Two Surfaces of Ambiguity: Complementary Detection for Text-to-SQL

Leonhard Liu
Berliner Hochschule für Technik
Berlin, Germany
lliu@bht-berlin.de

Patrick K. Erdelt
Berliner Hochschule für Technik
Berlin, Germany
patrick.erdelt@bht-berlin.de

Abstract

Natural-language interfaces to relational databases increasingly rely on LLMs, but real-world queries are routinely ambiguous, admitting multiple valid SQL interpretations. In interactive deployment, the system should detect this ambiguity and ask for clarification rather than commit to a plausible but wrong query. Two black-box strategies are typically treated as alternatives: prompting the model to flag ambiguous terms in the question (*introspection*), or sampling candidate SQLs and using their disagreement as an ambiguity signal (*sampling-based* detection). We show that they are complementary but not disjoint on BIRD-Interact-Lite, a Text-to-SQL benchmark with 300 tasks, annotated ambiguities, and an LLM-based user simulator [8]. Introspection catches what is visible in the prompt: vague terms, missing formulas, and unclear schema references. Sampling, by contrast, exposes choices the model silently commits to during SQL generation, such as NULL handling and join type. A union of both strategies' clarification questions raises recall by at least 6 pp over either method across three open-weight models, and by 4.8 pp on one proprietary model, using only three LLM calls. In the downstream agentic setting, the same union improves normalized reward across all three open-weight models; on the non-interactive AMBROSIA benchmark, it also improves coverage of valid SQL interpretations. As a byproduct, we trace an apparent recall gap on masked-knowledge ambiguities to the single-label limitation in BIRD-Interact's user-simulator encoder. A proposed multi-label grammar substantially narrows it without affecting downstream context.

VLDB Workshop Reference Format:
Leonhard Liu and Patrick K. Erdelt. Two Surfaces of Ambiguity. VLDB 2026 Workshop: Applied AI for Database Systems and Applications (AIDB 2026).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/BHT-Math/text2sql-ambiguity-detection.

1 Introduction

Natural-language interfaces to relational databases (NLIDBs), increasingly powered by LLM-based Text-to-SQL agents, are being deployed in interactive analytics pipelines: a user issues a question in natural language, the agent translates it to executable SQL, and the database returns the answer. On well-specified queries, modern LLMs achieve high accuracy on standard Text-to-SQL benchmarks such as Spider [24] and BIRD [12]. But realistic user queries are routinely *ambiguous*. The request *"Show the efficiency rating for each device"* admits multiple valid SQL interpretations depending on which formula, threshold, or aggregation is intended. In interactive deployment the right behavior is to ask a clarification rather than guess. The open question is then: *what should a black-box LLM agent ask about, before committing to a possibly-wrong answer?*

Ambiguity in this setting is not monolithic. Some lives in the *question*: a missing formula, a vague term, or a column name that maps to two tables. Other ambiguity hides in the SQL the model will generate, such as the choice of join type, the treatment of nulls, or the handling of division by zero. BIRD-Interact [8] defines these as *intent-level* ambiguities and *implementation-level* ambiguities. Both types produce what the *uncertainty quantification* (UQ) literature calls *aleatoric* uncertainty: it is inherent to an under-specified input and is resolvable only by asking the user. This contrasts with *epistemic* uncertainty, which is what the model itself does not know [7]. Ambiguity in interactive Text-to-SQL is overwhelmingly aleatoric. For the rest of the paper we use "ambiguity" rather than "uncertainty" to mark this restricted, user-resolvable case.

For black-box ambiguity detection, prior work has explored two main families. *Introspective prompting* asks the model to enumerate ambiguous terms in the question itself [3]. *Sampling-based detection* generates multiple candidate SQLs and treats their disagreement as an uncertainty signal [2, 5, 11]. A third family, token-level log-probability confidence, is a poor fit here [14]. Many SQL strings can compute the same query, so the model's probability mass for that query is split across its many equivalent strings. A single string's log-probability therefore measures confidence in *that string*, not in the underlying meaning. The problem is compounded for code, where much of the per-token probability mass sits on syntactic scaffolding such as SELECT and FROM rather than the tokens that distinguish interpretations. Sampling-based methods sidestep this by scoring meanings instead of strings [11]. The two viable families are typically compared as alternatives, although Xiong et al. [23] find partial complementarity in their broader study. We show that for Text-to-SQL the complementarity is structural, mapped to the two ambiguity surfaces above: **introspection sees the question; sampling sees the code** (Figure 1).

Research questions. Realistic user queries in interactive Text-to-SQL can be ambiguous in what the user intends or in how that intent should be implemented in SQL. The relevant question is therefore not only *whether* to ask for clarification, but *what* to ask about and how to detect it reliably. The paper is organized around three questions:

Proceedings of the VLDB Endowment. ISSN 2150-8097.

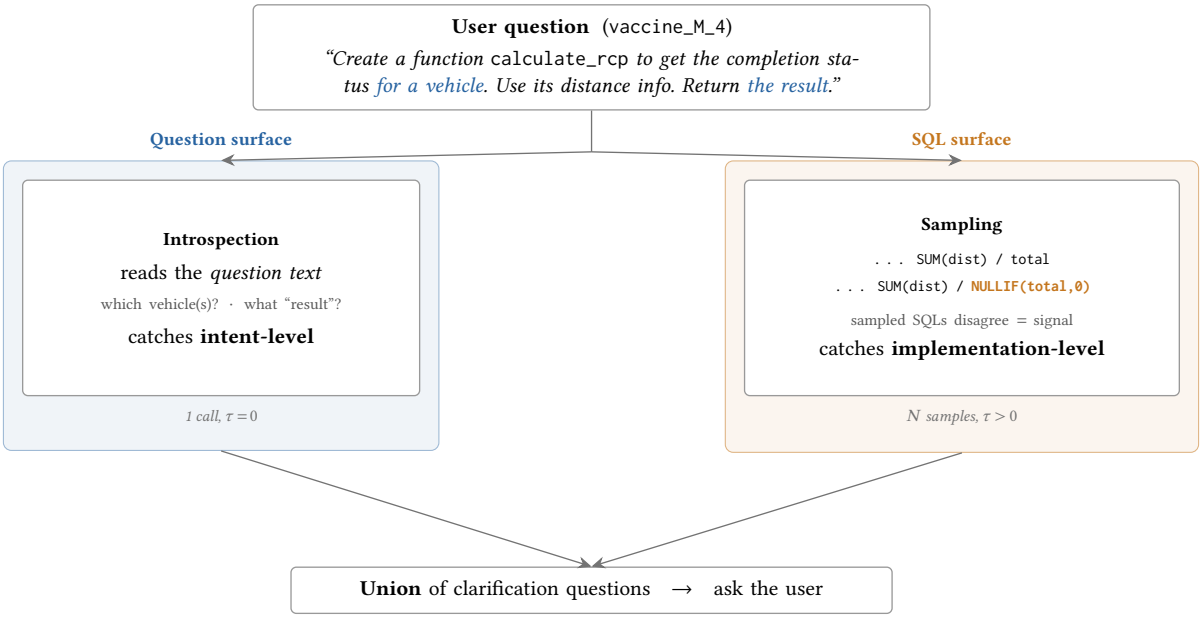

Figure 1: The two surfaces of one underspecified question, shown on `vaccine_M_4` (Table 8). Introspection reads the question text and catches the intent-level ambiguity (scope, and what "the result" is); sampling reads disagreement among generated SQLs and catches the implementation-level ambiguity (the divide-by-zero guard). The union of their questions is what the agent asks the user.

(1) **RQ1 (coverage).** Which detection method helps for which kind of ambiguity?
(2) **RQ2 (composition).** Can detection methods be combined, and is the combination strictly better than the best single method?
(3) **RQ3 (downstream effect).** How far do detection-level gains transfer to end-to-end task success, and what limits that transfer?

Methodology. We evaluate on BIRD-Interact, an interactive Text-to-SQL benchmark in which the agent interacts with a user simulator and a database. Its lite split provides 300 tasks and 1,550 ground-truth ambiguity terms across 15 typed categories at the intent, implementation, and knowledge levels. The four detection methods span two axes: candidate source (introspection on the question vs. sampling of SQLs) and generation budget (low vs. high), detailed in Section 3. For robustness across model families, results cover the open-weight GLM-4.5-Air, MiniMax-M2.5, and Qwen3.5-122B, plus the proprietary Gemini 3.1 Flash Lite. Cross-benchmark transfer is tested on AMBROSIA [16]. To measure whether detection improvements translate to task success, the agentic experiments run in BIRD-Interact's default budget-constrained *stress* mode, which scores the agent's end-to-end SQL via normalized reward (NR).

Concretely, our **contributions** are:

(1) **Per-type mechanism (RQ1).** A comparison across 15 ambiguity types and three open-weight models, the most granular such analysis we are aware of for structured-output ambiguity detection, showing that introspection reaches high but uneven recall across intent-level subtypes while sampling catches the implementation-level subtypes (`null`, `join`, `divide_zero`) that single-call introspection under-covers.
(2) **Detection-level complementarity (RQ2).** Pairing single-call introspection with a single SQL-sampling pass (three LLM calls in total) lifts aggregate detection recall by at least 6 pp over either single method on the three open-weight models and by 4.8 pp on Gemini 3.1 Flash Lite.
(3) **Downstream and cross-benchmark effects (RQ3).** On AMBROSIA, a non-interactive benchmark whose ambiguities are all intent-level, the same union improves AllFound coverage (every gold interpretation recovered) by 10.9 pp over the same-model baseline, showing that the recipe is not BIRD-Interact-specific. In BIRD-Interact's interactive agent, the union of two methods improves task success by roughly 3–7 pp NR over the released ReAct baseline.

Supporting analyses. We run two further studies, an ablation and a diagnostic. The ablation removes AST clustering. At the model scale we tested, it adds little beyond SQL sampling itself, so the useful signal lies in the diversity of the sampled SQL. The diagnostic samples the introspection prompt itself (sampled self-introspection). It recovers some implementation-level ambiguities without generating SQL, but only at 11× the LLM call budget.

Byproduct. BIRD-Interact's single-label encoder grammar undercredits masked-knowledge ambiguity, leaving a 20–30% recall floor independent of question quality. A small multi-label extension of its action grammar recovers ∼55 pp of masked-knowledge recall, without changing what the agent sees (Section 5.4).

2 Related Work

Verbalized vs. sampling-based confidence. The most direct precursor to our work is Xiong et al. [23], a broad comparison of verbalized (introspective) and sampling-based (behavioral) confidence across arithmetic, commonsense, symbolic, and professional-knowledge tasks. They find a hybrid recipe works best overall, with no single method dominating. Verbalized confidence has been studied widely [9, 13, 19]. On the sampling side, semantic entropy (SE) [5, 11] anchors our behavioral probe. **Our setting differs from this line on three axes.** *Task structure*: outputs are multi-token structured programs whose raw token-level probabilities are poorly calibrated as confidence signals [14]. *Mode of ambiguity*: prior benchmarks are dominated by epistemic uncertainty (the model either knows the fact or does not), whereas Text-to-SQL ambiguity is overwhelmingly aleatoric input under-specification. Uncertainty quantification has been shown to degrade most severely under exactly this kind of ambiguity [20]. *Granularity*: prior work reports aggregate results, not per-type detection recall. An aggregate gain from combining two methods would be expected even if both were merely noisy detectors of the same signal; the per-type breakdown is what separates that from our finding: two channels reading structurally distinct ambiguity surfaces (question-level vs. output-level).

Ambiguity detection for Text-to-SQL.. Within Text-to-SQL, black-box detection splits along the same introspective/sampling divide. The closest *introspective* baseline is AmbiSQL [3]: a hand-curated taxonomy of seven ambiguity types with in-context learning (ICL) exemplars, resolving ambiguities through iterative multiple-choice clarification over a curated set of 40 ambiguous BIRD/TAG queries. Because AmbiSQL supplies a predefined taxonomy and annotated exemplars, it operates in a more information-rich regime than our zero-shot, taxonomy-free probes, so we treat it as complementary prior work rather than a controlled baseline. On the *sampling* side, SIMBA [2] aggregates similarity over sampled SQL, and Sharma and David [18] cluster general code by symbolic-execution equivalence. Our probe clusters over the full structural AST equivalence relation, replacing the natural-language inference clustering used in the original SE, a design choice we ablate rather than assume (Section 6). **To our best knowledge, no prior work** (a) **compares introspection and sampling on the same Text-to-SQL questions**, (b) **decomposes detection by ambiguity type at scale**, (c) **ablates the structural clustering step itself against raw-SQL diversity**, or (d) **measures whether detection-level gains transfer to interactive agentic task success**. These are the four gaps our paper closes.

Benchmarks and interactive agents. Our primary testbed is BIRD-Interact, an interactive Text-to-SQL benchmark with per-term ambiguity annotations and an LLM user simulator. AMBROSIA is a non-interactive benchmark with intent-level coverage only. Recent systems optimize that setting with fine-tuning, routing, top-k decoding, or representation-based detectors [1, 6, 17]; we use it only as a cross-benchmark transfer check, not a supervised leaderboard target. A separate line of interactive Text-to-SQL work focuses on generation-time exploration and repair [4, 22]. Our work is earlier in the pipeline: it asks what should be clarified before the agent commits to a SQL interpretation.

Table 1: The four ambiguity-detection methods organized by candidate source (rows) and generation budget (columns). Each source reads a different ambiguity surface.

	Low budget	High budget
Introspection (question surface)	SI-1 (1 call)	SI-10 (11 calls)
Sampling (SQL surface)	MCS-1 (2 calls)	MCS-10 (11 calls)

3 Methods

We compare four ambiguity-detection methods with one ablation. All methods produce a numbered list of clarification questions per sample, which are individually evaluated by the BIRD-Interact user-simulator encoder. **Our evaluation measures detection only. None of these methods generate the final SQL submitted to the database environment.** A separate agentic experiment evaluates how detection translates to end-to-end task success.

The four main methods span a 2×2 grid over two axes: *candidate source*, whether the ambiguity signal is read from the question text or from generated SQL; *generation budget*, whether candidate generation uses a low or high number of LLM calls. Table 1 maps each method to this design and lists the exact call count. These methods serve as controlled black-box probes to test whether detection recall depends more on the inspected surface or on candidate budget.

Shared neutral analysis prompt. All four methods receive the same neutral analysis instruction: *"List every ambiguity you can identify."* The wording is deliberately untilted toward question-level vs. SQL-level ambiguities: in an internal control, biasing the instruction toward user intent (rather than SQL syntax) suppressed implementation-level detections substantially across the three open-weight models we test. Neutral form is used throughout.

3.1 Self-Introspection (SI-1)

SI-1 (*Self-Introspection*) asks the model to introspect on the question text itself, in the selective-clarification spirit of Kuhn et al. [10]. The model receives the user question, database schema, column meanings, and external knowledge base. Through the shared neutral analysis prompt, it is asked to identify every ambiguous term that could lead to different SQL interpretations and to produce one clarification question for each.

LLM calls: 1 analysis call. **Temperature:** 0.0. No SQL is generated.

3.2 Sampled Self-Introspection (SI-10)

When compute is plentiful, one option is to sample the introspection itself. SI-10 runs SI-1 ten times independently with different seeds at temperature $\tau = 0.7$, self-consistency [21] applied to the introspection channel rather than to final answers. Questions are generated with the same shared neutral analysis prompt as SI-1, and an additional deduplication of these questions is made via a single LLM call at $\tau = 0.0$. The final question list varies around ~ 9 to 12 questions per sample.

LLM calls: $10\times$ SI-1 + 1 LLM semantic-dedup call. **Temperature:** 0.7 for introspection, 0.0 for dedup.

Table 2: Mcs-1 single-pass reliability, each metric per call, averaged over the 300 tasks then mean ± sd across seeds 64/512/1028 **(*parseable SQLs* out of 10 requested).**

Model (τ)	parseable SQLs	clarif. Qs	% ≥1 clarif.
GLM-4.5-Air (1.0)	9.3±0.0	8.0±0.2	100
MiniMax-M2.5 (1.3)	9.5±0.1	9.0±0.1	100
Qwen3.5-122B (1.3)	9.2±0.1	5.5±0.0	98.8

3.3 Single-Pass Multi-Candidate SQL (Mcs-1)

Mcs-1 (*Single-Pass Multi-Candidate SQL*) reads the ambiguity signal from generated SQL rather than from the question text, drawing its candidate diversity from one prompt: a single high-temperature call, conditioned on the same inputs as Si-1, is instructed to *"produce 10 different SQL interpretations"* of the question. A second call at $\tau = 0$ then analyzes the candidates and emits the clarification questions.

Between generation and analysis sits a light structural scaffold: candidates are parsed with sqlglot [15], greedily clustered by weighted-Jaccard similarity (threshold θ) over structural fingerprints (tables, WHERE predicates, aggregations), and presented to the analysis call as k cluster representatives plus a summary of which structural slots differ. Section 6 ablates this scaffold against presenting the raw SQLs and finds it recall-neutral at the model scales we test; we retain it in the evaluated configuration throughout.

LLM calls: 1 multi-SQL generation call + 1 analysis call. **Temperature:** 1.0–1.3 for SQL generation, 0.0 for analysis.

Single-pass reliability. Because diversity in Mcs-1 comes from one call rather than independent samples, a higher temperature is required for SQL generation. At low temperature, the ten requested SQLs collapse to near-duplicates and yield no disagreement signal. At our chosen temperatures, across three models and three seeds (Table 2), each call returns most of the ten requested SQLs in parseable form (92–95% of generated SQLs parse) and emits at least one clarification question on essentially every task.

3.4 Multi-Pass Candidate SQL (Mcs-10)

Mcs-10 (*Multi-Pass Candidate SQL*) adapts the semantic-entropy idea of Kuhn et al. [11] to SQL by clustering sampled candidates with a deterministic AST equivalence relation. It runs the same pipeline as Mcs-1, differing only in where the diversity comes from: the 10 candidates are generated as 10 independent samples at temperature $\tau = 0.7$, rather than requested from one call.

LLM calls: 10× single SQL generation + 1 analysis. **Temperature**: 0.7 for SQL generation, 0.0 for analysis.

4 Experimental Setup

4.1 Models

Four models are used for the detection task: GLM-4.5-Air, MiniMax-M2.5, and Qwen3.5-122B (abbreviated GLM, MM, Qwen; Table 3), plus the proprietary Gemini 3.1 Flash Lite. We chose them for being recent and strong at code generation: each is capable enough for the agentic setting, yet small enough to self-host on a two-GPU node. They were served via vLLM; Gemini was accessed through Google Vertex. Each detection method has a sampling temperature tuned to its design. The analysis call uses a fixed temperature of 0.0 and a max output of 16,384 tokens.

Table 3: Open-weight models provided through HuggingFace: `zai-org/GLM-4.5-Air-FP8`, `MiniMaxAI/MiniMax-M2.5`, `Qwen/Qwen3.5-122B-A10B` (with thinking disabled). Serving: 2× Nvidia B200, `max-model-len=131072`. MoE: mixture-of-experts; params are total / active per token.

Model	Architecture	Total/Active params	Quantization
GLM-4.5-Air	MoE	106B / 12B	FP8
MiniMax-M2.5	MoE	230B / 10B	FP8
Qwen3.5-122B	MoE	122B / 10B	BF16

4.2 Why BIRD-Interact, and How We Measure

BIRD-Interact is an interactive Text-to-SQL benchmark in which the agent talks to an LLM-based *user simulator* rather than seeing a static question. The simulator has two stages that can be used independently: an **encoder**, which maps a clarification question to an annotated ambiguity when one applies, and a **decoder**, which turns the encoder's decision into the simulated user's answer. In the live interactive setting both stages run. For our detection study, we use only the encoder, which is what makes detection precision and recall computable.

Splits. All BIRD-Interact experiments, including the end-to-end agentic experiment, use BIRD-Interact-Lite (300 tasks across 18 databases, hereafter Lite). Across the Lite split, there are **1,550 ground-truth ambiguity terms** stemming from two disjoint sources (Table 4). The *user-query* set (1,285 terms, 14 subtypes) is split into intent-level and implementation-level. The *knowledge-level* set (n=265 across 233 tasks) is structural rather than lexical: BIRD-Interact ships a per-database knowledge base (KB) of domain formulas and concepts, and for these 233 tasks at least one KB entry is hidden from the agent while the user simulator retains the full KB. We call this category *masked-knowledge ambiguity* (ground-truth field `knowledge_ambiguity`) and use that term throughout.

Metric for end-to-end agentic evaluation. BIRD-Interact's agentic mode has two official metrics. **Success rate (SR)**: SR_{P_1} and SR_{P_2} are the fractions of samples that complete Phase 1 (ambiguity resolution) and Phase 2 (follow-up query) respectively. **Normalized reward (NR)** is the priority-weighted aggregate: $NR = 0.7 \cdot SR_{P_1} + 0.3 \cdot SR_{P_2}$, with Phase 2 requiring a successful Phase 1.

Recall and precision via the encoder. Each clarification question from a detection method is fed into BIRD-Interact's released single-label encoder (the v2 chain-of-thought variant shipped with the benchmark), one question at a time. The encoder returns one of three verdicts: *matched* (the question asks about a specific annotated ground-truth ambiguity term), *valid* (the question asks about a real ambiguity but one outside the annotated set), or *rejected* (neither applies). From these verdicts, **recall** = matched ground-truth terms / total ground-truth terms and **precision** = (matched + valid) questions / generated questions. To measure how two methods complement each other, their per-term union is also reported. A ground-truth term counts as found if either method asks about it. The encoder runs at τ=0 and is **cross-model**, so no model's

Table 4: BIRD-Interact-Lite ambiguity subtypes. Intent- and implementation-level subtypes relate to the user question (1,285 terms); masked-knowledge ambiguity covers KB entries hidden from the agent (265 terms). *n* counts ground-truth terms in the 300-task Lite split.

Subtype	*n*	Description
Intent-level (vague user intent)		
knowledge_linking	368	Query's link to a KB-defined concept is implicit/unclear
intent	171	Intended operation or ranking criterion underspecified (e.g. "top")
schema_linking	157	Term maps to multiple plausible schema elements
semantic	111	Grammatically correct but vague, lacking detail (e.g. "recent")
lexical	15	A token has multiple meanings/senses in context
syntactic	4	Sentence has multiple valid grammatical parses
Implementation-level (underspecified SQL choices)		
sort	130	Sort direction unspecified
decimal	115	Rounding precision unspecified
null	108	NULL handling unspecified
join	69	JOIN type unspecified
distinct	20	DISTINCT required or not
divide_zero	12	Division-by-zero handling
date_format†	3	Date format unspecified
rank	2	Ranking method unspecified
Masked-knowledge ambiguity (KB entry hidden)		
knowledge_ambiguity	265	KB row hidden from agent; user simulator retains the full KB

† `date_format` appears in the released data but not in the BIRD-Interact paper's published implementation-level taxonomy [8].

questions are judged by its own family. Qwen3.5-122B judges the clarification questions from GLM-4.5-Air, MiniMax-M2.5, and Gemini 3.1 Flash Lite. MiniMax-M2.5 judges those from Qwen3.5-122B.

Hyperparameters. The sampling temperatures were set by a 50-sample sweep and then frozen across the three open-weight models: τ=0.7 for Mcs-10 and Si-10, and τ=1.3 for Mcs-1 (1.0 for GLM-4.5-Air). The AST-similarity threshold is θ=0.70 for Mcs-10 and θ=0.50 for Mcs-1 (0.60 for GLM-4.5-Air). All analysis and deduplication calls use τ=0.0. Per-sample LLM-call costs are constant by method (Si-1: 1, Mcs-1: 2, Mcs-10: 11, Si-10: 11) plus encoder calls equal to the number of emitted clarifications (typically 4–9 for Si-1/Mcs-1/Mcs-10 and 9–12 for Si-10 after semantic deduplication of its ~67 raw questions).

5 Results

This section evaluates how well the detectors' clarification questions recover the annotated ambiguities, how the methods complement one another, and whether the resulting question pool from the union method improves downstream SQL success. The end-to-end agentic results use the user simulator's encoder and decoder in an interaction loop. Recall is reported under the cross-model encoder setup of Section 4.2 and matches an independent `claude-haiku-4-5` re-evaluation within ±2 pp on all nine open-weight cells.

5.1 Overall Detection Recall

Table 5 reports aggregate recall against the 1,550 ground-truth terms. The three single-analysis methods (Si-1, Mcs-1, Mcs-10) show no consistent winner on aggregate: they fall within a few points of one another on most models, with small cross-seed variance. Si-10 achieves the highest recall among the individual methods, but at a high call budget. Even so, its lead does not surpass the three-call Si-1 ∪ Mcs-1 union. We therefore treat it as a diagnostic (Section 6), not a deployment option.

5.2 Per-Type Recall

The aggregate numbers mask a sharp per-type divergence. Tables 6 and 7 break recall down by ambiguity subtype, revealing a clear split between intent-level and implementation-level cases.

With the exception of knowledge_linking, self-introspection leads the intent subtypes (Table 6). These ambiguities are usually visible from the question and database context alone, including vague terms, missing formulas, and unclear schema references.

Sampling-based methods show the opposite strength on the implementation level. On `null` and `join`, they substantially outperform introspection, often by roughly a factor of two (Table 7). These ambiguities are choices the model makes implicitly while writing SQL and become visible only when sampled SQLs disagree. Mcs-1 and Mcs-10 are broadly similar here, despite their very different call budgets. `decimal` sits near the floor for every method, and not for lack of signal. Across three sampling seeds, the sampled SQLs visibly disagree on rounding in 44% of decimal-annotated tasks, yet 87% of the misses on those tasks emit no rounding-related question at all. The disagreement reaches the analysis call and never becomes a question.

Two worked examples. Table 8 grounds the claim in two GLM-4.5-Air outputs. In `vaccine_M_4`, only the sampled SQLs expose a divide-by-zero choice that Si-1 never names. In `museum_4`, only the sampled SQLs reveal a join-type choice (LEFT vs INNER when related rows may be missing), while Si-1 catches the schema-linking phrase. Neither method is wrong, they read different surfaces of the same input. The complementarity is robust but not exclusive: each surface also recovers some of what the other reaches.

5.3 Complementarity: Pairing the Question and SQL Surfaces

For a combined deployment, we focus on Si-1 ∪ Mcs-1, which simply pools the clarification questions from both methods. It costs only 3 LLM calls, one for Si-1 and two for Mcs-1. This combines the question surface with the SQL surface without paying the ten-call sampling overhead of Mcs-10.

The Si-1 ∪ Mcs-1 union beats either method on every open-weight model, with a smaller lift on Gemini. This gain follows

Table 5: Aggregate recall (%) against 1,550 ground-truth terms (n=300), single-label, cross-model encoder. Token columns are the mean per sample across the four models, billed without prefix caching. SI-1 is single-seed, other rows are 3-seed mean ± std. SI-1 ∪ MCS-1 is the per-term union (∨) of SI-1 and MCS-1. Bold marks the best of the three single-analysis methods (SI-1, MCS-1, MCS-10) per model, including ties within 1 pp. Rows below the second rule are diagnostics and ablations, not deployment options.

		Tokens (k)		Recall (%)					
Method	LLM calls	In	Out	GLM-4.5-Air	MiniMax-M2.5	Qwen3.5-122B	Gemini 3.1 FL*	Q/samp	Prec
SI-1	1	9.7	0.3	**61.5**	62.8	**57.4**	**56.6**	6–8	91–98
MCS-1	2	10.7	2.2	59.8 ± 1.1	**66.0** ± 0.9	52.6 ± 1.2	47.2	6–9	90–96
MCS-10	11	98.8	2.5	**61.7** ± 0.7	64.1 ± 1.1	52.6 ± 0.9	47.1	4–6	89–95
SI-1 ∪ MCS-1	3	20.3	2.4	70.1 ± 0.5	72.4 ± 0.4	66.1 ± 0.4	61.4	12–17	90–97
SI-10	11	97.1	2.4	67.0 ± 0.5	69.5 ± 0.4	63.3 ± 0.4	—	9–12	87–95
MCS-1 (no AST)†	2	10.7	2.2	59.1 ± 0.9	66.4 ± 0.5	53.1 ± 0.6	50.1	6–10	90–97
MCS-10 (no AST)†	11	98.8	2.5	62.6 ± 1.1	66.1 ± 1.1	51.7 ± 1.2	46.7	6–9	91–96

* Gemini is single-seed, SI-10 unmeasured for Gemini.

† no AST: the analysis call sees raw SQL.

Table 6: Per-subtype recall (%) on intent-level ambiguities; SI-1 is single-seed, other rows are 3-seed mean. Cells: GLM-4.5-Air / MiniMax-M2.5 / Qwen3.5-122B / Gemini 3.1 Flash Lite. Bold marks the best of SI-1, MCS-1, MCS-10 for each model within a subtype; ties within 1 pp are all bolded. Subtypes with small n (≤20) are indicative only. SI-10 is reference only. SI-1 ∪ MCS-1 is recommended for deployment. Subtypes with n<5 are omitted.

Subtype	n	SI-1	MCS-1	MCS-10	SI-10	SI-1 ∪ MCS-1
knowledge_linking	368	90 / 90 / **86 / 91**	**93 / 95 / 86** / 82	89 / 91 / 80 / 80	95 / 97 / 90 / —	97 / 98 / 95 / 95
intent	171	**94 / 95 / 86 / 86**	82 / 89 / 74 / 70	88 / 88 / 76 / 78	95 / 97 / 92 / —	96 / 97 / 92 / 92
schema_linking	157	**87 / 91 / 80 / 81**	70 / 86 / 56 / 53	73 / 83 / 66 / 51	93 / 97 / 87 / —	96 / 97 / 90 / 87
semantic	111	**90 / 93 / 88 / 87**	84 / 90 / 80 / 73	87 / 86 / 81 / 67	91 / 97 / 92 / —	94 / 96 / 95 / 90
lexical	15	**67 / 93 / 80 / 80**	64 / 73 / 49 / 67	64 / 62 / 49 / 67	82 / 89 / 73 / —	82 / 93 / 84 / 93

Table 7: Per-subtype recall (%) on implementation-level ambiguities, same convention as Table 6.

Subtype	n	SI-1	MCS-1	MCS-10	SI-10	SI-1 ∪ MCS-1
null	108	21 / 17 / 23 / 12	35 / **38 / 28 / 14**	**46 / 38 / 29 / 13**	33 / 27 / 42 / —	41 / 40 / 38 / 18
join	69	30 / 38 / 20 / 25	51 / **86** / 40 / **39**	**70** / 79 / **44** / 35	43 / 51 / 32 / —	62 / 89 / 48 / 49
sort	130	**68 / 63 / 58 / 37**	49 / 53 / 43 / 28	50 / 51 / 44 / 32	74 / 79 / 69 / —	75 / 73 / 61 / 40
decimal	115	5 / 7 / **10** / **4**	6 / 15 / 4 / 2	**8 / 18 / 10 / 3**	17 / 17 / 18 / —	9 / 19 / 12 / 6
distinct	20	15 / 15 / **10 / 25**	17 / 22 / **10** / 5	**23 / 28** / 7 / 15	35 / 32 / 27 / —	25 / 25 / 13 / 25
divide_zero	12	17 / 8 / 8 / 8	19 / 11 / 11 / 8	**31 / 19 / 17 / 17**	14 / 14 / 14 / —	28 / 11 / 11 / 8

the per-type split: sampling adds implementation-level coverage, especially for `null` and `join`, while introspection supplies most intent-level subtypes that sampling under-covers.

Proprietary-model replication. Re-running the detectors on Gemini 3.1 Flash Lite (same split, prompts, and encoder mechanism; single seed, detection only) reproduces the union result in the same direction. The mechanism is narrower here: SI-1 still leads the intent-level subtypes, but sampling's implementation-level advantage concentrates in `join` rather than spreading across subtypes. We read Gemini as a directional replication, with the broader per-subtype claim resting on the open-weight models and AMBROSIA.

5.4 Masked-Knowledge Recall Ceiling

The low masked-knowledge recall in Table 9 is mainly a scoring limitation of BIRD-Interact's released single-label encoder, not a failure to ask useful questions. When a KB entry is hidden, the benchmark often represents the same missing concept twice: once as the question-side phrase the agent can see (`knowledge_linking`, e.g. *"property leverage"*), and once as the canonical knowledge entry hidden from the agent (`knowledge_ambiguity`, e.g. *"Loan-to-Value Ratio (LTV)"*). A single clarification can resolve both, but the released encoder credits only one label per question, typically the question-side phrase, leaving the masked KB label uncredited.

To understand this recall ceiling, the encoder action was temporarily extended from a single-label action to one with a primary

Table 8: Two BIRD-Interact-Lite samples (GLM-4.5-Air detection outputs) where the two surfaces produce different clarification questions for the same NL query. Each example shows the verbatim SI-1 question (intent surface) and the verbatim MCS-1 question (SQL surface). The last row of each example notes which ground-truth ambiguity term each method caught.

Example 1: `vaccine_M_4` (DB: `vaccine`). NL: "*Create a function calculate_rcp to get the completion status for a vehicle. Use its distance info. Return the result.*" The full annotated query specifies returning NULL if total distance is zero, but the visible NL text omits this.

SI-1 asks:	"*What format should the result be returned in? A percentage value (0–100), a decimal (0–1), a string description, or a more complex data structure?*"
MCS-1 asks:	"*How should the function handle edge cases such as zero total distance, zero speed, or null values?*"
Ground-truth term caught:	SI-1 catches `result` (`intent_ambiguity`, critical); MCS-1 catches `division by zero handling` (`divide_zero_ambiguity`, non-critical).

Example 2: `museum_4` (DB: `museum`). NL: "*…For each dynasty, show the dynasty name, their total priority score, the count of artifacts with adequate funding, the count with insufficient funding, the budget efficiency value, and a budget status that indicates 'Budget Crisis'…*"

SI-1 asks:	"*Which specific priority metric should be used for calculating the dynasty's total priority score?*"
MCS-1 asks:	"*How should records be joined when there might be missing data in related tables (e.g., LEFT JOIN vs INNER JOIN)?*"
Ground-truth term caught:	SI-1 catches `priority score` (`schema_linking_ambiguity`, critical); MCS-1 catches `join` (`join_ambiguity`, non-critical).

Table 9: Recall (%) by ground-truth category, single-label encoder. The multi-label-encoder value is shown in parentheses only for the masked-knowledge row, where it is material. Cells: GLM-4.5-Air / MiniMax-M2.5 / Qwen3.5-122B; *n* per category in the first column.

Category	Method	GLM	MM	Qwen
Intent (826)	SI-1	90.0	91.9	85.2
	MCS-1	84.6	91.0	76.2
	MCS-10	85.0	87.9	76.1
	SI-10	93.7	97.1	89.8
Implem. (459)	SI-1	31.6	30.3	28.3
	MCS-1	32.7	42.1	26.6
	MCS-10	39.6	41.8	29.3
	SI-10	41.8	42.4	40.7
Masked-kn. (265)	SI-1	24.5 (77.0)	28.3 (83.0)	21.1 (72.1)
	MCS-1	29.7 (86.2)	29.3 (86.4)	24.0 (76.8)
	MCS-10	27.4 (84.4)	28.7 (84.3)	19.9 (77.7)
	SI-10	27.4	30.2	19.7

The multi-label encoder also lifts Intent by 0.3–7.8 and Implementation by 1.1–4.4 pp. SI-10 masked-knowledge cells use the single-label encoder only.

label and optional secondary labels, allowing one clarification to credit paired labels when they refer to the same hidden concept. This raises masked-knowledge recall from the 20–30% band to 72–86% (Table 9). The official single-label encoder is kept for all primary results to preserve comparability with prior work. The multi-label run is a diagnostic that isolates a scoring limitation in BIRD-Interact's encoder. The decoder still sees only the primary label, so released harness behavior is unchanged.

5.5 Downstream and Cross-Benchmark Effects

RQ3 asks whether better ambiguity detection improves downstream SQL success. This is tested in two settings: the budget-constrained agentic interactive mode of BIRD-Interact, and the non-interactive coverage benchmark AMBROSIA.

BIRD-Interact agentic interactive mode. The benchmark is run in its default *stress* regime, where the agent has only a small clarification budget. The detection scaffold runs once before interaction and injects candidate clarification questions (the hint pool) into the agent context (Figure 2). For the union method SI-1 ∪ MCS-1 the hint pool is semantically deduped at the analysis step of MCS-1, resulting in around ten questions per sample. All methods share the same agent template and differ only in the source of this pool. The user-simulator decoder is held fixed on Qwen3.5-122B and the scoring-relevant encoder on Gemini 3.1 Flash Lite.

Table 10 extends the end-to-end evaluation on our open-weight models. Our proposed union method SI-1 ∪ MCS-1 improves NR over the ReAct baseline on all three. The lift is significant on GLM-4.5-Air and MiniMax-M2.5 (paired McNemar on Phase-1, $p < 0.01$) and positive but not significant on Qwen3.5-122B. Notably, SI-1 alone essentially matches the union's downstream performance ($p \geq 0.76$); Section 6 explores why the union's broader detection coverage does not yield a higher reward here.

AMBROSIA transfer. For cross-benchmark robustness, our proposed methods are evaluated on AMBROSIA's set of 1,149 ambiguous questions. AMBROSIA is a non-interactive benchmark with multiple gold SQL interpretations per question. Without a user simulator to confirm candidate clarifications, SI-1 instead enumerates possible interpretations verbatim and writes one SQL for each. For the union methods, these interpreted SQLs are combined with the SQLs generated by MCS-1 or MCS-10. All generated SQLs are executed against the database and one representative SQL per distinct execution result is kept as the final set. Against our same-model baseline, SI-1 improves AllFound by 4.7 pp, and SI-1 ∪ MCS-1 improves it by 10.9 pp (Table 11), at a cost to precision. Since AMBROSIA contains only intent-level ambiguity, this is not a test of question-vs.-SQL complementarity, but it shows that the black-box union recipe transfers beyond BIRD-Interact.

6 Analysis

The complementarity established in Section 5 reflects what each method reads. Introspection reads the model's *explicit* account of

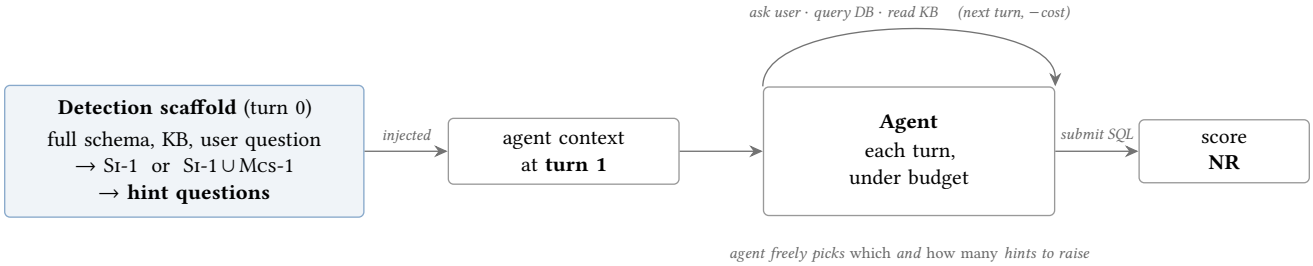

Figure 2: End-to-end pipeline in BIRD-Interact's interactive mode. Hints are injected into agent context at turn 1.

Table 10: End-to-end agentic results on BIRD-Interact-Lite (n=300) across three open-weight models. Methods differ only in the clarification-pool wired into the agent; "ReAct baseline" is the unmodified BIRD-Interact runner. Stress mode: user patience 3 (a 6-coin budget at 2 coins/ask), BIRD-Interact's default.

Model	Method	SR_{P_1}	SR_{P_2}	NR	Avg. turns	ΔNR
GLM-4.5-Air	ReAct baseline	16.3	5.3	13.0	20.5	—
	SI-1 only	25.7	8.0	20.4	21.8	+7.3
	SI-1 ∪ MCS-1	25.3	8.3	20.2	22.1	+7.2
MiniMax-M2.5	ReAct baseline	22.7	6.7	17.9	17.3	—
	SI-1 only	30.3	11.3	24.6	20.7	+6.7
	SI-1 ∪ MCS-1	31.3	11.0	25.2	20.8	+7.3
Qwen3.5-122B	ReAct baseline	28.3	15.0	24.3	16.0	—
	SI-1 only	31.3	15.0	26.4	19.7	+2.1
	SI-1 ∪ MCS-1	32.3	15.0	27.1	19.6	+2.8

Table 11: AMBROSIA cross-benchmark transfer (n=1,149 ambiguous questions). AllFound and the SI-1/Union pipeline are defined in the text. Llama3-70B figures are as reported by Saparina and Lapata [16]. Bold marks the best value per column among our rows.

Model	Method	AllFound	Precision	Avg. unique SQLs
Llama3-70B	Baseline (Prompt)	1.9	42.7	—
	Beam (top-5)	1.4	—	5
Qwen3.5-122B (ours)	Baseline	2.5	**45.2**	2.0
	SI-1	7.2	44.5	2.3
	Union (SI-1 ∪ MCS-10)	9.4	38.0	3.4
	Union (SI-1 ∪ MCS-1)	**13.4**	20.9	6.9

the question, sampling its *generative* behavior: the same underspecified input, observed on two surfaces. The taxonomy names where an ambiguity originates; our result concerns which observation channel exposes it, and at what budget. The mapping is strong but not definitional: introspection leads `sort`, an implementation-level subtype, on every model, and sampling leads `knowledge_linking`, an intent-level subtype, on two (Tables 6 and 7). The remainder of this section presents three follow-up analyses: an AST-clustering ablation, the reach and limits of SI-10, and how far detection-level gains reach end-to-end task success.

AST clustering adds no consistent signal. For the ablation, the ten sampled SQLs are held fixed, and only their presentation to the analysis call varies: raw SQLs versus cluster representatives with a difference summary. Removing the AST scaffold has no consistent effect on recall. Across the eight no-AST cells in Table 5, deltas are small, unsigned, and within each row's seed-to-seed noise. The analysis model recovers the structural disagreement directly from the raw samples, so the value of the sampling method lies in the diversity of the SQL samples, not in the clustering on top. The scaffold was designed to compress the sampled SQLs and help the model detect ambiguity. At the model scales we tested, however, its effect is recall-neutral at best. Whether AST clustering with a difference summary helps weaker models in their reasoning remains open.

The signal is also present in sampled introspection. SI-10 runs the single introspection call ten times at higher temperature and unions the deduplicated questions. This makes it a sampled superset of SI-1. Topping the individual methods on recall (Table 5) therefore follows

by construction. Sampling the question channel alone recovers much of the implementation-level coverage that one greedy call misses. The ambiguity was latent in the introspection distribution all along, and a single call just reports its most salient reading. What the extra samples cannot reach is more telling: `null` and `join` remain the exception even at higher-temperature sampling. Here, repeated introspection trails SQL sampling on `join` across every model and on `null` for two of three (Table 7). The model settles these two choices only through SQL generation, so they stay underrepresented on the question surface. The SQL surface reveals them more directly. Despite its far larger budget, SI-10 still trails the three-call SI-1 ∪ MCS-1 union on both recall and precision. It therefore remains a diagnostic, not a deployment option.

The question surface drives the scored downstream gains. Detection gains generally translate into task success, but under BIRD-Interact's scoring the lift comes almost entirely from introspection: the union detects more, yet its extra catches barely move the score, for two reasons. First, the additional detections are predominantly implementation-level ambiguities, which the benchmark annotates as non-critical throughout and whose effects are often normalized away by execution-based evaluation. Second, the tight budget clips the larger hint pool before many of those questions reach the user. On AMBROSIA, where neither gate applies, the union does pull clear of SI-1. A budget-aware ranker that raises the critical hints first could recover more of the detection advantage here.

7 Discussion

7.1 Choosing a Method

Ambiguity handling in deployment has two stages: *detection* (what could be asked) and *selection* (which of those a real user sees under limited patience). This paper measures detection in full and probes selection only through BIRD-Interact's heuristic. A principled cross-pool ranker is out of scope. As a coverage recommendation, SI-1 ∪ MCS-1 stands out: only three LLM calls, the largest recall gain, and roughly a quarter of MCS-10's or SI-10's call count. Its pool runs to about 10 clarifications per sample after deduplication, so a deployed system at realistic user patience needs a ranking layer on top. Fall back to SI-1 alone under a strict single-call budget; use SI-10 and / or a union with SI-1 ∪ MCS-10 only when compute is plentiful and implementation-level ambiguity detection coverage is paramount.

7.2 Beyond Text-to-SQL

We conjecture complementarity is not specific to SQL. It comes from two surfaces that every underspecified request has: the wording of the question, which can be vague, and the structured output the model produces, which has to fix choices the wording left open. Introspection reads the first and sampling the second. Beyond SQL, code generation is the clearest example: a short description leaves choices open, such as edge cases or error handling, that candidate implementations resolve differently. Tool and API calls plausibly behave the same way, with open choices such as default argument values.

This is a conjecture, bounded by two conditions. First, the output must genuinely leave decisions open. If the request fully determines the output, the sampled candidates agree and there is nothing for sampling to catch. Second, the disagreement among candidates must be visible: two SQL queries can be compared directly by structure, whereas for other outputs the difference may surface only when executed. Where both conditions hold, one introspection call plus one sampling call is a cheap default that requires no hand-built taxonomy.

7.3 Limitations

Scope. The AST ablation used three strong open-weight models. On weaker or context-limited models the AST scaffold may carry signal we did not measure. The 300-task Lite split limits statistical power for the rarest implementation subtypes. The detection-level findings span four models, and the end-to-end agentic result spans three open-weight models.

Threats to validity. (i) User simulator dependence. All headline numbers route through BIRD-Interact's released single-label encoder. The multi-label extension addresses the encoder's scoring limitation, but other user simulator biases may remain. We report precision alongside recall so inflation, if any, is visible. *(ii) Stress-mode evaluation.* The agentic numbers reflect BIRD-Interact's default budget pressure, not a detection ceiling.

Future work. Three directions follow. *Closing the residual gap:* even the union leaves roughly a third of annotated terms undetected. `decimal` fails twice over. Often no sampled SQL rounds at all, so there is no disagreement to read. When the samples do disagree, the analysis call rarely asks, apparently discounting rounding as cosmetic next to semantic differences. That judgment is defensible in deployment but costs recall wherever the benchmark marks the choice as material. *Taxonomy-based detectors:* a controlled comparison with, or combination of, taxonomy-plus-ICL systems such as AmbiSQL [3] requires equalizing the information budget between zero-shot probes and curated exemplars. Whether exemplars fitted to one benchmark's ambiguity distribution transfer beyond it is itself an open question. *The result surface:* sampled candidates whose execution results differ could be shown to the user directly, with the output schema as a proxy for intent. We leave this unevaluated since execution draws down the same interaction budget that clarifications compete for. Where execution is cheap, the result set is the natural third surface.

8 Conclusion

A single introspection pass can readily identify ambiguities visible in a user's request, but many implementation-level ambiguities remain difficult to surface until the model commits to a concrete program. Our results suggest that ambiguity in Text-to-SQL appears on two complementary surfaces: the natural-language question and the generated SQL. Pairing lightweight introspection with multi-candidate sampling exposes both, revealing silent choices around join type, NULL handling, and division by zero that are often absent from the model's verbalized analysis. These detection gains carry into downstream task success even under a tight clarification budget, and the union's broader coverage leaves further headroom for a selection layer that raises the most critical questions first. As interactive analytics systems become increasingly autonomous,

evaluation frameworks should move beyond tracking user intent alone and also measure how well agents detect, expose, and safely resolve the implementation-level ambiguities they introduce during generation.

References

[1] Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. 2023. Benchmarking and Improving Text-to-SQL Generation under Ambiguity. *arXiv preprint arXiv:2310.13659* (2023). doi:10.48550/arXiv.2310.13659

[2] Debarun Bhattacharjya, Balaji Ganesan, Junkyu Lee, Radu Marinescu, Katya Mirylenka, Michael Glass, and Xiao Shou. 2025. SIMBA UQ: Similarity-Based Aggregation for Uncertainty Quantification in Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 15880–15894. doi:10.18653/v1/2025.findings-emnlp.859

[3] Zhongjun Ding, Yin Lin, Tianjing Zeng, Rong Zhu, Bolin Ding, and Jingren Zhou. 2025. AmbiSQL: Interactive Ambiguity Detection and Resolution for Text-to-SQL. *arXiv preprint arXiv:2508.15276* (2025). doi:10.48550/arXiv.2508.15276

[4] Timo Eckmann, Matthias Urban, Jan-Micha Bodensohn, and Carsten Binnig. 2026. HLR-SQL: Human-like Reasoning for Text-to-SQL with the Human in the Loop. *Information Systems* 138 (2026), 102670. doi:10.1016/j.is.2025.102670

[5] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting Hallucinations in Large Language Models Using Semantic Entropy. *Nature* 630, 8017 (2024), 625–630. doi:10.1038/s41586-024-07421-0

[6] Zhibo Hu, Chen Wang, Yanfeng Shu, Hye-Young Paik, and Liming Zhu. 2025. Ambiguity in LLMs is a Concept-Missing Problem. *arXiv preprint arXiv:2505.11679* (2025). doi:10.48550/arXiv.2505.11679

[7] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning* 110, 3 (2021), 457–506. doi:10.1007/s10994-021-05946-3

[8] Nan Huo, Xiaohan Xu, Jinyang Li, et al. 2025. BIRD-INTERACT: Re-Imagining Text-to-SQL Evaluation for Large Language Models via Lens of Dynamic Interactions. *arXiv preprint arXiv:2510.05318* (2025). doi:10.48550/arXiv.2510.05318

[9] Saurav Kadavath et al. 2022. Language Models (Mostly) Know What They Know. *arXiv preprint arXiv:2207.05221* (2022). doi:10.48550/arXiv.2207.05221

[10] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2022. CLAM: Selective Clarification for Ambiguous Questions with Generative Language Models. *arXiv preprint arXiv:2212.07769* (2022). doi:10.48550/arXiv.2212.07769

[11] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*. doi:10.48550/arXiv.2302.09664

[12] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as a Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*. doi:10.48550/arXiv.2305.03111

[13] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching Models to Express Their Uncertainty in Words. *Transactions on Machine Learning Research* (2022).

[14] Terrance Liu, Shuyi Wang, Daniel Preoţiuc-Pietro, Yash Chandarana, and Chirag Gupta. 2025. Calibrating LLMs for Text-to-SQL Parsing by Leveraging Subclause Frequencies. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. https://aclanthology.org/2025.emnlp-main.859/

[15] Toby Mao. 2023. *SQLGlot: Python SQL Parser and Transpiler*. https://github.com/tobymao/sqlglot

[16] Irina Saparina and Mirella Lapata. 2024. AMBROSIA: A Benchmark for Parsing Ambiguous Questions into Database Queries. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*. doi:10.48550/arXiv.2406.19073

[17] Irina Saparina and Mirella Lapata. 2025. Disambiguate First, Parse Later: Generating Interpretations for Ambiguity Resolution in Semantic Parsing. In *Findings of the Association for Computational Linguistics (ACL Findings)*. doi:10.48550/arXiv.2502.18448

[18] Arindam Sharma and Cristina David. 2025. Assessing Correctness in LLM-Based Code Generation via Uncertainty Estimation. *arXiv preprint arXiv:2502.11620* (2025). doi:10.48550/arXiv.2502.11620

[19] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. 2023. Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

[20] Tim Tomov, Dominik Fuchsgruber, Tom Wollschläger, and Stephan Günnemann. 2025. The Illusion of Certainty: Uncertainty Quantification for LLMs Fails under Ambiguity. *arXiv preprint arXiv:2511.04418* (2025). doi:10.48550/arXiv.2511.04418

[21] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*. doi:10.48550/arXiv.2203.11171

[22] Guanming Xiong, Junwei Bao, Hongfei Jiang, Yang Song, and Wen Zhao. 2025. Multi-Turn Interactions for Text-to-SQL with Large Language Models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25)*. doi:10.1145/3746252.3761052

[23] Miao Xiong, Zhiyuan Hu, Xinyang Lu, et al. 2024. Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*. doi:10.48550/arXiv.2306.13063

[24] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 3911–3921. doi:10.18653/v1/D18-1425