ 0.4	42.1 $\pm$ 0.5	25.9 $\pm$ 0.2	77.3 $\pm$ 0.6	4.0 $\pm$ 1.7	2.9 $\pm$ 0.9	54.9 $\pm$ 0.6	7.2 $\pm$ 0.6
	Gemma-3-27B		46.2 $\pm$ 0.6	43.3 $\pm$ 0.5	32.0 $\pm$ 0.1	95.4 $\pm$ 0.3	8.2 $\pm$ 2.8	3.4 $\pm$ 0.8	100.0 $\pm$ 0.0	53.2 $\pm$ 0.5
AmbiTab-AmbiQT	Deterministic		56.3 $\pm$ 0.7	73.7 $\pm$ 0.3	19.5 $\pm$ 0.2	58.3 $\pm$ 0.5	—	—	—	13.1 $\pm$ 0.3
	Qwen3-14B	33.5 $\pm$ 0.8	57.7 $\pm$ 0.4	74.6 $\pm$ 0.3	20.4 $\pm$ 0.1	60.7 $\pm$ 0.3	1.6 $\pm$ 0.5	-0.3 $\pm$ 0.2	0.3 $\pm$ 0.1	9.9 $\pm$ 0.3
	Gemma-3-27B		69.3 $\pm$ 0.7	73.0 $\pm$ 0.6	31.1 $\pm$ 0.0	93.0 $\pm$ 0.3	2.9 $\pm$ 0.7	0.4 $\pm$ 0.2	100.0 $\pm$ 0.0	70.9 $\pm$ 0.4
AmbiTab-TACO	Deterministic		72.8 $\pm$ 1.5	83.2 $\pm$ 1.0	29.1 $\pm$ 0.4	85.1 $\pm$ 0.8	—	—	—	1.0 $\pm$ 0.6
	Qwen3-14B	23.9 $\pm$ 1.4	74.1 $\pm$ 0.8	83.7 $\pm$ 1.1	28.9 $\pm$ 0.3	84.2 $\pm$ 1.3	7.0 $\pm$ 1.9	1.1 $\pm$ 1.1	20.5 $\pm$ 1.1	0.7 $\pm$ 0.6
	Gemma-3-27B		79.9 $\pm$ 2.4	83.3 $\pm$ 0.7	33.4 $\pm$ 0.3	97.6 $\pm$ 0.7	0.7 $\pm$ 0.6	0.4 $\pm$ 1.6	97.3 $\pm$ 0.0	49.5 $\pm$ 1.8

deterministic oracle retains its measured reference recall. The diagnostic values reproduce Table 14, including negative SR differences, without subtracting the deterministic reference from $\Delta \text{Rec}_{\text{spec}}^Q$.

Pooling and uncertainty. Within each inference seed, we pool corpus splits using the eligible denominator of each metric, then report the mean and sample standard deviation across seeds. The error bars describe evaluation-seed variation, not uncertainty from sampling new tasks or training new policies. In Figure 3, SR, precision, and recall use five inference seeds, while leakage uses the separate three-seed diagnostic campaign, without pairing seeds between campaigns. The populations also differ: the leakage campaign evaluates the development splits of AmbiTab-Bird (127 records) and AmbiTab-EW-Bird (446) and a 534-record AmbiTab-Ambrosia, whereas the inference columns pool 664, 2,537, and 2,670 records; the other three corpora use the same records in both campaigns. The correlations in Figure 6 use the 18 corpus–prompt means, subtracting each corpus’s own mean from both variables before computing Pearson correlation; lines are ordinary least-squares fits to each corpus’s three prompt means, and asking gain is computed relative to each run’s own no-disambiguation SR. This centering removes between-corpus mean differences; the remaining associations are descriptive, not estimates of causal effects.

C.4 COMPLETE ORACLE COMPARISON

Table 8 reports the complete oracle comparison and leakage diagnostics that Figure 3 plots. The no-disambiguation column retains only the middle (Qwen3-14B) row for each corpus; asker and full-asker success rates remain oracle-specific. The four leakage diagnostics reproduce the three-seed results of Table 14, alongside the five-seed inference results.

C.5 PROMPT-DRIVEN PRECISION–RECALL TRADEOFF

With the deterministic oracle and solver fixed, we compare three asker prompts. *Direct* asks about unclear parts of the request, while *Checklist (chat)* and *Checklist (ledger)* first enumerate potentially ambiguous terms. The checklist variants differ in dialogue representation and stopping instructions (Appendix C.6).

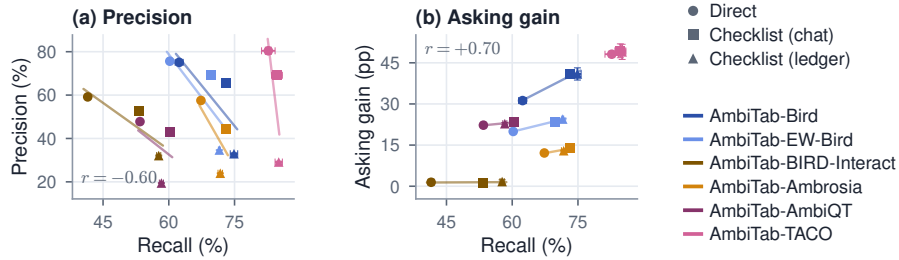


Figure 6: Prompt tradeoffs under the deterministic oracle: **(a)** precision versus recall; **(b)** asking gain versus recall. Colors identify datasets and shapes identify prompts; error bars show seed standard deviations. Lines fit each dataset’s three prompt means; correlations remove dataset means (Appendix C.3).

Table 9: Clarification strategies. All prompts request at most one clarification question per turn.

Strategy	Instructions and history
Direct	Ask about unclear parts of the request without an explicit checklist; stop when the request is clear or no different ambiguity remains to be raised.
Checklist (chat)	Enumerate potentially ambiguous terms; read prior turns as native chat messages; stop when all terms are resolved or no distinct unasked term remains.
Checklist (ledger)	Enumerate potentially ambiguous terms; read the original request and explicit lists of questions with resolved and unavailable responses; continue while an ambiguity remains open, subject to the budget.

Both checklist prompts increase recall and reduce precision relative to Direct on every dataset (Figure 6a). We use Checklist (ledger) by default because it has the highest mean recall across datasets. Higher recall is associated with greater asking gain across prompts, though not for every prompt pair (Figure 6b). This precision–recall tradeoff motivates training the asker to improve both.

C.6 CLARIFICATION PROMPTS AND COMPLETE PROMPT COMPARISON

The three strategies use the same question budget, deterministic oracle, and solver. Table 9 summarizes their instructions and dialogue history. Checklist (ledger) is the default for the oracle and trained-policy comparisons. Checklist (chat) differs from it in both dialogue representation and stopping instructions, so the comparison does not isolate the contribution of either change.

Table 10 reports the full prompt comparison. Only the evaluated asker uses the varied clarification prompt; the no-disambiguation policy and scripted full asker are unchanged, so their SR columns are omitted from this table. The default strategy reuses its five-seed deterministic-oracle evaluation; the other strategies each use three seeds. The mean rows average each metric equally across the six corpora.

D TRAINING THE ASKING POLICY: IMPLEMENTATION AND ADDITIONAL RESULTS

This appendix specifies the rewards, optimization settings, and evaluation protocol for the main question-F1 policy and ablations in Section 4.4. We distinguish native validation, which uses the adapted Qwen3-8B model to generate SQL, from the common inference evaluation, which pairs each asker with the fixed Coder-30B solver. The distinction matters because improvements in the native solver can change success rate without improving clarification.

Table 10: Clarification-prompt comparison under the deterministic oracle and otherwise the setup of Table 8, which the Checklist (ledger) rows repeat. Corpus entries are $\text{mean} \pm \text{SD}$ over 5 seeds for Checklist (ledger) and 3 for Direct and Checklist (chat); rates are percentages. Mean rows average the six corpus means equally, without SD; bold marks the highest mean recall. All prompts share the same turn budget; AmbiTab-Bird and AmbiTab-EW-Bird pool training and development splits. The two control arms do not read the prompt and are omitted. All rows of a corpus run the same instances; Table 1 gives their counts.

Corpus	Prompt	Success rate	Asking	
		Asker	Precision	Recall
AmbiTab-Bird	Direct	56.6 $\pm$ 0.4	75.1 $\pm$ 0.9	62.4 $\pm$ 0.9
	Checklist (chat)	66.1 $\pm$ 0.8	65.5 $\pm$ 0.5	73.2 $\pm$ 0.8
	Checklist (ledger)	66.7 $\pm$ 1.4	33.0 $\pm$ 0.4	74.9 $\pm$ 0.8
AmbiTab-EW-Bird	Direct	68.3 $\pm$ 0.6	75.6 $\pm$ 0.1	60.2 $\pm$ 0.3
	Checklist (chat)	71.3 $\pm$ 0.2	69.2 $\pm$ 0.5	69.7 $\pm$ 0.3
	Checklist (ledger)	72.4 $\pm$ 0.3	34.7 $\pm$ 0.1	71.5 $\pm$ 0.2
AmbiTab-BIRD-Interact	Direct	9.6 $\pm$ 0.9	59.2 $\pm$ 0.9	41.5 $\pm$ 0.3
	Checklist (chat)	9.2 $\pm$ 0.4	52.8 $\pm$ 0.5	53.2 $\pm$ 0.4
	Checklist (ledger)	9.7 $\pm$ 0.3	32.1 $\pm$ 0.2	57.7 $\pm$ 0.5
AmbiTab-Ambrosia	Direct	34.4 $\pm$ 0.5	57.5 $\pm$ 0.4	67.3 $\pm$ 0.2
	Checklist (chat)	36.2 $\pm$ 0.6	44.4 $\pm$ 0.1	73.1 $\pm$ 0.3
	Checklist (ledger)	35.0 $\pm$ 0.2	24.0 $\pm$ 0.2	71.7 $\pm$ 0.5
AmbiTab-AmbiQT	Direct	55.9 $\pm$ 0.5	47.8 $\pm$ 0.6	53.4 $\pm$ 0.6
	Checklist (chat)	57.1 $\pm$ 0.9	43.1 $\pm$ 0.4	60.4 $\pm$ 0.5
	Checklist (ledger)	56.3 $\pm$ 0.7	19.5 $\pm$ 0.2	58.3 $\pm$ 0.5
AmbiTab-TACO	Direct	72.5 $\pm$ 1.2	80.4 $\pm$ 1.6	82.7 $\pm$ 1.5
	Checklist (chat)	73.6 $\pm$ 1.3	69.1 $\pm$ 0.3	84.7 $\pm$ 1.2
	Checklist (ledger)	72.8 $\pm$ 1.5	29.1 $\pm$ 0.4	85.1 $\pm$ 0.8
Mean	Direct	49.5	65.9	61.3
	Checklist (chat)	52.3	57.4	69.1
	Checklist (ledger)	52.2	28.7	69.9

D.1 QUESTION-F1 REWARD AND TRAJECTORY-LEVEL CREDIT

Our main configuration trains Qwen3-8B on the 537-request AmbiTab-Bird training split with the deterministic oracle and the question-F1 reward of Equation (2). We use a multi-turn GRPO variant with DAPO-style asymmetric clipping and zero-advantage group filtering (Shao et al., 2024; Yu et al., 2025). For a trajectory τ as in Section 2, let n be its number of questions and use the credited ambiguity sets $\text{cred}_i \subseteq \text{Amb}$ and $\text{cred}_{\leq i} = \bigcup_{j=1}^i \text{cred}_j$ from Definition 2. When $n > 0$ and $|\text{Amb}| > 0$, define the episode precision and recall terms

$$p(\tau) = \frac{|\{i \in \{1, \dots, n\} : \text{cred}_i \not\subseteq \text{cred}_{\leq i-1}\}|}{n}, \quad r(\tau) = \frac{|\text{cred}_{\leq n}|}{|\text{Amb}|}. \quad (1)$$

These are the quantities averaged in the theory’s Prec and Rec. The question-F1 reward takes their harmonic mean within each episode, with explicit conventions for requests without ambiguities and trajectories without productive questions:

$$R_{\text{F1}}(\tau) = \begin{cases} 1, & |\text{Amb}| = 0 \text{ and } n = 0, \\ \frac{2p(\tau)r(\tau)}{p(\tau) + r(\tau)}, & |\text{Amb}| > 0, n > 0, p(\tau) + r(\tau) > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Thus R_{F1} is not the harmonic mean of the population metrics Prec and Rec. The annotation-based transcript scorer supplies the credits against the registered ambiguities. A request with ambiguities receives zero reward if no question is productive, including when the asker stops immediately. A request without ambiguities receives one for abstaining and zero for asking, which defines the training reward where recall is undefined; native rollouts give such requests a single decision, meant for

stopping, so the zero applies to a question emitted instead. Repeated and irrelevant questions reduce $p(\tau)$ without increasing $r(\tau)$; a stop decision does not count toward n .

Ablation objective. The *outcome reward* uses the theory’s interpretation-dependent verifier and the final submitted solution $\hat{s}$:

$$R_{\text{SR}}(\theta, \tau) = V(\theta, \hat{s}) \in \{0, 1\}. \quad (3)$$

Its expectation is the success rate $J(\pi_{\text{ask}}; \mathcal{O})$ for the solver used during training. Question-F1 training ends after clarification, whereas outcome-reward training also generates and verifies SQL. Changing the training oracle can change both the dialogue and the credits supplied by the transcript scorer.

Trajectory-level advantages. For each request, we sample $K = 16$ independent trajectories $\tau_1, \dots, \tau_K$ and standardize their rewards, with R denoting the chosen reward above and suppressing the fixed interpretation θ for outcome reward:

$$\hat{A}_k = \frac{R(\tau_k) - \bar{R}}{s_R + 10^{-6}}, \quad \bar{R} = \frac{1}{K} \sum_{k=1}^K R(\tau_k), \quad s_R^2 = \frac{1}{K-1} \sum_{k=1}^K (R(\tau_k) - \bar{R})^2. \quad (4)$$

Each retained clarification output, including a stop decision, receives its trajectory’s advantage. Oracle responses and SQL outputs carry no loss. Groups with constant reward, or reward standard deviation at most 10^{-6} , receive zero advantage and are filtered before the actor update. If every group has zero advantage, one complete group is retained to avoid an empty training batch; only the regularization terms below then contribute to the loss gradient.

Per-turn policy objective. Let j index retained clarification decisions in an optimizer batch $\mathcal{B}$, let $k(j)$ identify the trajectory containing decision j , and let T_j be its number of unmasked output tokens. For output token y_{jt} with dialogue and token prefix h_{jt} , define the probability ratio and sampled reference penalty as

$$\rho_{jt}(\phi) = \frac{\pi_\phi(y_{jt} | h_{jt})}{\pi_{\text{old}}(y_{jt} | h_{jt})}, \quad d_{jt}^{\text{k2}}(\phi) = \frac{1}{2} \left(\log \frac{\pi_\phi(y_{jt} | h_{jt})}{\pi_{\text{ref}}(y_{jt} | h_{jt})} \right)^2, \quad (5)$$

where ϕ denotes trainable policy parameters, π_{old} is the pre-update actor, and π_{ref} is the frozen reference policy. Writing $H_{jt}(\phi)$ for the entropy of $\pi_\phi(\cdot | h_{jt})$, the minimized loss is

$$\begin{aligned} \mathcal{L}(\phi) &= \frac{1}{B} \sum_{j \in \mathcal{B}} \frac{1}{T_j} \sum_{t=1}^{T_j} \ell_{jt}(\phi), \\ \ell_{jt}(\phi) &= -\min \left\{ \rho_{jt}(\phi) \hat{A}_{k(j)}, \text{clip}(\rho_{jt}(\phi), 0.8, 1.28) \hat{A}_{k(j)} \right\} \\ &\quad + 0.1 d_{jt}^{\text{k2}}(\phi) - 0.01 H_{jt}(\phi). \end{aligned} \quad (6)$$

The batch normalization uses the nominal decision-sample batch size $B = 512$, including for partial final batches. Tokens are averaged within each decision, and decision losses are summed without normalizing by the number of decisions in a trajectory; dialogues with more retained decisions therefore contribute more terms.

D.2 HYPERPARAMETERS AND SHARED MODEL PARAMETERS

Table 11 reports the training hyperparameters. For the main configuration with the deterministic oracle and question-F1 reward, the selected checkpoint follows 40 rollout batches. Each rollout batch samples eight requests and 16 trajectories per request. Each retained model decision becomes a separate training sample, with a nominal global batch size of 512 samples, partial final batches, and dynamic token packing. With zero-advantage filtering, each rollout batch retained at most one nominal batch of decision samples, so every rollout batch produced exactly one optimizer update. The learning-rate schedule assumes four decisions per trajectory; this estimate does not cap the number of retained decisions.

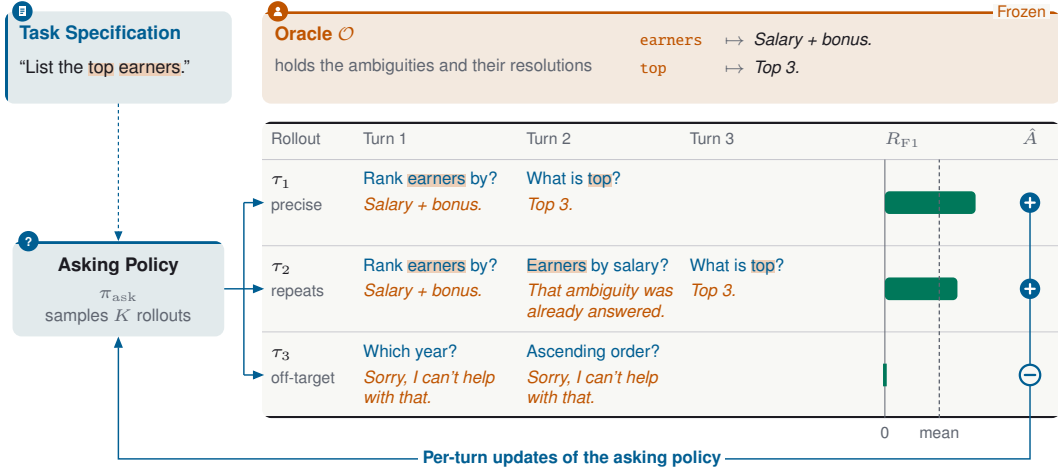


Figure 7: Asking-policy training with a multi-turn GRPO variant with DAPO-style asymmetric clipping and zero-advantage group filtering. The main question-F1 reward R_{F1} balances productive questions and ambiguity coverage; every retained asking decision receives its trajectory’s normalized advantage. Three trajectories illustrate the update; training uses $K = 16$.

Table 11: Asker-training hyperparameters. Oracle and reward choices vary as specified in Appendix D.3.

Setting	Value
Initialization / backend	Qwen3-8B / FSDP
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.98)$, $\epsilon = 10^{-8}$
Learning rate / schedule	Peak 5×10^{-5} ; cosine decay to 0, 5% linear warmup
Schedule horizon	136 optimizer updates, estimated from four decisions per trajectory
Weight decay / gradient clipping	0 / 1.0
LoRA rank / alpha / dropout	32 / 64 / 0 (adapter scaling $\alpha/r = 2$)
LoRA targets	Attention q/k/v/o and feed-forward gate/up/down projections
Requests per rollout / group size	8 / 16 trajectories per request
Rollout / nominal training batch	128 trajectories / 512 retained decision samples
Policy ratio clipping	[0.8, 1.28]
Loss aggregation	Token mean per retained decision; sum divided by nominal batch size
Group filtering	Zero-advantage groups dropped; retain one group if all are zero
Reference-policy penalty	Sampled k2, coefficient 0.1
Entropy coefficient	0.01
Policy decoding	Temperature 0.8; thinking disabled
Main reward	Question F1 (R_{F1} ; Equation (2))
Main training oracle	Deterministic; no oracle LLM is instantiated
LLM-oracle ablation	Qwen3-14B; temperature 0; thinking disabled
Rollout context / response ceiling	32,768 / 256 tokens
Clarification prompt	Checklist (ledger); question budget visible
Question budget	At most $ \text{Amb} + 2$ questions for $ \text{Amb} > 0$, then a stop decision
Checkpoint / validation cadence	Every eight rollout batches; initial validation
Native validation samples	One per request

The asker and native SQL solver share the same frozen Qwen3-8B base and trainable LoRA adapter. Masking SQL outputs out of the loss does not freeze the solver: updates from clarification tokens also change the adapter used for SQL generation.

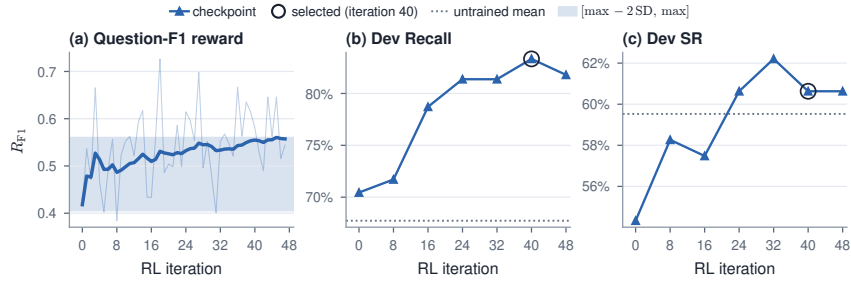


Figure 8: Learning curves through iteration 48 for the main deterministic-oracle configuration with question-F1 reward R_{F1} : (a) training reward, (b) native AmbiTab-Bird development recall, and (c) SR. Faint and solid lines show mean trajectory rewards per rollout batch and their smoothed trend; rings mark the selected checkpoint at iteration 40. The blue band in (a) extends from the maximum smoothed reward down by two sample standard deviations of the per-batch mean rewards. Dotted lines show the untrained development-only inference mean across five seeds, using the deterministic-oracle runs underlying Table 8. Native and reference evaluations use different protocols and solvers.

Native rollouts on requests with registered ambiguities allow $|\text{Amb}| + 3$ decisions, reserving the final decision for stopping. On requests without registered ambiguities, only a stopping decision is allowed. Question-F1 training skips SQL generation.

D.3 ABLATION DESIGN AND CHECKPOINT SELECTION

We compare the main policy trained with question-F1 reward and the deterministic oracle with two outcome-reward ablations, one using the deterministic oracle and one using Qwen3-14B as the oracle. All three use AmbiTab-Bird and its 127-request development split, with one training seed per configuration. Every configuration updates clarification outputs only. We evaluate the designated checkpoints: iteration 40 for question-F1 reward, 16 for outcome reward with the deterministic oracle, and 40 for outcome reward with the LLM oracle. These checkpoints are fixed for the reported comparison rather than selected by a common validation-score rule. All RL learning curves and ablation figures use a shared plotting window through iteration 48. This window contains zero-based rollout batches 0-47, which precede checkpoint 48, and native evaluations through checkpoint 48. All three configurations have complete common-inference evaluations across three seeds on all six corpora. The untrained reference retains its existing five seeds. The AmbiTab-Bird development split was used to inspect training and checkpoints, so its scores are development results, not held-out test estimates. Because every rollout batch produced one optimizer update, the shared window corresponds to 48 updates per configuration and the evaluated checkpoints to 40, 16, and 40 updates.

D.4 NATIVE LEARNING CURVES AND OBJECTIVE MISMATCH

Figures 8 to 10 and table 12 separate changes in clarification quality from changes in native SQL success.

In Figure 8, training reward averages one reward per logged trajectory before groups without training signal are filtered; the solid curve is a debiased exponential moving average with weight 0.95. Iteration 0 is the run’s native evaluation before training. Native checkpoints use the Qwen3-8B solver and native validation protocol; the untrained reference uses the fixed Coder30B solver and common inference protocol. The reference uses the 127-request AmbiTab-Bird development slice of the deterministic-oracle inference runs underlying Table 8, which pools training and development requests in its displayed AmbiTab-Bird row. The dotted reference lines show the development-only mean across five seeds. Common-solver results appear in Table 2.

In Figure 9(a-c), the trained checkpoints use the deterministic validation oracle and adapted Qwen3-8B solver. Every row shows differences from the same untrained Qwen3-8B asker reference on the 127-request AmbiTab-Bird development split, averaged over all five evaluation seeds from the deterministic-oracle inference runs underlying Table 8. Gray reference bands show one sample standard deviation on either side of the reference mean, using the same development-only reference

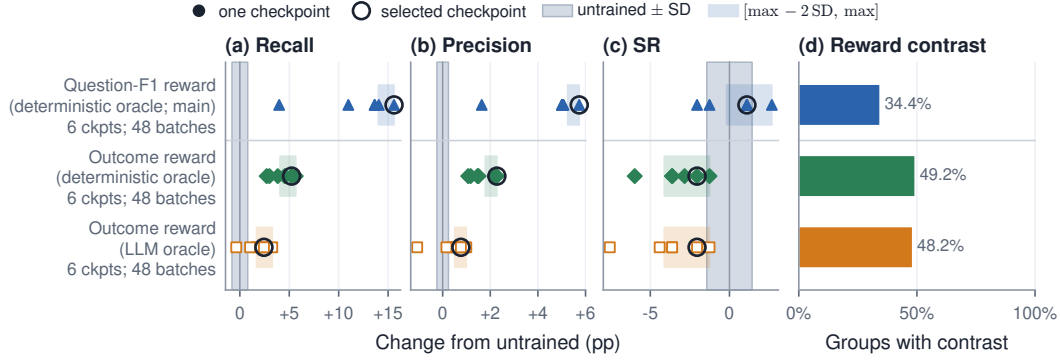


Figure 9: The main policy trained with question-F1 reward and the oracle/reward ablations through iteration 48: native AmbiTab-Bird development checkpoints relative to the five-seed untrained inference reference. **(a-c)** Recall, precision, and SR differences in percentage points. Rings mark the designated evaluation checkpoints, including iteration 16 for outcome reward with the deterministic oracle. Gray shading shows $\pm$ one sample standard deviation across the five development-only reference seeds from the deterministic-oracle runs underlying Table 8. Colored row bands span $[\max - 2 \text{ SD}, \max]$, using each row’s maximum trained-checkpoint value and the corresponding reference SD. Trained checkpoints use native validation with the Qwen3-8B solver; the reference uses the common inference protocol with the fixed Coder-30B solver. **(d)** Fraction of logged request groups with nonconstant trajectory rewards before group filtering, pooling counts over the same 48 rollout batches for each configuration.

as Figure 8. Colored row bands show a two-SD tolerance: each extends from that row’s maximum trained-checkpoint value down by twice the corresponding reference standard deviation. The reference uses the common inference protocol and fixed Coder-30B solver, whereas the trained checkpoints use native validation; these differences therefore include protocol effects, and SR also includes the solver difference. Table 13 provides the comparison under the common inference protocol. Colored points represent the available trained checkpoints, and rings identify the designated evaluation checkpoint for each configuration. The main policy trained with question-F1 reward appears first, followed by the ablations. Row labels give the numbers of trained checkpoints and rollout batches; R_{SR} denotes the outcome reward. Each configuration contributes one training run, so the checkpoint comparisons are descriptive.

At iteration 40, the main policy trained with question-F1 reward reaches 83.3% native recall and 60.6% native SR, compared with 70.4% and 54.3% at initialization. With outcome reward, the deterministic-oracle policy at iteration 16 reaches 73.0% recall and 57.5% SR; the LLM-oracle policy at iteration 40 reaches 70.2% recall and 57.5% SR. Outcome reward depends on the solver’s ability to use the responses; F1 balances question efficiency and ambiguity coverage. The fixed-solver evaluation in Table 13 measures the trained asking policies under a common solution policy and protocol. The two evaluations need not rank checkpoints in the same order.

The outcome-reward runs provide the training-oracle comparison under the common inference protocol. There is no LLM-oracle F1 run in this ablation set. One training seed per configuration does not establish a general ranking of objectives or training oracles.

Dataset choice matters. A separate outcome-reward run on AmbiTab-BIRD-Interact crashed before completing training. Its trajectory failed to sustain improvements in clarification quality over the shown window (Figure 10), illustrating that the training dataset matters for learning to ask. The run uses the LLM oracle and its 120-request native validation split; the shown window extends through iteration 48. The columns show reward, training recall, training precision, and optimization loss. Faint lines show rollout batches; solid lines use a debiased exponential moving average with weight 0.95. Reward averages retained decision samples after constant-reward groups are filtered, so it is not the unfiltered rollout SR. Native recall falls from 66.8% at iteration 8 to 59.4% at the end of the shown window, iteration 48, where native SR remains low at 6.7%. Few request groups provide differing binary rewards, limiting the within-request reward contrast available for policy

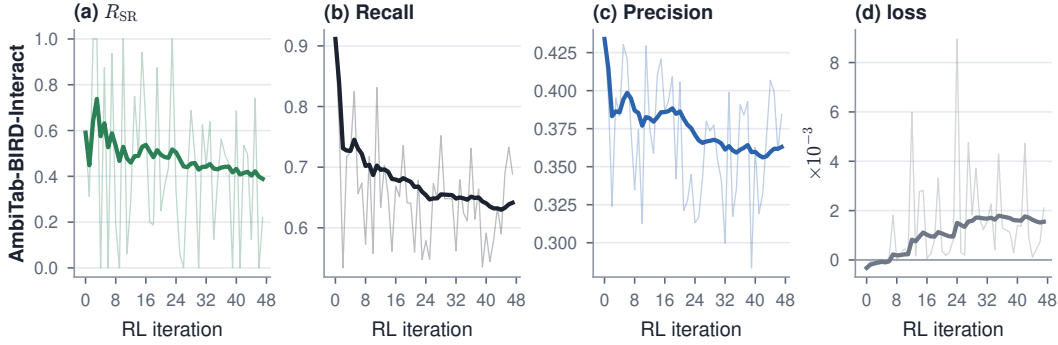


Figure 10: Outcome-reward training dynamics on AmbiTab-BIRD-Interact through iteration 48. Training recall and precision decline over the window, and binary rewards provide limited within-request contrast.

Table 12: Native validation before training and at the selected checkpoint, using the adapted Qwen3-8B solver. Rates are percentages. All rows use AmbiTab-Bird dev ($n = 127$). Question-F1 reward is the main configuration, trained with the deterministic oracle; the remaining trained policies are ablations. Selection and oracle settings are given in Appendices D.3 and D.5.

Policy	SR	Precision	Recall	F1
Question-F1 reward (untrained)	54.3	32.2	70.4	43.7
Question-F1 reward	60.6	36.8	83.3	50.6
Outcome reward (deterministic oracle) (untrained)	55.9	31.0	68.1	42.2
Outcome reward (deterministic oracle)	57.5	33.3	73.0	45.3
Outcome reward (LLM oracle) (untrained)	52.8	30.6	68.1	41.9
Outcome reward (LLM oracle)	57.5	31.8	70.2	43.4

updates. The curves document the shown training window but do not identify the cause of the later crash. This run also used a larger response budget and a different schedule estimate than the current AmbiTab-Bird runs, so it does not isolate the effect of the training corpus. We retain this run only as an illustration of unsuccessful training; it is excluded from the configuration and results tables.

Reward contrast. Figure 9(d) reports the fraction of request groups whose available trajectory rewards are not all equal, with 16 trajectories sampled per request. Counts are logged before group filtering and pooled across the same 48 rollout batches for each configuration; row labels give the number of batches. Such groups supply the within-request reward contrast used for policy updates, whereas constant-reward groups have zero policy-gradient advantage (Equation (4)). These fractions describe reward contrast over the shared plotting window. They measure available reward contrast rather than question quality.

D.5 COMMON INFERENCE PROTOCOL AND AGGREGATION

Fixed solver and oracle. Table 13 evaluates the trained askers with the deterministic oracle and a fixed Qwen3-Coder-30B-A3B-Instruct solver, using Checklist (ledger), temperature 0.7, and no SQL debugging. The main and complete tables use the same designated checkpoints; the question-F1 policy and both outcome-reward ablations each have three completed evaluation seeds. The question budget matches native rollouts on requests with registered ambiguities; requests without ambiguities skip clarification entirely. Native validation instead generates SQL with the adapted Qwen3-8B policy at temperature 0.8. On AmbiTab-Bird dev, its oracle is deterministic on the 119 requests with registered ambiguities and silent on the remaining eight, including for policies trained with the LLM oracle. AmbiTab-BIRD-Interact native validation uses the LLM oracle. The untrained rows in Table 12 report each run’s own iteration-0 native validation with the untrained Qwen3-8B solver and its single recorded training seed.

Untrained reference and uncertainty. The untrained reference uses the Qwen3-8B asker with the same clarification protocol, except for nucleus sampling: its asker samples at $\text{top-}p = 0.8$, whereas the trained-policy evaluations leave the asker at the serving default of $\text{top-}p = 0.95$; asker temperature, $\text{top-}k = 20$, solver decoding, prompt, budget, and oracle are identical. Its scores retain all five evaluation seeds from the deterministic-oracle runs used in Table 8, taking the development split alone for AmbiTab-Bird and AmbiTab-EW-Bird. Trained-policy cells average all three evaluation seeds; Table 13 also reports their sample standard deviations. These seeds vary inference for one trained policy per configuration; they do not measure variation across training runs. All gains subtract the five-seed untrained mean from the trained policy’s three-seed mean, including the differences in Table 2. Table 8 instead pools each of these two corpora’s train and development splits.

Eligible populations. SR averages $V(\theta, \hat{s})$ over requests. Precision and recall are macro averages of the episode quantities in Definition 2, omitting precision when $n = 0$ and recall when $|\text{Amb}| = 0$. Their denominators can differ: for example, AmbiTab-Bird dev recall uses 119 of 127 requests. Reported F1 is the average of episode-level F1, not the harmonic mean of macro precision and recall.

AmbiTab-BIRD-Interact pools its full dataset within each evaluation seed using each metric’s eligible denominator. AmbiTab-Ambrosia uses the full ambiguous split of Ambrosia, and AmbiTab-AmbiQT uses the column-synonym subset of AmbiQT. The Mean column in Table 13 equally weights all six evaluation families, AmbiTab-Bird dev, AmbiTab-EW-Bird dev, AmbiTab-BIRD-Interact, AmbiTab-Ambrosia, AmbiTab-AmbiQT, and AmbiTab-TACO, instead of pooling requests. For the family mean, we first average all six families within each evaluation seed, then report the mean and sample standard deviation of these per-seed averages over all three trained-policy seeds or all five untrained-reference seeds.

D.6 COMPLETE INFERENCE RESULTS

Table 13 reports complete three-seed evaluations of the main policy trained with question-F1 reward and both outcome-reward ablations across all six corpora. The main-text summary and appendix use the same runs and checkpoints. For outcome-reward training, the policy trained with the deterministic oracle at iteration 16 has six-corpus means of 29.2% precision, 70.7% recall, and 51.7% SR, compared with 29.2%, 70.9%, and 51.9% for the policy trained with the LLM oracle. Both outcome-reward ablations fall below the question-F1 policy on these three means, while differences between the two training oracles are small and vary across corpora.

E LEAKAGE EXPERIMENTS

All runs use the main evaluation settings: the Qwen3-8B asker with Checklist (ledger), the fixed Qwen3-Coder-30B-A3B-Instruct solver, the Qwen3-14B and Gemma-3-27B oracles, all six datasets, and three evaluation seeds (234 runs, 141,024 episodes). The datasets are the development splits of AmbiTab-Bird (127 records) and AmbiTab-EW-Bird (446), a 534-record AmbiTab-Ambrosia, and the full AmbiTab-BIRD-Interact, AmbiTab-AmbiQT, and AmbiTab-TACO pools. Levels are mean $\pm$ sample standard deviation over seeds; paired differences carry 95% AVT-cluster bootstrap intervals. All values are in percentage points.

Untrained asker. Table 14 reports the four diagnostics. ΔJ_{resp} and ΔJ_{full} are SR differences; $\Delta \text{Rec}_{\text{spec}}^Q$ is a recall and $\Delta \text{Rec}_{\text{none}}$ a recall difference, both over the records with at least one ambiguity. The two LLM-oracle entries are not paired estimates of their difference. On AmbiTab-AmbiQT the reference recall of the specification-randomized policy is 13.1%, so $\Delta \text{Rec}_{\text{spec}}^Q$ there carries a large reference term; differences between oracles are unaffected, because the term cancels.

E.1 RESPONDER ACCESS TO THE REFERENCE SQL

We repeat the experiments of Table 14 with one change: the LLM responder’s access to the full reference SQL. On AmbiTab-Ambrosia, AmbiTab-AmbiQT, and AmbiTab-TACO the default responder does not read it, and the new runs add it; on AmbiTab-EW-Bird and AmbiTab-Bird the

Table 13: Training and transfer with a fixed Coder-30B solver and deterministic oracle. Question-F1 reward is the main configuration, trained with the deterministic oracle; the remaining trained policies are ablations. Question-F1 reward uses iteration 40; Outcome reward (deterministic oracle) uses iteration 16; Outcome reward (LLM oracle) uses iteration 40. Selected trained policies use all three evaluation seeds; the untrained baseline retains five seeds. Rates are percentages, reported as means with sample SD in subscripts. Mean equally weights all six families within each seed, using only seeds complete across all six. Bold values mark the highest mean in each column within each metric, including ties. AmbiTab-Bird and AmbiTab-EW-Bird use dev; AmbiTab-AmbiQT uses column synonyms. Dashes denote unavailable results. Protocol details appear in Appendices D.3 and D.5.

SQL success							
Policy	AmbiTab-Bird	AmbiTab-EW-Bird	AmbiTab-BIRD-Interact	AmbiTab-Ambrosia	AmbiTab-AmbiQT	AmbiTab-TACO	Mean
Untrained	59.5 $\pm$ 1.4	69.0 $\pm$ 0.8	9.7 $\pm$ 0.3	35.0 $\pm$ 0.2	56.3 $\pm$ 0.7	72.8 $\pm$ 1.5	50.4 $\pm$ 0.4
Question-F1 reward	71.9$\pm$0.5	72.2$\pm$1.0	9.7 $\pm$ 0.3	37.3$\pm$0.2	59.4$\pm$0.3	77.0$\pm$1.4	54.6$\pm$0.2
Outcome reward (deterministic oracle)	62.7 $\pm$ 1.2	71.1 $\pm$ 1.6	9.8$\pm$0.1	35.5 $\pm$ 0.7	56.8 $\pm$ 0.2	74.3 $\pm$ 1.9	51.7 $\pm$ 0.4
Outcome reward (LLM oracle)	64.3 $\pm$ 1.2	71.4 $\pm$ 1.6	9.7 $\pm$ 0.6	35.9 $\pm$ 0.6	56.8 $\pm$ 0.5	73.4 $\pm$ 0.8	51.9 $\pm$ 0.6
Precision							
Policy	AmbiTab-Bird	AmbiTab-EW-Bird	AmbiTab-BIRD-Interact	AmbiTab-Ambrosia	AmbiTab-AmbiQT	AmbiTab-TACO	Mean
Untrained	31.0 $\pm$ 0.2	33.6 $\pm$ 0.5	32.1 $\pm$ 0.2	24.0 $\pm$ 0.2	19.5 $\pm$ 0.2	29.1 $\pm$ 0.4	28.2 $\pm$ 0.1
Question-F1 reward	37.0$\pm$0.5	38.0$\pm$0.3	36.4$\pm$0.4	27.3$\pm$0.1	21.7$\pm$0.1	30.7$\pm$0.3	31.8$\pm$0.2
Outcome reward (deterministic oracle)	33.4 $\pm$ 0.4	34.6 $\pm$ 0.3	33.0 $\pm$ 0.7	24.6 $\pm$ 0.1	19.8 $\pm$ 0.2	29.7 $\pm$ 0.5	29.2 $\pm$ 0.2
Outcome reward (LLM oracle)	32.2 $\pm$ 0.7	35.8 $\pm$ 0.5	33.1 $\pm$ 0.4	24.5 $\pm$ 0.1	19.7 $\pm$ 0.1	29.8 $\pm$ 0.5	29.2 $\pm$ 0.2
Recall							
Policy	AmbiTab-Bird	AmbiTab-EW-Bird	AmbiTab-BIRD-Interact	AmbiTab-Ambrosia	AmbiTab-AmbiQT	AmbiTab-TACO	Mean
Untrained	67.7 $\pm$ 0.8	69.8 $\pm$ 1.1	57.7 $\pm$ 0.5	71.7 $\pm$ 0.5	58.3 $\pm$ 0.5	85.1 $\pm$ 0.8	68.4 $\pm$ 0.3
Question-F1 reward	83.4$\pm$1.1	79.3$\pm$0.6	65.1$\pm$0.7	81.8$\pm$0.2	65.0$\pm$0.3	89.8$\pm$0.8	77.4$\pm$0.3
Outcome reward (deterministic oracle)	72.9 $\pm$ 0.5	72.0 $\pm$ 0.7	59.2 $\pm$ 1.4	73.7 $\pm$ 0.4	59.3 $\pm$ 0.6	87.0 $\pm$ 1.5	70.7 $\pm$ 0.3
Outcome reward (LLM oracle)	71.3 $\pm$ 1.7	74.7 $\pm$ 0.5	59.4 $\pm$ 0.6	73.5 $\pm$ 0.2	59.2 $\pm$ 0.3	87.3 $\pm$ 1.5	70.9 $\pm$ 0.5

default responder reads it and its clause segments, and the new runs remove both. The matcher is unchanged, so each difference compares two runs of the same policy under the actual oracle, paired by task: the untrained asker for ΔJ_{resp} , the adversarial full-disambiguation policy for ΔJ_{full} , and the generic policy for $\Delta \text{Rec}_{\text{none}}$. The new runs comprise 82 evaluation runs and 47,640 episodes; a few untrained-asker runs with the SQL shown are reused from an earlier study with the same settings. The two conditions come from different inference-server sessions.

Showing the SQL raises ΔJ_{full} on AmbiTab-Ambrosia and AmbiTab-AmbiQT under both oracles, by 1.24 to 10.36 points, and ΔJ_{resp} under Gemma on both and under Qwen on AmbiTab-Ambrosia, by 1.82 to 7.05 points; all these intervals exclude zero. On AmbiTab-EW-Bird and AmbiTab-Bird, removing it lowers ΔJ_{resp} in all four dataset-oracle pairs, by 3.15 to 7.09 points, with intervals excluding zero in three, and lowers ΔJ_{full} on AmbiTab-EW-Bird under both oracles, by 2.84 and 4.86 points, so Table 14 includes this channel there. On AmbiTab-TACO every interval includes zero: no effect is detectable at this sample size. $\Delta \text{Rec}_{\text{none}}$, which depends only on the unchanged matcher, stays within 0.36 points of zero except for Qwen on AmbiTab-TACO, -2.50 , whose interval reaches zero. Hidden SQL can still reach the responder through annotations, the schema, and matcher actions that quote an SQL segment.

Table 14: Leakage diagnostics of the untrained asker, mean $\pm$ SD over three seeds. AmbiTab-Bird and AmbiTab-EW-Bird are development splits, and AmbiTab-Ambrosia has 534 records. Bold marks the larger of the two LLM-oracle values. For the deterministic oracle, which is the reference, ΔJ_{resp} , ΔJ_{full} , and $\Delta \text{Rec}_{\text{none}}$ are 0 by definition, and $\Delta \text{Rec}_{\text{spec}}^Q$ is the reference recall of the specification-randomized policy.

Oracle	Dataset	ΔJ_{resp}	$\Delta \text{Rec}_{\text{spec}}^Q$	ΔJ_{full}	$\Delta \text{Rec}_{\text{none}}$
Qwen3-14B	AmbiTab-Bird dev	3.94 ± 1.36	0.00 ± 0.00	2.36 ± 4.09	7.84 ± 0.97
	AmbiTab-EW-Bird dev	3.36 ± 0.22	0.22 ± 0.00	2.77 ± 2.15	0.82 ± 0.26
	AmbiTab-BIRD-Interact	-0.44 ± 0.67	0.14 ± 0.05	-0.11 ± 0.95	1.88 ± 0.10
	AmbiTab-Ambrosia	4.00 ± 1.68	7.24 ± 0.57	2.93 ± 0.94	54.87 ± 0.56
	AmbiTab-AmbiQT	1.61 ± 0.47	9.85 ± 0.29	-0.31 ± 0.23	0.35 ± 0.10
	AmbiTab-TACO	6.95 ± 1.93	0.71 ± 0.56	1.07 ± 1.07	20.50 ± 1.11
Gemma-3-27B	AmbiTab-Bird dev	5.77 ± 0.91	0.84 ± 0.42	3.15 ± 2.36	74.09 ± 0.24
	AmbiTab-EW-Bird dev	-4.48 ± 1.78	1.10 ± 0.24	3.59 ± 0.45	60.99 ± 0.47
	AmbiTab-BIRD-Interact	0.33 ± 0.00	11.04 ± 0.28	0.50 ± 0.00	42.69 ± 0.21
	AmbiTab-Ambrosia	8.24 ± 2.76	53.25 ± 0.47	3.37 ± 0.82	100.00 ± 0.00
	AmbiTab-AmbiQT	2.94 ± 0.65	70.94 ± 0.44	0.37 ± 0.19	100.00 ± 0.00
	AmbiTab-TACO	0.71 ± 0.62	49.47 ± 1.75	0.36 ± 1.63	97.33 ± 0.00
Deterministic	AmbiTab-Bird dev	0	3.50 ± 0.24	0	0
	AmbiTab-EW-Bird dev	0	2.91 ± 0.29	0	0
	AmbiTab-BIRD-Interact	0	2.13 ± 0.24	0	0
	AmbiTab-Ambrosia	0	0.50 ± 0.29	0	0
	AmbiTab-AmbiQT	0	13.09 ± 0.32	0	0
	AmbiTab-TACO	0	0.98 ± 0.62	0	0

Table 15: Change in the leakage diagnostics when the LLM responder reads the full reference SQL, shown minus hidden, three seeds pooled, with paired 95% AVT-cluster bootstrap intervals. AmbiTab-Bird and AmbiTab-EW-Bird are development splits, and AmbiTab-Ambrosia is a 534-record. New run: the condition added for this study.

Oracle	Dataset	New run	Difference, reference SQL shown minus hidden		
			ΔJ_{resp}	ΔJ_{full}	$\Delta \text{Rec}_{\text{none}}$
Qwen3-14B	AmbiTab-Bird dev	SQL hidden	$3.15 [-0.26, 6.56]$	$0.52 [-2.36, 3.67]$	$0.28 [0.00, 1.12]$
	AmbiTab-EW-Bird dev	SQL hidden	$3.96 [1.42, 6.35]$	$2.84 [0.22, 5.61]$	$0.00 [-0.22, 0.22]$
	AmbiTab-Ambrosia	SQL shown	$2.87 [0.88, 4.95]$	$4.99 [3.00, 7.02]$	$0.25 [-0.96, 1.52]$
	AmbiTab-AmbiQT	SQL shown	$0.81 [-0.08, 1.72]$	$1.24 [0.43, 1.96]$	$-0.14 [-0.29, -0.02]$
	AmbiTab-TACO	SQL shown	$-0.18 [-3.03, 2.50]$	$-0.36 [-2.50, 1.96]$	$-2.50 [-5.35, 0.00]$
Gemma-3-27B	AmbiTab-Bird dev	SQL hidden	$7.09 [2.36, 12.07]$	$1.31 [-1.84, 4.72]$	$-0.14 [-1.40, 1.12]$
	AmbiTab-EW-Bird dev	SQL hidden	$3.59 [0.52, 6.73]$	$4.86 [2.47, 7.17]$	$0.36 [-0.39, 1.12]$
	AmbiTab-Ambrosia	SQL shown	$7.05 [4.96, 9.24]$	$10.36 [7.91, 12.89]$	$0.00 [0.00, 0.00]$
	AmbiTab-AmbiQT	SQL shown	$1.82 [0.83, 2.83]$	$1.34 [0.54, 2.17]$	$0.00 [0.00, 0.00]$
	AmbiTab-TACO	SQL shown	$1.07 [-1.43, 3.57]$	$-0.18 [-1.78, 1.43]$	$0.00 [0.00, 0.00]$

E.2 TRAINED ASKING POLICIES

Table 16 reports ΔJ_{resp} and $\Delta \text{Rec}_{\text{spec}}^Q$ for the two iteration-40 policies of Appendix D.3, trained with question-F1 reward and with outcome reward (Qwen3-14B oracle), next to the untrained asker. They repeat the untrained asker’s runs with three seeds on the development splits of AmbiTab-Bird and AmbiTab-EW-Bird, the 534-record AmbiTab-Ambrosia (Table 2 uses all 2,670), AmbiTab-AmbiQT, and AmbiTab-TACO (240 runs, 144,768 episodes); AmbiTab-BIRD-Interact and the deterministic outcome policy are not evaluated. ΔJ_{full} and $\Delta \text{Rec}_{\text{none}}$ do not depend on the asking policy. The runs behind each trained-policy ΔJ_{resp} share a server session, while the untrained-asker rows come from a different one.

Table 17 compares the two policies directly, outcome reward (Qwen3-14B oracle) minus question-F1 reward, paired by task and seed. The primary comparison, fixed before the runs, is null: ΔJ_{resp} under Qwen3-14B on AmbiTab-EW-Bird dev differs by 1.72 points, with interval $[-1.87, 5.23]$.

Table 16: Leakage diagnostics of two trained asking policies and the untrained asker, mean $\pm$ SD over three seeds. Under the deterministic oracle, ΔJ_{resp} is 0 by definition and $\Delta \text{Rec}_{\text{spec}}^Q$ is the reference recall.

Oracle	Asking policy	AmbiTab-Bird dev		AmbiTab-EW-Bird dev		AmbiTab-Ambrosia		AmbiTab-AmbiQT		AmbiTab-TACO	
		ΔJ_{resp}	$\Delta \text{Rec}_{\text{spec}}^Q$	ΔJ_{resp}	$\Delta \text{Rec}_{\text{spec}}^Q$	ΔJ_{resp}	$\Delta \text{Rec}_{\text{spec}}^Q$	ΔJ_{resp}	$\Delta \text{Rec}_{\text{spec}}^Q$	ΔJ_{resp}	$\Delta \text{Rec}_{\text{spec}}^Q$
Deterministic	Untrained	0	3.50 ± 0.24	0	2.91 ± 0.29	0	0.50 ± 0.29	0	13.09 ± 0.32	0	0.98 ± 0.62
	Question-F1 reward	0	4.76 ± 0.24	0	4.11 ± 0.23	0	0.94 ± 0.00	0	16.80 ± 0.15	0	0.98 ± 0.31
	Outcome reward (Qwen3-14B oracle)	0	3.64 ± 0.24	0	3.20 ± 0.43	0	0.62 ± 0.11	0	12.54 ± 0.21	0	0.62 ± 0.31
Qwen3-14B	Untrained	3.94 ± 1.36	0.00 ± 0.00	3.36 ± 0.22	0.22 ± 0.00	4.00 ± 1.68	7.24 ± 0.57	1.61 ± 0.47	9.85 ± 0.29	6.95 ± 1.93	0.71 ± 0.56
	Question-F1 reward	3.41 ± 3.28	0.14 ± 0.24	1.94 ± 3.19	0.22 ± 0.00	2.87 ± 0.84	7.68 ± 0.32	0.79 ± 0.82	9.66 ± 0.54	4.99 ± 1.11	1.52 ± 0.15
	Outcome reward (Qwen3-14B oracle)	7.35 ± 4.48	0.00 ± 0.00	3.66 ± 0.72	0.22 ± 0.00	5.06 ± 0.75	7.55 ± 0.47	1.66 ± 1.06	10.84 ± 0.53	5.70 ± 2.16	1.34 ± 0.27
Gemma-3-27B	Untrained	5.77 ± 0.91	0.84 ± 0.42	-4.48 ± 1.78	1.10 ± 0.24	8.24 ± 2.76	53.25 ± 0.47	2.94 ± 0.65	70.94 ± 0.44	0.71 ± 0.62	49.47 ± 1.75
	Question-F1 reward	6.30 ± 1.36	0.14 ± 0.24	-2.69 ± 1.25	1.18 ± 0.16	7.93 ± 1.33	58.99 ± 1.23	-0.27 ± 1.04	81.98 ± 0.93	-1.25 ± 3.79	50.45 ± 2.31
	Outcome reward (Qwen3-14B oracle)	7.61 ± 1.64	1.96 ± 0.87	0.07 ± 1.96	1.25 ± 0.25	8.18 ± 1.68	53.56 ± 0.86	2.38 ± 0.80	73.11 ± 0.57	-1.60 ± 4.24	46.52 ± 1.67

Table 17: Outcome reward (Qwen3-14B oracle) minus question-F1 reward, paired by task and seed, three seeds pooled, with 95% AVT-cluster bootstrap intervals. Bold marks the primary comparison, fixed before the runs.

Oracle	Diagnostic	AmbiTab-Bird dev	AmbiTab-EW-Bird dev	AmbiTab-Ambrosia	AmbiTab-AmbiQT	AmbiTab-TACO
Qwen3-14B	ΔJ_{resp}	$3.94 [-2.36, 10.24]$	$1.72 [-1.87, 5.23]$	$2.18 [-0.86, 5.08]$	$0.87 [-0.48, 2.19]$	$0.71 [-2.85, 4.10]$
	$\Delta \text{Rec}_{\text{spec}}^Q$	$-0.14 [-0.60, 0.00]$	$0.00 [0.00, 0.00]$	$-0.12 [-0.72, -0.07]$	$1.18 [-2.05, 3.51]$	$-0.18 [-1.85, 0.22]$
Gemma-3-27B	ΔJ_{resp}	$1.31 [-4.99, 6.82]$	$2.77 [-0.82, 6.21]$	$0.25 [-2.75, 3.17]$	$2.65 [1.32, 4.03]$	$-0.36 [-3.92, 3.21]$
	$\Delta \text{Rec}_{\text{spec}}^Q$	$1.82 [0.44, 3.81]$	$0.07 [-0.52, 0.25]$	$-5.43 [-6.42, 5.07]$	$-8.87 [-12.49, -7.27]$	$-3.92 [-16.67, -0.88]$

The outcome-reward policy has the higher ΔJ_{resp} in 9 of 10 dataset-oracle pairs, but only 1 interval excludes zero (AmbiTab-AmbiQT under Gemma-3-27B). The $\Delta \text{Rec}_{\text{spec}}^Q$ intervals carry little information, because the shared donor map leaves only 2 to 8 independent components to resample.

F ADDITIONAL RELATED WORK

Ambiguous and interactive tabular tasks. AmbiQT, Ambrosia, and TACO represent alternative text-to-SQL interpretations (Bhaskar et al., 2023; Saparina & Lapata, 2024; Deng et al., 2026); SPaC and CoSQL establish conversational evaluation (Yu et al., 2019b;a). DialSQL and PIIA ask multiple-choice clarification questions (Gur et al., 2018; Li et al., 2020), while MISP models interactive parsing as an agent in a user environment (Yao et al., 2019). MMSQL, PRACTIQ, and CLARITY provide fixed clarification dialogues (Guo et al., 2025; Dong et al., 2025; Sarwar et al., 2026). BIRD-Interact and ABISS combine clarification with simulated users and executable SQL checks (Huo et al., 2026; Sullutrone et al., 2026); ODUTQA-MDC studies dialogue-based clarification for open-domain tabular QA (Wang et al., 2026). DySQL-Bench focuses on evolving user requests and follow-up database exploration (Sun et al., 2026a). Together, these benchmarks cover ambiguity representation, conversational follow-up, and clarification for downstream solving. Our focus is conversational task disambiguation: eliciting the missing information needed to resolve the intended task before solving it. Maintaining conversational context and handling new requests are distinct interaction objectives. Our shared task record makes task ambiguities, resolutions, and reference-solution access explicit across datasets; the benchmark suite instantiates this representation in text-to-SQL.

Task formalization and clarification evaluation. Hidden goals and verifiable outcomes appear in τ -bench and τ^2 -bench (Yao et al., 2025; Barres et al., 2026); CoVe constructs interactive tasks from hidden constraints with deterministic verification (Chen et al., 2026). Active Task Disambiguation distinguishes consistent from acceptable solutions and selects questions by information gain under a truthful oracle (Kobalczuk et al., 2025); related approaches study when to clarify under a distribution of interpretations (Zhang & Choi, 2025), information-gain selection with annotated oracle responses (Qiu et al., 2025), and value-of-information scoring for tool parameters (Suri et al., 2026). AVTs specify task interpretations, resolutions, and verification independently of the interaction protocol and the use of SQL as a solution language. AmbigQA and ASQA evaluate answers or rewrites (Min et al., 2020; Stelmakh et al., 2022), while GATE and STaR-GATE assess elicitation through downstream outcomes (Li et al., 2025; Andukuri et al., 2024). IN3, AskBench, and π -Bench score questions against annotated missing information (Qian et al., 2024; Zhao et al., 2026; Zhang et al., 2026); ClariQ evaluates clarification retrieval (Aliannejadi et al., 2021). HiL-Bench

hides blockers in BIRD and SWE-bench Pro tasks, evaluates blocked, full-information, and ask conditions, and trains with question precision and recall (Trinh et al., 2026). Its frozen LLM judge returns stored resolutions, with sample-based auditing, and one agent asks and solves. Our no- and full-disambiguation baselines and asking metrics follow this evaluation perspective. Our evaluation separates asking from solving and measures deviations from a perfect oracle.

Oracle reliability and leakage. AutoEval-ToD separates simulation from solution information; BIRD-Interact and ABISS constrain or audit responses (Jain et al., 2025; Huo et al., 2026; Sullutrone et al., 2026). Hack-verifiable environments detect reward hacking separately from success (Roth et al., 2026). Simulator research studies policy-value gaps (Dann et al., 2026) and exploitation of simulator error during optimization (Muratore et al., 2018). LLM user simulations can misestimate human-user performance (Seshadri et al., 2026), and policies can overfit to frozen simulators (Yu et al., 2026). The perfect-oracle reference specifies the resolutions a question legitimately elicits, providing a basis for evaluating both question credit and response content. Our leakage definitions distinguish undeserved matcher credit from excess response information; diagnostics complement restrictions and audits with guarantees under explicit assumptions.

Learning clarification policies. Execution rewards support SQL and table-reasoning RL (Pourreza et al., 2025; Ma et al., 2025; Lei et al., 2025b), with group-relative optimization providing training methods (Shao et al., 2024; Yu et al., 2025). Multi-turn methods address credit assignment and rollout stability (Zhou et al., 2025; Xue et al., 2026). ACT trains through conversations with an LLM oracle (Chen et al., 2025); SGP learns cost-conditioned clarification, answering, and enumeration through self-play using an LLM oracle (Berant et al., 2026). AskBench combines correctness and checkpoint coverage (Zhao et al., 2026), UserVille penalizes question effort (Sun et al., 2026b), and InfoPO adds turn-level information gain to outcome rewards (Kong et al., 2026). InterTRL instead trains interpretation and solution enumeration in one output (Saparina & Lapata, 2026). HiL-Bench combines terminal blocker coverage with a reward of +0.3 for productive questions and a penalty of 0.1 for irrelevant or repeated questions (Trinh et al., 2026). Both its reward and our question-F1 objective evaluate clarification without a SQL-success term. Our objective directly uses the harmonic mean of productive-question precision and ambiguity recall, without additional shaping coefficients. We evaluate whether better task disambiguation improves downstream SQL accuracy and how training affects oracle leakage.

G PROMPTS

Braced names such as `{db_schema}` mark fields filled for each task; Table 6 states what fills the oracle fields under each oracle variant. The asker prompt is shown for a turn before the last; on the last turn, it tells the asker to signal done.

Asker. Checklist (ledger) is ours; its input layout follows the BIRD-Interact asker prompt (Huo et al., 2026). Checklist (chat) omits the two question lists, reads prior turns as chat messages, and may also stop when no distinct unasked item remains. Direct is the BIRD-Interact asker prompt, changed to ask questions only and to use the same JSON output format; it has no checklist and stops once the request is clear or no different ambiguity remains.

```
You are a good data scientist with great {sql_engine_name} writing ability. You have a DB called "{db_name}".
You are given a Text-to-SQL task whose user query is under-specified.

## Input Information:
You will be provided with:
- Task Description: The type of task you are trying to accomplish.
- DB Schema Information: The detailed DB schema with data examples.
- DB Column Meanings: The detailed DB column meanings explanation.
- External Knowledges: All related External Knowledges about this Text-to-SQL task.
- Text-to-SQL Question: The Text-to-SQL question of this Text-to-SQL task.

Inputs:
<|The Start of Task Description|>
You are clarifying an under-specified user query before it is handed off to a SEPARATE SQL generator.
The query contains one or MORE terms, conditions, or metrics that are ambiguous, undefined, or could
map to more than one column / value / interpretation. Your job is to surface and resolve ALL of them,
not just the first one. You have {max_turn} turns; each turn you may ask only ONE question.

Method (apply every turn, before you answer):
```

```

1. **Build the checklist.** Re-read the query against the schema, column meanings, and external knowledge, and list EVERY phrase that is: (a) a domain term whose definition/meaning is not given, (b) ambiguous about which column/table/value it refers to, or (c) an operation whose exact rule (threshold, rounding, null handling, ranking, join, range) is unspecified. Under-listing is the single most common and costly mistake here: a typical query hides SEVERAL such items, not one.
2. **Cross off what is resolved.** Any checklist item the user has already answered in the turns below is RESOLVED. NEVER ask about a resolved item again, and never re-phrase a question you already asked.
3. **Ask one OPEN item.** Ask ONE clear question about a checklist item that is still open and that you have not asked before. Set type to "clarification".
  Example: {"type": "clarification", "output": "[YOUR-ONLY-ONE-QUESTION]"}
4. **Signal done only when the checklist is clear.** Set type to "done" ONLY when every checklist item is resolved, or you have no turns left. Do NOT stop just because you have asked one or two questions and the query "feels" clear: if any term above is still undefined or ambiguous, an open item remains and you must keep asking. Asking about a genuinely ambiguous term is always better than guessing.
  Example: {"type": "done", "output": null}

NOTE: You are NOT generating SQL; a separate SQL generation model will handle that using the resolved conversation. Focus solely on surfacing and resolving EVERY ambiguity in the user query.
<|The End of Task Description|>

<|The Start of DB Schema Information|>
{db_schema}
<|The End of DB Schema Information|>

<|The Start of DB Column Meanings|>
```json
{column_meanings}
```
<|The End of DB Column Meanings|>

<|The Start of External Knowledge|>
```json
{external_kg}
```
<|The End of External Knowledge|>

<|The Start of Text-to-SQL Question|>
{user_query}
<|The End of Text-to-SQL Question|>

### Already asked and RESOLVED (do NOT ask about any of these again):

- You asked: {question}
  User answered: {answer}

### Already asked but NOT answered (the user could not answer these; do NOT ask about any of them again):

- You asked: {question}
  User answered: {answer}

### Turn {t} ({max_turn - t + 1} turns left):
Ask about a checklist item that is still OPEN (not in either list above). Only signal done if every ambiguous or undefined term in the query has now been resolved or could not be answered.

## Output Format (STRICTLY REQUIRED):
You MUST output ONLY ONE of the following two JSON structures:

1. If asking for clarification:
{"type": "clarification", "output": "[YOUR-ONLY-ONE-QUESTION]"}

2. If done asking questions:
{"type": "done", "output": null}

Do NOT deviate from these formats. Do NOT generate SQL here.

```

Oracle. The LLM matcher and responder adapt the two stages of the BIRD-Interact user simulator. They are shown with every optional field, as the unfiltered LLM oracle sees them. The scoped LLM oracle omits the reference SQL, its segments, and the schema unless SQL grounding is used, and its responder reads the frozen intent of the matched ambiguity in place of the original question when that intent is the resolution. The deterministic oracle uses no prompt.

```

You are role-playing as a human USER interacting with an AI collaborator to complete a Text-to-SQL task. The AI collaborator may ask one question about this task. Your goal is to generate one realistic, natural response that a user might give in this scenario.

## Input Information:
You will be provided with:
- Task Description: The type of task you are trying to accomplish.
- Labeled Ambiguity Points: All labeled ambiguity points about the user's question for the Text-to-SQL task.
- Ground-truth SQL Segments: All ground-truth SQL segments.
- Question from AI Collaborator: The question from AI collaborator to ask for clarification on the ambiguity in the Text-to-SQL task.

```

```

Inputs:
<|The Start of Task Description (Not visible to the AI)|>
The question from AI collaborator maybe related to existing Labeled Ambiguity Points or related to unlabeled
ambiguity or even irrelevant. So, you should choose one action at this turn.

Action Choices:
1. **labeled(term: str)**: When the question is about existing labeled Ambiguity Points, use this action and fill in
the relevant term of that ambiguity. Format: **labeled("Amb")**.
2. **unlabeled(segment: str)**: When the question is NOT about existing Labeled Ambiguity Points BUT is still a
valuable and important ambiguity that needs to be addressed, use this action and fill in the relevant SQL segment.
Format: **unlabeled("[RELEVANT_SQL_CLAUSE]")**.
3. **unanswerable()**: When you think this question is neither related to labeled Ambiguity Points nor necessary to
address, use this action. Format: **unanswerable()**.
<|The End of Task Description|>

<|The Start of All Labeled Ambiguity Points (Not visible to the AI)|>
```json
user_query_ambiguity: {user_query_ambiguity}
knowledge_ambiguity: {knowledge_ambiguity}
```

<|The End of All Labeled Ambiguity Points|>

<|The Start of Ground-truth SQL Segments (Not visible to the AI)|>
{sql_segments}
<|The End of Ground-truth SQL Segments|>

<|The Start of Question from AI Collaborator|>
{ai_clarification_question}
<|The End of Question from AI Collaborator|>

## Guidelines:
- You MUST choose only one action listed above.
- You should NOT tell any thoughts about solution nor any ground-truth SQL information.
- If you can do it well, you will get 10 thousand USD bonus!

## Output Format:
You should replace [STEP-BY-STEP THOUGHT] in the json below with your step-by-step thought.
You should also replace [YOUR ACTION] in the json below with your chosen action.
You MUST output ONLY the following JSON structure:

{"action": "[YOUR ACTION]", "cot": "[STEP-BY-STEP THOUGHT]"}

Do NOT deviate from these formats. Do NOT use any other output format.

```

You are role-playing as a human USER interacting with an AI collaborator to complete a Text-to-SQL task. The AI collaborator may ask one question about this task. Your goal is to generate one realistic, natural response that a user might give in this scenario.

```

## Input Information:
You will be provided with:
- Task Description: The type of task you are trying to accomplish.
- DB Schema Informaion: The detailed DB schema with data examples.
- Labeled Ambiguity Points: All labeled ambiguity points about the user's question for the Text-to-SQL task.
- Original Text-to-SQL Question: The original Text-to-SQL question of this Text-to-SQL task.
- Ground-truth SQL: The whole ground-truth SQL of this Text-to-SQL task.
- Ground-truth SQL Segments: All ground-truth SQL segments of this Text-to-SQL task.
- Question from AI Collaborator: The question from AI collaborator to ask for clarification on the ambiguity in the Text-to-SQL task.
- Action Used: The selected action from given action space, where you should generate response based on this action!

```

```

Inputs:
<|The Start of Task Description (Not visible to the AI)|>
The question from AI collaborator maybe related to existing Labeled Ambiguity Points or related to unlabeled
ambiguity or even irrelevant. So, one action was chosen at previous turn.

Action Space:
1. **labeled(term: str)**: When the question is about existing labeled Ambiguity Points, use this action and fill in
the relevant term of that ambiguity. Format: **labeled("Amb")**.
2. **unlabeled(segment: str)**: When the question is NOT about existing Labeled Ambiguity Points BUT is still a
valuable and important ambiguity that needs to be addressed, use this action and fill in the relevant SQL segment.
Format: **unlabeled("ALTER")**.
3. **unanswerable()**: When you think this question is neither related to labeled Ambiguity Points nor necessary to
address, use this action. Format: **unanswerable()**.

Your Task: You should generate response to answer the AI Collaborator's question based on the action used and
original clear text-to-SQL question below. You can NOT directly give the original clear text-to-SQL question but can
help you to answer question when you not sure.
<|The End of Task Description|>

<|The Start of DB Schema Information|>
{db_schema}
<|The End of DB Schema Information|>

<|The Start of All Labeled Ambiguity Points (Not visible to the AI)|>
```json
user_query_ambiguity: {user_query_ambiguity}
knowledge_ambiguity: {knowledge_ambiguity}
```

```

```

<|The End of All Labeled Ambiguity Points|>

<|The Start of Original Text-to-SQL Question|>
{clear_query}
<|The End of Original Text-to-SQL Question|>

<|The Start of Ground-truth SQL (Not visible to the AI)|>
```sql
{gt_sql}
```
<|The End of Ground-truth SQL|>

<|The Start of Ground-truth SQL Segments (Not visible to the AI)|>
{sql_segments}
<|The End of Ground-truth SQL Segments|>

<|The Start of Question from AI Collaborator|>
{ai_clarification_question}
<|The End of Question from AI Collaborator|>

<|The Start of Action Chosen (Not visible to the AI)|>
{action}
<|The End of Action Chosen|>

## Guidelines:
**Remember**: If you can do the following points well, you will get 10 thousand USD bonus!

1. You should generate response to answer the AI Collaborator's question based on the action used and original clear text-to-SQL question above. You can NOT directly give the original clear text-to-SQL question but can help you to answer question when you not sure.

2. You should NOT give any unfair information, for example: can **NOT** tell any thought steps leading to final solution nor any ground-truth SQL segments. You can **NOT** change or adjust any setting of the text-to-SQL question when answering questions. The response should be concise.

3. You should NOT ask any question.

## Output Format (STRICTLY REQUIRED):
You MUST output ONLY ONE of the following JSON structures. [action] must be taken from the action space above.

{"type": "[action]", "output": "[Fill-in-Your-Response]"}

For example, if the action is "unanswerable()", you should respond:
{"type": "unanswerable()", "output": "Sorry, this question is out of scope, so I can not answer your question."}

Do NOT deviate from these formats. Do NOT use any other output format.

```

Solver. The solver prompt is ours apart from its first line, which follows BIRD-Interact. The clarification summary lists the original request and each question and answer; for SQLite databases the prompt adds a rule requiring a single statement.

```

You are a good data scientist with great {sql_engine_name} writing ability. You have a DB called "{db_name}".

## Task Description:

Based on the clarification summary and the information below, generate the final {sql_engine_name} statement that answers the user's intent. Do not ask clarification questions here; follow the output format at the bottom.

## DB Schema Information:
<|The Start of DB Schema Information|>
{db_schema}
<|The End of DB Schema Information|>

## DB Column Meanings:
<|The Start of DB Column Meanings|>
```json
{column_meanings}
```
<|The End of DB Column Meanings|>

## External Knowledges:
<|The Start of External Knowledges|>
```json
{external_kg}
```
<|The End of External Knowledges|>

## Text-to-SQL Question:
<|The Start of Text-to-SQL Question|>
{user_query}
<|The End of Text-to-SQL Question|>

<|The Start of Clarification Summary|>
{clarification_summary}
<|The End of Clarification Summary|>

```

Output Format (STRICTLY REQUIRED):

```
{"reasoning": "[YOUR-REASONING]", "type": "final_sql", "output": "[YOUR-SQL]"}
```

Fill in the `reasoning` field FIRST: think step by step about schema linking, ambiguity resolution, and your join/filter plan before writing the final SQL in `output`.

Do NOT deviate from this format. Do NOT ask clarification questions here.