KNOW WHEN TO ABSTAIN: CONFORMAL CONFIDENCE ESTIMATION FOR TEXT-TO-SQL

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-to-SQL systems often return queries that execute successfully but answer the wrong question. Detecting these errors at deployment is hard because execution correctness is defined against a gold query, which exists only in labeled data. We formulate confidence estimation for Text-to-SQL as probabilistic query validation under a latent correctness variable that is observable at calibration time through gold-SQL execution and latent at deployment. Our estimator separates generation from validation. A generator LLM writes and executes the query, and a *different* critic LLM judges the query and its execution result. The critic's validation and execution-grounded alignment scores are fused with the generator's verbalized confidence, token-level uncertainty, and SQL-structural features in a Random Forest, using two sequential LLM calls. Two calibration steps give finite-sample guarantees. A Learn-then-Test threshold ensures, with probability $1 - \delta$, that the error rate among accepted queries stays below a user-chosen budget. A conformal p-value bounds the false abstention rate. On Spider we use GPT-OSS-120B as generator and Gemma-4-31B as critic. The estimator obtains an AUARC of 0.899. Without abstention, 22.6% of queries are wrong. With a 15% risk budget (δ=0.1), a threshold is certified on all 10 resampled splits, and the system accepts 52.4% of queries on average with 9.1% observed error. Five different critic models give similar results. In a controlled comparison on Spider with Olmo-3-32B as generator, the estimator obtains an AUARC of 89.5, compared with 81.2 for the strongest logit-based baseline. Applied to BIRD without retraining, the confidence scores remain informative (AUROC 0.72, against 0.79 on Spider), but the risk guarantee does not transfer, as expected without exchangeability.

1 INTRODUCTION

LLMs reach high execution accuracy on Text-to-SQL benchmarks (Yu et al., 2018; Li et al., 2023), yet many of their errors are queries that execute and return plausible but wrong results (Gong et al., 2025; Somov & Tutubalina, 2025). Such silent failures pass into downstream reports without any execution-time signal, and enterprise workloads with large schemas and domain-specific semantics make them harder to spot (Lei et al., 2025; Chen et al., 2025b). A deployed system therefore needs a per-query confidence score that decides whether to return a result or abstain.

The difficulty is that correctness is defined relative to a gold query. A generated query is correct when its execution result matches that of the gold SQL. On labeled benchmarks this can be checked exactly, but at deployment no gold SQL exists, so correctness is *latent*. Execution success is observable, but it is not correctness. Any confidence score must therefore be inferred from signals available at generation and execution time, and any abstention rule must be fit on labeled data and then applied where no labels exist.

Existing confidence estimators for Text-to-SQL mostly rely on one type of signal. Sampling-based methods treat agreement among K sampled outputs as a proxy for correctness (Wang et al., 2023; Kuhn et al., 2023). They cost K LLM calls, and a model that systematically misreads a schema can produce consistent but wrong queries (Stengel-Eskin & Van Durme, 2023; Ramachandran & Sarawagi, 2024). Logit-based methods aggregate token probabilities, possibly by SQL clause (Entezari Maleki et al., 2026; Ramachandran & Sarawagi, 2024). They need a single pass, but token probabilities reflect next-token uncertainty and can remain high for fluent queries with the wrong

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

intent (Stengel-Eskin & Van Durme, 2023; Desai & Durrett, 2020). These methods are usually evaluated by discrimination and calibration, and with few exceptions (Chen et al., 2025a) the abstention threshold is not tied to a stated statistical property.

We treat confidence estimation as probabilistic query validation and separate the roles of generation and validation. A generator LLM writes the query and reports its own confidence. A *different* critic LLM then validates the query and checks whether its execution result answers the question. Using a separate critic means the judgment is not made by the model whose reasoning produced the error. We combine the critic's validation and execution-grounded alignment scores with the generator's verbalized confidence, token-level uncertainty, and SQL-structural features. These signals take two sequential LLM calls, one per model, plus local computation. A Random Forest fuses them and is trained where correctness is observable. The acceptance threshold is then chosen on a labeled calibration set with Learn then Test (Angelopoulos et al., 2025). The result guarantees, with high probability, that the error rate among accepted queries stays below a budget the user chooses. A conformal p-value computed from the same scores also bounds the fraction of correct queries on which the system abstains.

On Spider, with GPT-OSS-120B as generator and Gemma-4-31B as critic, a 15% risk budget is certified on every resampled split. At that budget the system accepts 52.4% of queries with 9.1% observed error, against 22.6% error without abstention. In a controlled comparison with a shared generator, the estimator obtains an AUARC of 89.5, compared with 81.2 for SAC-Product and 82.1 for five-sample self-consistency. Our contributions are:

- **Formulation.** We formalise confidence estimation for Text-to-SQL under latent execution correctness, separating what is observable at calibration time from what is latent at deployment. We state which selective-prediction quantities are guaranteed and which are empirical (Section 3).
- **Estimator.** We propose a generator–critic estimator. A separate critic model validates the generator's query and its execution result, and a Random Forest fuses the critic's judgments with generator-side signals. An ablation shows that the LLM-derived signals carry most of the information, so a variant without logit access retains most of the performance (Sections 4, 5.4).
- **Calibrated abstention.** We select the acceptance threshold with Learn then Test, which controls the error rate among accepted queries with high probability. We also derive a conformal bound on false abstention, and we characterise the resulting coverage–risk tradeoff (Sections 3.1, 5.2).
- **Evaluation.** We compare five critic models with self-critique. We also report a controlled baseline comparison, results for four generator backbones, 10-seed resampling, zero-shot transfer to BIRD and NL2SQL-BUGs, and a variant without logit access (Section 5).

2 Related Work

Error detection and confidence for Text-to-SQL. Prior work estimates query-level confidence from model probabilities and their rescaling (Ramachandran & Sarawagi, 2024; Stengel-Eskin & Van Durme, 2023), from SQL-aware logit aggregation and execution-based grounding (Entezari Maleki et al., 2026; Somov & Tutubalina, 2025), or from sub-clause frequencies with multivariate Platt scaling (Liu et al., 2025a). SQLens combines database and LLM signals to detect and correct clause-level errors (Gong et al., 2025), SQLFixAgent repairs queries through multi-agent consistency checks (Cen et al., 2025), and NL2SQL-BUGs provides a benchmark for semantic error detection (Liu et al., 2025b). These methods produce scores or corrections. We combine several signal families into one estimator and attach an abstention rule with a stated calibration property.

Uncertainty estimation for LLMs. White-box methods use token-level probabilities (Hendrycks & Gimpel, 2017; Malinin & Gales, 2021). Black-box methods use sampling consistency or semantic equivalence (Wang et al., 2023; Kuhn et al., 2023; Farquhar et al., 2024), whose underlying assumption is examined by Xiao et al. (2025), or elicit verbalized confidence (Kadavath et al., 2022; Xiong et al., 2024; Tian et al., 2023). We use verbalized and logit signals as input features rather than as standalone scores.

Selective prediction and conformal methods. Selective classification abstains on low-confidence inputs to lower the error rate among accepted predictions (Geifman & El-Yaniv, 2017); Wen et al. (2025) survey abstention in LLMs. TrustSQL evaluates Text-to-SQL reliability with penalty-based scoring that rewards abstention (Lee et al., 2024), and Chen et al. (2025a) use conformal prediction to abstain during schema linking. Conformal prediction gives finite-sample guarantees under exchangeability (Vovk et al., 2005; Angelopoulos & Bates, 2023), and has been applied to language generation (Quach et al., 2024) and to LLMs without logit access (Su et al., 2024). We control the selective error rate with Learn then Test (Angelopoulos et al., 2025), which casts threshold selection as multiple hypothesis testing. Our conformal confidence is the outlier-detection construction (Bates et al., 2023) with correct queries as the reference class, and it controls false abstention.

3 Problem Formulation

Setting. Let $x = (q, s)$ pair a natural-language question q with a database schema s, and let D be the database instance. Throughout, s denotes the schema, $\hat{s}$ the generated SQL query, and s^* the gold SQL query. Two models are involved. A *generator* θ_G defines an autoregressive distribution $P_{\theta_G}(\hat{s} \mid x) = \prod_{t=1}^{T} P_{\theta_G}(\hat{s}_t \mid \hat{s}_{<t}, x)$, where T is the number of tokens in $\hat{s}$. A decoding procedure returns a single query $\hat{s}$, whose tokens $\tau = (t_1, \ldots, t_T)$ have probabilities $\pi_1, \ldots, \pi_T$. A *critic* θ_C, a different LLM from θ_G, judges $\hat{s}$ but does not generate it.

Execution correctness as a latent variable. Let $\mathrm{Exec}(\hat{s}, D)$ denote deterministic execution. The execution-correctness variable is

$$Z(x, \hat{s}) = \mathbb{1}[\mathrm{Exec}(\hat{s}, D) = \mathrm{Exec}(s^*, D)]\,. \tag{1}$$

The result $r = \mathrm{Exec}(\hat{s}, D)$ is always obtainable for read-only SELECT queries. Z is latent because s^* is unavailable at deployment, not because execution is infeasible. On labeled data, Z is computed exactly by execution matching, which is how we obtain training and calibration labels.

Confidence function. A confidence function $C : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ maps $(x, \hat{s})$ to a score using only deployment-time signals: q, s, $\hat{s}$, the token probabilities $\{\pi_t\}$, the execution result r, the generator's verbalized confidence, and the critic's judgments. Since Z is unobserved at deployment, we assess C on labeled test data by discrimination (ranking correct above incorrect queries), calibration (agreement of scores with empirical correctness), and selective prediction.

3.1 Selective Prediction and What Is Guaranteed

Given a confidence score $S(x, \hat{s})$ and a threshold λ, the selective predictor is

$$\mathcal{A}_\lambda(x) = \hat{s} \;\text{ if } S(x, \hat{s}) \geq \lambda, \qquad \mathcal{A}_\lambda(x) = \perp \text{ (abstain) otherwise.} \tag{2}$$

We use four population quantities:

$$\mathrm{Coverage}(\lambda) = \mathbb{P}(S \geq \lambda)\,, \qquad \mathrm{Risk}(\lambda) = \mathbb{P}(Z = 0 \mid S \geq \lambda)\,, \tag{3}$$

$$\mathrm{FAR}(\lambda) = \mathbb{P}(S < \lambda \mid Z = 1)\,, \qquad \mathrm{TAR}(\lambda) = \mathbb{P}(S < \lambda \mid Z = 0)\,. \tag{4}$$

FAR (false abstention rate) is the fraction of correct queries withheld, and TAR (true abstention rate) the fraction of incorrect queries withheld. With base accuracy $\rho = \mathbb{P}(Z{=}1)$, they determine the other two quantities:

$$\mathrm{Coverage}(\lambda) = \rho\,(1 - \mathrm{FAR}) + (1 - \rho)(1 - \mathrm{TAR}), \qquad \mathrm{Risk}(\lambda) = \frac{(1 - \rho)(1 - \mathrm{TAR})}{\mathrm{Coverage}(\lambda)}. \tag{5}$$

Both calibration rules below use the fused estimate $p(x, \hat{s}) = \hat{\mathbb{P}}(Z{=}1 \mid \mathbf{x})$ from Section 4.4, trained on a separate split, and a held-out calibration set $\mathcal{C}$ of size m with observed Z_i. Queries that fail to execute are always rejected.

Conformal confidence and false abstention. Let $\mathcal{C}_1 = \{i \in \mathcal{C} : Z_i = 1\}$ with $m_1 = |\mathcal{C}_1|$, and define the nonconformity score $\eta = -\log p$. The calibrated confidence is the conformal p-value

$$\hat{c}(x, \hat{s}) = \frac{1 + \left|\{i \in \mathcal{C}_1 : \eta_i \geq \eta(x, \hat{s})\}\right|}{m_1 + 1}, \tag{6}$$

with $\hat{c} = 0$ when $\hat{s}$ fails to execute. The *conformal rule* uses $S = \hat{c}$ and $\lambda = 1 - \alpha$ for an operating level $\alpha \in (0, 1]$.

Proposition 1 (Finite-sample bound on false abstention)**.** *Suppose a correct test query $(x, \hat{s})$ with $Z = 1$ is exchangeable with $\{(x_i, \hat{s}_i)\}_{i \in \mathcal{C}_1}$, conditional on the trained estimator p. Then $\mathbb{P}(\hat{c} \leq t \mid Z = 1) \leq t$ for every $t \in [0, 1]$. Consequently, the conformal rule satisfies*

$$\mathrm{FAR}(1-\alpha) \;\leq\; 1-\alpha, \qquad \textit{equivalently} \qquad \mathbb{P}\big(\hat{c} \geq 1-\alpha \mid Z=1\big) \;\geq\; \alpha, \tag{7}$$

where the probability is over the calibration set and the test query.

This is the validity of conformal p-values used for outlier detection (Angelopoulos & Bates, 2023, Sec. 4.4; see also Bates et al., 2023), with correct queries as the reference class (proof in Appendix E). By Eq. 5, the conformal rule accepts about a fraction α of correct queries, so a small α gives high precision at low coverage by construction. It does not bound Risk.

Risk control. To bound the error rate among accepted queries, the user fixes a risk budget ε and a failure probability δ, and we select the threshold with Learn then Test (Angelopoulos et al., 2025). Let Λ be a grid of thresholds on p fixed before calibration. For each $\lambda \in \Lambda$, let $n(\lambda)$ be the number of calibration queries with $p_i \geq \lambda$, and $k(\lambda)$ the number of those with $Z_i = 0$. We test $H_\lambda : \mathrm{Risk}(\lambda) > \varepsilon$ with the binomial p-value $u_\lambda = \mathbb{P}(\mathrm{Bin}(n(\lambda), \varepsilon) \leq k(\lambda))$, and set $\hat{\lambda} = \min\{\lambda \in \Lambda : u_\lambda < \delta/|\Lambda|\}$. The *risk-controlled rule* uses $S = p$ and $\lambda = \hat{\lambda}$. If no λ qualifies, it abstains on every query.

Proposition 2 (Finite-sample risk control)**.** *If the calibration and test queries are i.i.d. and p is trained on data disjoint from $\mathcal{C}$, then*

$$\mathbb{P}_{\mathcal{C}}\big(\mathrm{Risk}(\hat{\lambda}) \leq \varepsilon\big) \;\geq\; 1-\delta. \tag{8}$$

Conditional on $n(\lambda)$, $k(\lambda)$ is binomial with success probability $\mathrm{Risk}(\lambda)$, so u_λ is a valid p-value for H_λ (Angelopoulos & Bates, 2023, Sec. 5.5). The Bonferroni correction controls the probability of certifying any threshold whose risk exceeds ε (Appendix E). Both propositions need gold SQL only on the calibration set, and hold at deployment only while deployment queries follow the calibration distribution (exchangeable for Proposition 1, i.i.d. for Proposition 2). Coverage under the risk-controlled rule depends on how well p separates correct from incorrect queries. A tight budget may not be certifiable with few calibration queries, in which case the rule abstains on every query.

4 METHOD

The estimator uses two different LLMs with separate roles (Figure 1). The generator θ_G writes the query, and the critic $\theta_C \neq \theta_G$ judges it. Stage 1 (generator, LLM call 1) generates $\hat{s}$ with a verbalized confidence and token probabilities, and executes $\hat{s}$. Stage 2 (critic, LLM call 2) validates $\hat{s}$ and scores its execution result. Stage 3 (local) computes probabilistic and structural features from the generator's token probabilities. Stage 4 (local) fuses all features, applies conformal calibration, and selects the risk-controlled threshold. Stages 2 and 3 run in parallel, so the pipeline makes two sequential LLM calls, one to each model. Table 1 lists the feature families and the model or stage that produces each.

4.1 STAGE 1: GENERATION AND EXECUTION (GENERATOR)

The generator receives (q, s), where the schema includes sample rows, and a generation prompt (Appendix A.1). Before answering, it checks its query internally for question alignment, grounding in the schema, literal values, SQLite syntax, logical correctness, result sanity, and minimality. Its output contains only $\hat{s} = f_{\mathrm{gen}}(q, s)$ and a verbalized confidence $c_{\mathrm{gen}} \in [0, 100]$. The prompt asks for a confidence that reflects the weakest critical aspect of the query, not its syntactic validity. The generator returns no validation verdict. Verdicts come only from the critic. We record the probabilities π_t of the tokens of $\hat{s}$ for Stage 3. We then execute $\hat{s}$ on D to obtain $r = \mathrm{Exec}(\hat{s}, D)$. If execution fails, $Z{=}0$ is certain, and the query receives $\hat{c}{=}0$ without further processing.

4.2 STAGE 2: CRITIC VALIDATION AND EXECUTION-GROUNDED ALIGNMENT (CRITIC)

The critic receives $(q, s, \hat{s}, r)$ in a single prompt (Appendix A.2). It does not see c_{gen} or the generator's reasoning, and it has no access to gold SQL or gold answers. It judges the executed result

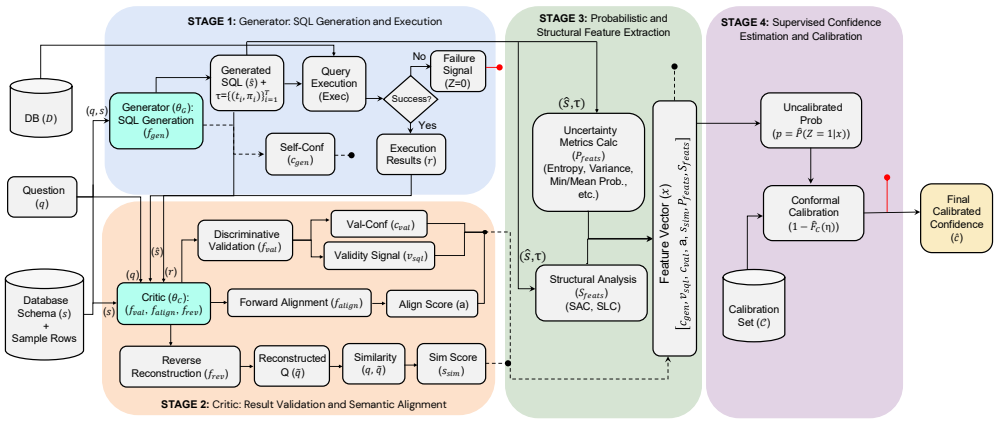

Figure 1: The four-stage generator–critic estimator. Stage 1 (generator θ_G, LLM call 1) generates $\hat{s}$ with its tokens and probabilities τ and a self-confidence c_{gen}, then executes $\hat{s}$ on D. A failed execution sets $Z{=}0$ and $\hat{c}{=}0$. Stage 2 (critic $\theta_C \neq \theta_G$, LLM call 2) receives $(q, s, \hat{s}, r)$. It returns a validity verdict v_{sql} with confidence c_{val}, an alignment score a, and the similarity s_{sim} between q and the question $\tilde{q}$ reconstructed from r. Stage 3 (local) computes uncertainty and structural features from the generator's $(\hat{s}, \tau)$ and runs in parallel with Stage 2. Stage 4 (local) fuses the feature vector $\mathbf{x}$ with a Random Forest and maps its posterior to the calibrated confidence $\hat{c}$ using a calibration set $\mathcal{C}$ labeled offline.

against the question, not merely whether the values look plausible. Verbalized judgments can surface semantic problems that token probabilities do not (Kadavath et al., 2022; Xiong et al., 2024).

Discriminative validation. The critic checks whether r correctly and completely answers q. The checks cover required columns and rows, the requested operation or aggregation, filters and comparison operators, ordering and top-k limits, grouping granularity, date and time constraints, the consistency of returned values, and completeness. Violating any substantive requirement makes the result incorrect. The critic returns a binary verdict $v_{\text{sql}} = f_{\text{val}}(q, s, \hat{s}, r) \in \{0, 1\}$ and its confidence in that verdict, $c_{\text{val}} \in [0, 100]$.

Execution-grounded alignment. Ignoring q, the critic reconstructs the most precise question $\tilde{q} = f_{\text{rev}}(s, r)$ that r answers, using only r and the schema, without repairing missing conditions. It also returns an alignment score $a = f_{\text{align}}(q, s, \hat{s}, r) \in [0, 100]$, which measures how well the result answers q in terms of the requested information, filters, aggregation, ordering, grouping, dates, and completeness. The critic also returns the semantic similarity $s_{\text{sim}} = \text{Sim}(q, \tilde{q}) \in [0, 100]$. The prompt caps a and s_{sim} when $v_{\text{sql}} = 0$. A low a or s_{sim} flags results that are topically related but answer a different question.

4.3 Stage 3: Probabilistic and Structural Features

From the generator's token probabilities $\{\pi_t\}_{t=1}^{T}$ of $\hat{s}$, we compute without further LLM calls:

$$\text{FTC} = \prod_t \pi_t, \quad \pi_{\min} = \min_t \pi_t, \quad \mu_{\text{arith}} = \tfrac{1}{T}\sum_t \pi_t, \quad \mu_{\text{geom}} = \left(\prod_t \pi_t\right)^{1/T}, \tag{9}$$

$$H = -\sum_t \pi_t \log \pi_t, \quad \sigma^2 = \tfrac{1}{T}\sum_t (\pi_t - \mu_{\text{arith}})^2, \quad \ell_{\text{CLM}} = -\sum_t \log \pi_t. \tag{10}$$

We also recompute the first five statistics on SQL keyword and non-keyword tokens separately, so that uncertainty confined to identifiers or literals is not averaged away. We add the probability range and a 13-bin probability histogram. Token probabilities do not distinguish schema identifiers from other tokens, so we add SQL-aware features (Entezari Maleki et al., 2026; Somov & Tutubalina, 2025). Let $\mathcal{G}$ partition the tokens of $\hat{s}$ into clause groups (SELECT, WHERE, table identifiers, and so on), and let τ_{sch} be the tokens that name schema elements. We compute

$$\text{SAC}_{\text{geom}} = \left(\prod_{g\in\mathcal{G}} \tfrac{1}{|g|}\sum_{t\in g} \pi_t\right)^{1/|\mathcal{G}|}, \qquad \text{SLC} = \left(\prod_{t\in\tau_{\text{sch}}} \pi_t\right)^{1/|\tau_{\text{sch}}|} \ (\text{SLC}{=}0 \text{ if } \tau_{\text{sch}}{=}\emptyset). \tag{11}$$

Table 1: Features in $\mathbf{x}$, the model and stage that produce them, and the failure mode they target. θ_G: generator; θ_C: critic. The last row is listed in full in Appendix Table 7.

Family	Features	Source	Dim	Targeted failure mode
Verbalized	c_{gen}	Stage 1, θ_G	1	Generator uncertainty
	v_{sql}, c_{val}	Stage 2, θ_C	2	Result violates question
Alignment	a, s_{sim}	Stage 2, θ_C	2	Plausible but wrong results
Logit (P_{feats})	$\text{FTC}, \pi_{\min}, \mu_{\text{arith}}, \mu_{\text{geom}}, H, \sigma^2, \ell_{\text{CLM}}$	Stage 3, local	7	Token-level uncertainty
SQL-structural (S_{feats})	SAC_{geom}, SLC	Stage 3, local	2	Schema grounding errors
Variants	Subset statistics, range, histogram, schema-table mean, C3S	Stage 3, local	27	Localised uncertainty
Total			**41**	

SLC targets schema grounding, a primary failure mode in cross-domain Text-to-SQL (Yu et al., 2018; Li et al., 2023). A schema-table mean and a clause-consistency score (C3S) complete this family. The full vector has 41 features (Table 1; each one is listed in Appendix B). Some are near-duplicates, for example $\ell_{\text{CLM}} = -\log \text{FTC}$, and a tree ensemble is unaffected by such redundancy.

4.4 Stage 4: Supervised Fusion and Conformal Calibration

A Random Forest f_{RF} maps $\mathbf{x}$ to $p = \hat{\mathbb{P}}(Z{=}1 \mid \mathbf{x})$. It is trained on labeled queries whose Z is computed by execution matching. The forest handles features on different scales without normalisation and can use feature interactions, for example high FTC with low alignment. No single feature separates the two classes well (Appendix C). On a disjoint calibration split, we compute $\eta_i = -\log p_i$ for correct queries and use Eq. 6 to map p to $\hat{c}$. Because $\hat{c}$ is a non-decreasing function of p, calibration preserves the ranking of queries up to ties, and we report AUROC and AUARC of p. The three phases are separated as follows. Training and calibration use gold SQL to compute Z. Deployment uses only the generator, the critic, the local Stage 3 features, the frozen forest, and the stored calibration scores.

5 Experiments

5.1 Experimental Setup

Data and models. We use 3,181 question–SQL pairs from the Spider dev and test sets (Yu et al., 2018), covering 200 databases. After removing pairs whose gold SQL fails to execute, 3,175–3,179 remain, depending on the generation run. For zero-shot transfer we use the BIRD dev set (Li et al., 2023) (1,534 samples) and NL2SQL-BUGs (Liu et al., 2025b) (1,019 correct and 999 semantically erroneous queries). Model details are in Appendix G.1.

Generator and critic models. Different models always serve in Stages 1 and 2, and Gemma-4-31B-it is the critic unless stated otherwise. The main configuration uses GPT-OSS-120B as generator. The four-backbone study uses GPT-OSS-120B, Olmo-3-32B-Think, Qwen2.5-72B-Instruct, and Llama-3.3-Nemotron-Super-49B-v1.5 as generators. The controlled comparison and the ablations use Olmo-3-32B-Think. To test the choice of critic, we also run Qwen3.8-27B, DeepSeek-V4.1-Flash, Nemotron-3-Super, and Qwen2.5-72B-Instruct as critics for GPT-OSS-120B. As a reference, we report self-critique, where the generator acts as its own critic ($\theta_C = \theta_G$).

Protocol. Spider is split, stratified by Z, into training (60%), calibration (20%), and test (20%) sets, for example 1,907 / 636 / 636 queries. For each configuration, a separate Random Forest is trained on features computed from that configuration's outputs, with hyperparameters chosen by 5-fold cross-validated random search on the training split (Appendix D). Conformal calibration uses the calibration split, and all metrics are computed on the test split. For zero-shot transfer, forests and calibration scores fit on Spider are applied unchanged to BIRD and NL2SQL-BUGs (Section 5.6). No target-domain data are used for training or calibration.

Metrics. *Discrimination*: AUROC, and AUARC (area under the accuracy–rejection curve, our primary metric). *Calibration*: ECE (10 equal-width bins), ACE (equal-mass bins), and Brier score. *Selective prediction*: Coverage, Risk, TAR, and FAR (Eqs. 3–4), estimated on the test split, and

Table 2: Critic comparison on the Spider test split (n=636, generator GPT-OSS-120B). Selective metrics use the conformal rule at α=0.1, for which Proposition 1 bounds FAR ≤ 0.9 in expectation. The self-critique run uses a separate generation pass, so its base accuracy differs.

	Self-critique	Critic				
	($\theta_C = \theta_G$)	Gemma-4-31B	Qwen3.8-27B	DeepSeek-V4.1	Nemotron-3-S	Qwen2.5-72B
Base accuracy ρ	0.786	0.774	0.774	0.774	0.774	0.774
ECE ↓	0.149	0.158	0.155	0.156	0.142	0.166
AUROC ↑	0.781	0.785	0.840	0.811	0.779	0.826
AUARC ↑	0.913	0.899	0.922	0.916	0.901	0.915
Coverage	0.077	0.079	0.063	0.080	0.074	0.082
CondAcc = 1−Risk	0.980	0.940	1.000	1.000	0.957	0.981
TAR	0.993	0.979	1.000	1.000	0.986	0.993
FAR (bound: ≤ 0.9)	0.904	0.904	0.919	0.896	0.909	0.896

Table 3: Risk-controlled abstention (Proposition 2, δ=0.1) on Spider over 10 resampled splits. Cert.: splits on which a threshold is certified; otherwise the system abstains on every query. Cov.: coverage (%) averaged over all 10 splits, counting uncertified splits as zero. Risk: test-split error among accepted queries (%), mean (max) over certified splits. Critic rows use generator GPT-OSS-120B; generator rows use the Gemma critic. *Generation run of the four-backbone study.

	$\varepsilon = 0.10$			$\varepsilon = 0.15$			$\varepsilon = 0.20$		
	Cert.	Cov.	Risk	Cert.	Cov.	Risk	Cert.	Cov.	Risk
Critic: Gemma-4-31B	0/10	0.0	—	10/10	52.4	9.1 (11.7)	10/10	74.2	14.1 (16.1)
Critic: Qwen3.8-27B	7/10	27.6	4.7 (8.0)	10/10	64.1	9.9 (14.6)	10/10	79.8	14.2 (16.0)
Critic: DeepSeek-V4.1	6/10	21.9	4.9 (9.1)	10/10	55.4	8.7 (11.9)	10/10	75.3	14.0 (15.9)
Critic: Nemotron-3-Super	6/10	19.6	5.0 (8.2)	10/10	54.2	9.3 (11.4)	10/10	72.7	14.6 (17.2)
Gen.: Olmo-3-32B	1/10	1.9	7.4 (7.4)	10/10	38.8	8.2 (12.1)	10/10	67.5	14.1 (17.8)
Gen.: Qwen2.5-72B	4/10	14.4	5.9 (7.8)	10/10	62.2	9.7 (12.6)	10/10	87.7	15.5 (17.0)
Gen.: Llama-49B	2/10	7.2	7.0 (7.1)	10/10	51.9	8.6 (11.5)	10/10	74.1	14.3 (16.0)
Gen.: GPT-OSS-120B*	8/10	23.4	4.1 (6.9)	10/10	57.1	9.3 (12.7)	10/10	76.6	14.2 (16.4)

conditional accuracy $\mathrm{CondAcc} = 1 - \mathrm{Risk}$, the accuracy among accepted queries. Risk is covered by Proposition 2 for the risk-controlled rule. FAR is covered by Proposition 1 for the conformal rule. Unless stated otherwise, we use $\delta = 0.1$ and a grid $\Lambda = \{0.01, 0.02, \ldots, 0.99\}$. Appendix F gives full definitions. Per-query latency is reported in Appendix G.9.

5.2 Generator–Critic Results

Table 2 compares five critics with self-critique. AUARC ranges from 0.899 (Gemma-4-31B) to 0.922 (Qwen3.8-27B), and self-critique obtains 0.913. These differences are of the same order as the 10-seed variability (Section 5.5), so a separate critic keeps discrimination while removing self-judgment. Under the conformal rule at α=0.1, every configuration accepts 6.3–8.2% of queries, 94.0–100% of them correct, with FAR (0.896–0.919) close to its bound of 0.9. Partial configurations are in Appendix G.4. Across four generator backbones with the Gemma critic, AUARC ranges from 0.895 to 0.918 (Appendix Table 11).

Risk-controlled abstention. The conformal rule at α=0.1 accepts few queries by construction, since it fixes the accepted fraction of correct queries at about α (Section 3.1). The risk-controlled rule instead takes the most lenient threshold for which the calibration data certify $\mathrm{Risk} \leq \varepsilon$. Table 3 reports it over 10 resampled splits. At ε=0.15, a threshold is certified on all 80 splits of the eight configurations, and the system accepts 38.8–64.1% of queries on average. With the Gemma critic it accepts 52.4%, compared with 7.9% under the conformal rule. At ε=0.20, coverage reaches 67.5–87.7%. Across all 194 certified runs, the test-split risk never exceeds ε. A budget of ε=0.10 is certified on only 0–8 of 10 splits. With 636 calibration queries, the binomial bound is too wide for the small accepted sets at the strict end. The low coverage of the conformal rule therefore reflects the rule, not the estimator (sweep in Appendix G.3).

Table 4: Controlled comparison on Spider. All methods use Olmo-3-32B-Think as generator and share splits and evaluation code. †: logit-based. Calls: LLM calls per query. Per-query latency of our pipeline is reported in Appendix G.9.

Method	**Calls**	**AUARC**	**AUROC**	**ECE**
Self-Consistency (K=3)	3	79.3	68.4	0.198
Self-Consistency (K=5)	5	82.1	71.2	0.183
Single CoT Verify	2	78.9	63.6	0.030
SAC-Product† (Entezari Maleki et al., 2026)	1	81.2	72.4	0.161
Pooled Token-Level† (Ramachandran & Sarawagi, 2024)	1	79.7	70.8	0.174
Ours	**2**	**89.5**	**81.6**	0.126

Table 5: Leave-one-family-out ablation (generator Olmo-3-32B-Think, critic Gemma-4-31B, Spider). 10 seeds: mean ± s.d. over resampled splits. ECE of the forest posterior p.

Configuration	AUARC	10 seeds	ECE
Full model	89.5	88.5 ± 0.8	0.126
w/o Verbalized	87.6	87.6 ± 0.8	0.124
w/o Alignment	87.8	87.4 ± 0.8	0.134
w/o Logit P_{feats}	90.5	88.3 ± 0.9	0.143
w/o Structural S_{feats}	89.0	88.1 ± 0.8	0.122
Logit-free (5 LLM scores)	88.6	86.9 ± 1.2	0.187
SAC-Product (baseline)	*81.2*	—	*0.161*

Table 6: Incremental build-up from Single CoT Verify (same setting), retraining the forest at each row. All additions are local.

Configuration	AUARC	Δ
Single CoT Verify (2 calls)	78.9	—
Forest on verbalized scores	84.5	+5.6
+ Logit/structural features	87.9	+9.0
+ Alignment (Full)	89.5	+10.6

5.3 Controlled Comparison Against Baselines

All methods in Table 4 are re-implemented on Olmo-3-32B-Think with the same splits and evaluation code. Baselines are SAC-Product (Entezari Maleki et al., 2026) and Pooled Token-Level (Ramachandran & Sarawagi, 2024) (logit-based), Self-Consistency with K=3, 5 (Wang et al., 2023), and Single CoT Verify, which uses the same generator and critic calls without feature fusion or conformal calibration. Numbers reported in the original papers use other backbones and protocols. We list them in Appendix G.10 for reference only.

Our estimator obtains an AUARC of 89.5, compared with 81.2 for SAC-Product and 82.1 for Self-Consistency (K=5). A bootstrap over the test split (B=1,000) gives a 95% interval of $(86.7, 91.9)$, whose lower bound lies above every baseline in the table. Single CoT Verify uses the same two calls and obtains 78.9. The 10.6-point gap therefore comes from feature fusion, not from additional LLM calls. Self-Consistency with five calls remains 7.4 points lower, which is consistent with agreement among samples failing to detect systematic errors (Kuhn et al., 2023; Ramachandran & Sarawagi, 2024). Single CoT Verify has the lowest ECE but ranks queries poorly; our method targets ranking and risk control.

5.4 Ablation Study

Table 5 removes one feature family at a time and retrains the forest. Removing the verbalized or alignment family lowers AUARC by 0.9 and 1.1 points over 10 seeds. Removing the logit or structural family changes AUARC by less than the seed variability, and on the single split removing the logit family even raises it. With a separate critic, the LLM-derived families carry most of the signal. Conformal calibration does not change the ranking (a monotone map); it provides the thresholds of Propositions 1 and 2. Since $\hat{c}$ is a conformal p-value, not a probability, we report ECE for p only.

In Table 6, fusing the three verbalized scores raises AUARC from 78.9 to 84.5, token-level and structural features add 3.4 points, and alignment adds 1.6.

Logit-free variant. Closed APIs often do not expose logits. Keeping only the five LLM-derived scores (no token-based feature) gives an AUARC of 88.6, 0.9 points below the full model (1.6 over

10 seeds) and 7.4 above SAC-Product, which itself needs logits. We emulate this setting and have not evaluated closed-API models directly.

5.5 Seed Variability and Signal Reliability

We resampled the splits 10 times with hyperparameters fixed, for all eight configurations of Table 3 (Appendix G.5). The 10-seed mean AUARC lies within 0.013 of the single-split value in each case. For the Gemma critic, AUARC is 0.900 ± 0.010. Single-split AUROC differs from the 10-seed mean by up to 0.029, so AUROC differences of that size should not be over-interpreted. Mean FAR at α=0.1 is 0.889–0.907, close to the bound of Proposition 1. Each LLM-derived score has a standalone AUROC between 0.525 and 0.771, above the 0.5 chance level, for all four generators and all four critics (Appendix G.6), and below the forest's AUROC of 0.765–0.840. Appendix G.7 analyses the errors that the critic misses.

5.6 Zero-Shot Cross-Dataset Transfer

We apply the Spider-fit forests and calibration scores to BIRD and NL2SQL-BUGs without retraining or recalibration. The target data are not exchangeable with Spider, so neither proposition applies.

BIRD. BIRD has more complex joins, nested aggregations, and longer schemas than Spider, and only 34.2% of generated queries are correct. The estimator obtains an AUROC of 0.724 (0.791 in-domain) and an AUARC of 0.515, where random and perfect rankings give 0.343 and 0.709. Its lower ECE than on Spider (0.091 vs 0.133) reflects the base rate, not better calibration (Appendix G.8). The risk threshold certified on Spider does not carry over: at ε=0.15 the observed BIRD risk is 30.1%, which shows why Proposition 2 requires i.i.d. data.

NL2SQL-BUGs. This benchmark supplies SQL written by other systems, so only Stage 2 applies. *Our Stage 2 critic alone, with an open 31B model and execution grounding, reaches 77.4% accuracy on NL2SQL-BUGs, on par with the best models reported by Liu et al. (2025b) (76.8%)*, although prompts, models, and inputs differ. A Spider-trained forest on the critic's scores transfers less well (F1 0.708 vs. 0.779 for the verdict), as generator-side features are missing and the error rate differs.

6 Conclusion

We studied confidence estimation for Text-to-SQL when execution correctness is observable on labeled data but latent at deployment. A generator writes and executes the query, a different critic model validates it, and a Random Forest fuses both models' signals. A Learn-then-Test threshold controls the error rate among accepted queries with high probability, and a conformal p-value bounds false abstention. With GPT-OSS-120B as generator and Gemma-4-31B as critic, a 15% risk budget is certified on every resampled split of Spider, and the system accepts 52.4% of queries at 9.1% observed error. In a controlled comparison with a shared generator, the estimator improves AUARC by 8.3 points over the strongest logit-based baseline. Five critic models give similar results, and a variant without logit access loses 1.6 AUARC. On BIRD, the Spider-trained estimator still separates correct from incorrect queries without retraining (AUROC 0.72), but the Spider risk threshold no longer holds.

Limitations. (1) Scope of the guarantees. Both propositions assume that deployment queries follow the calibration distribution, and neither holds under distribution shift. Coverage is not guaranteed, and with 636 calibration queries a 10% risk budget is often not certifiable. **(2) Dependence on LLM judgments.** The largest feature families come from LLM judgments. A separate critic removes self-judgment, but the critic is itself an LLM, and errors that it misreads remain hard to detect (Appendix G.7). **(3) Logit access and overhead.** The full model needs the generator's token probabilities and two sequential LLM calls, about 4.3 s per query in our setup (Appendix G.9). **(4) Evaluation scope.** We evaluate SQLite benchmarks and open-weight models; Spider 2.0 (Lei et al., 2025) and other dialects remain untested. Only the alignment step and the structural features depend on SQL; code with unit tests has an execution analogue, but we have not evaluated other tasks (Appendix H).

AI USE STATEMENT

We used Large Language Models (LLMs) to correct grammar and to assist with writing. LLMs were not involved in any part of the research discussions, analyses, or idea generation. All insights, contributions, and intellectual content are entirely the authors' own.The language models evaluated in this paper, as generators and critics, are objects of study, not authoring tools.

ETHICS STATEMENT

This work uses public benchmarks (Spider, BIRD, NL2SQL-BUGs) that contain no personal data, and open-weight models. Its aim is to reduce silently wrong database answers reaching users. A deployment that treats accepted queries as verified could still act on the remaining errors. The error rate among accepted queries is bounded only with probability $1 - \delta$ and under i.i.d. sampling (Section 3.1).

REPRODUCIBILITY STATEMENT

Section 3 states the formal setting and assumptions. Section 5.1 describes the data, splits, and protocol. Appendix A.1 and A.2 give the full prompts, Appendix D the forest hyperparameters, Appendix F the metric definitions, and Appendix G.1 the model checkpoints. Code for feature extraction, training, calibration, and evaluation will be released upon paper acceptance.

REFERENCES

Anastasios N Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023. Section, proposition, and theorem numbers cited in this paper refer to arXiv:2107.07511v6.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025.

Stephen Bates, Emmanuel Candès, Lihua Lei, Yaniv Romano, and Matteo Sesia. Testing for outliers with conformal p-values. *The Annals of Statistics*, 51(1):149–178, 2023.

Jipeng Cen, Jiaxin Liu, Zhixu Li, and Jingjing Wang. Sqlfixagent: Towards semantic-accurate text-to-sql parsing via consistency-enhanced multi-agent collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 49–57, 2025.

Kaiwen Chen, Yueting Chen, Nick Koudas, and Xiaohui Yu. Reliable text-to-sql with adaptive abstention. *Proceedings of the ACM on Management of Data*, 3(1):1–30, 2025a.

Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. BEAVER: An enterprise benchmark for text-to-SQL. In *Table Representation Learning Workshop at ACL*, 2025b.

Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 295–302, 2020.

Sepideh Entezari Maleki, Mohammadreza Pourreza, and Davood Rafiei. Confidence estimation for text-to-SQL in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32474–32482, 2026. arXiv:2508.14056.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

Yue Gong, Chuan Lei, Xiao Qin, Kapil Vaidya, Balakrishnan Narayanaswamy, and Tim Kraska. SQLens: An end-to-end framework for error detection and correction in text-to-SQL. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023.

Gyubok Lee, Woosog Chay, Seonhee Cho, and Edward Choi. Trustsql: Benchmarking text-to-sql reliability with penalty-based scoring. *arXiv preprint arXiv:2403.15879*, 2024.

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*, 2025.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357, 2023.

Terrance Liu, Shuyi Wang, Daniel Preoţiuc-Pietro, Yash Chandarana, and Chirag Gupta. Calibrating llms for text-to-sql parsing by leveraging sub-clause frequencies. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16958–16982, 2025a.

Xinyu Liu, Shuyu Shen, Boyan Li, Nan Tang, and Yuyu Luo. Nl2sql-bugs: A benchmark for detecting semantic errors in nl2sql translation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5662–5673, 2025b.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S Jaakkola, and Regina Barzilay. Conformal language modeling. In *International Conference on Learning Representations*, 2024.

Ashwin Ramachandran and Sunita Sarawagi. Text-to-sql calibration: No need to ask–just rescale model probabilities. *arXiv preprint arXiv:2411.16742*, 2024.

Oleg Somov and Elena Tutubalina. Confidence estimation for error detection in text-to-sql systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25137–25145, 2025.

Elias Stengel-Eskin and Benjamin Van Durme. Calibrated interpretation: Confidence estimation in semantic parsing. *Transactions of the Association for Computational Linguistics*, 11:1213–1231, 2023.

Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. API is enough: Conformal prediction for large language models without logit-access. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 979–995, 2024.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. 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*, pp. 5433–5442, 2023.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR 2023*, 2023. URL https://arxiv.org/abs/2203.11171.

Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics*, 13:529–556, 2025.

Quan Xiao, Debarun Bhattacharjya, Balaji Ganesan, Radu Marinescu, Katsiaryna Mirylenka, Nhan H Pham, Michael Glass, and Junkyu Lee. The consistency hypothesis in uncertainty quantification for large language models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2025.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations*, 2024.

Tao Yu, Rui Zhang, Michihiro Yasunaga, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of EMNLP*, 2018.

Appendix

A Prompts **14**

A.1 Stage 1 Prompt (Generator) 14

A.2 Stage 2 Prompt (Critic) 17

B Full Feature List **20**

C Feature Distributions Across Generator Backbones **20**

C.1 GPT-OSS-120B 21

C.2 Olmo-3-32B-Think 22

C.3 Llama-3.3-Nemotron-Super-49B-v1.5 23

C.4 Qwen2.5-72B-Instruct 24

D Random Forest Training **25**

E Proofs **25**

F Evaluation Metrics **25**

G Additional Experimental Results **26**

G.1 Model Details 26

G.2 Full Results for Four Generator Backbones 27

G.3 Operating Points 27

G.4 Partial Critic Configurations 29

G.5 Seed Variability 29

G.6 Reliability of Individual LLM-Derived Scores 30

G.7 Error Analysis 30

G.8 Zero-Shot Transfer to BIRD 30

G.9 Latency Breakdown. 31

G.10 Published Results of Existing Methods 31

H Applicability Beyond Text-to-SQL **32**

A PROMPTS

A.1 STAGE 1 PROMPT (GENERATOR)

The generator θ_G receives only the question and the schema with sample rows. It validates its query internally but outputs only the query and its confidence c_{gen} (`confidence`), with no validation verdict. Long lines are wrapped for display.

```
You are a Text-to-SQL expert and a strict SQL correctness auditor.

Your task is to generate ONE SQLite-compatible SQL query that answers the
  given
natural-language question using ONLY the provided database schema and
  examples.

You must internally perform both SQL generation and rigorous correctness
  validation, but
your final output MUST contain only the SQL query and its confidence score.

--------------------
INPUTS
--------------------

Natural Language Question:
<QUESTION>

Database Schema with Examples:
<TABLE_SCHEMA>

--------------------
SQL GENERATION AND INTERNAL VALIDATION
--------------------

Generate a SQLite-compatible SQL query that correctly answers the question.

Before assigning confidence, internally verify ALL of the following:

1. QUESTION ALIGNMENT
- The SQL must directly answer the intent of the question.
- Ensure every selected column, aggregation, filter, grouping, ordering, join,
  and limit
  is justified by the question.
- Do not answer a different or narrower interpretation of the question.
- Do not introduce unnecessary joins, filters, aggregations, or conditions.

2. CONTEXT-ONLY GENERATION
- Use ONLY tables, columns, relationships, values, and concepts explicitly
  available in
  the provided schema and examples.
- Do NOT hallucinate tables, columns, relationships, categorical values,
  identifiers,
  dates, entities, or business concepts.
- Do NOT assume information that is not supported by the provided context.
- Do NOT invent literal filter values.
- If a required value is not supported by the question or schema/examples, do
  not
  fabricate it.

3. VALUE AND LITERAL VALIDATION
- Every literal value used in WHERE, HAVING, JOIN, CASE, IN, or other
  conditions must be
  supported by the question or the provided schema/examples.
- Preserve the exact meaning and representation of values supported by the
  context.
```

```
- Do not translate, normalize, or reinterpret categorical values unless the
  provided
  context supports doing so.
- If the question contains a value, use it faithfully unless the schema
  clearly requires
  a supported representation.
- Never introduce a value merely because it seems plausible.

4. SQL CORRECTNESS
- The query must follow valid SQLite syntax.
- Use SQLite-supported functions, operators, expressions, date handling,
  aggregation,
  joins, subqueries, CTEs, and window functions only.
- Do not use database-specific syntax from PostgreSQL, MySQL, SQL Server,
  Oracle, or
  other SQL dialects when it is not valid SQLite.
- Ensure table aliases, column references, GROUP BY, HAVING, ORDER BY, JOIN
  conditions,
  subqueries, and CTEs are syntactically and logically valid.
- Ensure referenced tables and columns actually exist in the provided schema.
- The SQL must be complete and executable.

5. LOGICAL CORRECTNESS
- Check that the SQL computes what the question asks.
- Check joins for correct relationships and avoid accidental Cartesian
  products.
- Check aggregation levels and grouping.
- Check filtering conditions and comparison operators.
- Check ordering and LIMIT semantics where applicable.
- Check NULL handling where relevant.
- Check that the query does not unintentionally duplicate or eliminate
  records.

6. RESULT SANITY
- The query should be capable of producing the type of result requested by the
  question.
- Do not add restrictive conditions merely to make the query return a
  particular result.
- Do not assume that a query is correct simply because it is syntactically
  valid.
- A potentially empty result is not automatically incorrect if the question
  and schema
  legitimately allow it.

7. MINIMAL AND FAITHFUL SQL
- Prefer the simplest SQL that correctly answers the question.
- Do not add unnecessary complexity.
- Do not introduce alternative interpretations when the question has a clear
  meaning.
- Do not use information outside the provided context.

--------------------
CONFIDENCE CALIBRATION
--------------------

Assign a confidence score from 0 to 100 representing your confidence that the
  generated
SQL is BOTH:

1. Correctly aligned with the question, and
2. Fully grounded in the provided context.

Confidence must reflect the result of the complete internal validation, not
  merely
confidence that the SQL is syntactically valid.
```

```
Use these rules:

- 90-100:
  The question is unambiguous, all required schema elements and values are
    explicitly
  supported, the SQL is clearly aligned with the question, and the SQLite
    syntax and
  logic are highly certain.

- 70-89:
  The SQL is likely correct, but there is minor interpretation or
    schema-related
  uncertainty.

- 40-69:
  There is meaningful uncertainty about intent, values, relationships, or SQL
    logic.

- 1-39:
  Significant uncertainty exists, required information is missing, or the SQL
    may
  contain unsupported assumptions.

- 0:
  The SQL cannot be reliably grounded in the provided context or cannot
    correctly answer
  the question.

IMPORTANT:
- A syntactically valid SQL query does NOT justify high confidence.
- If the SQL contains a hallucinated table, column, relationship, value, or
  unsupported
  assumption, confidence MUST be low.
- If the SQL is not clearly aligned with the question, confidence MUST be low.
- If SQLite compatibility is uncertain, confidence MUST be low.
- Confidence must reflect the weakest critical aspect of the generated SQL
  rather than
  averaging independent checks.

--------------------
FINAL OUTPUT
--------------------

Return EXACTLY ONE valid JSON object.

The JSON MUST contain ONLY these two fields:

{
  "sql_query": "<generated SQLite-compatible SQL query>",
  "confidence": <integer between 0 and 100>
}

Do NOT output:
- explanations
- reasoning
- validation results
- evaluation scores
- additional fields
- markdown
- code fences
- comments outside the JSON

The output must be valid JSON and nothing else.
```

A.2 Stage 2 Prompt (Critic)

The critic $\theta_C \neq \theta_G$ receives the question, schema, generated query, and execution result in one call. Step 1 yields v_{sql} (`evaluation_score`) and c_{val} (`evaluation_confidence`). Step 2 yields $\tilde{q}$ (`generated_question`), and Step 3 yields a (`alignment_score`) and s_{sim} (`question_similarity`).

```
You are a strict, conservative, outcome-driven evaluator for Text-to-SQL.

Your task is to determine whether the EXECUTED SQL RESULT correctly and
completely answers the ORIGINAL QUESTION. Do not assume access to a gold
SQL or gold answer.

Inputs:
1. Natural language question
2. Database schema
3. Generated SQL query
4. SQL execution result

Perform the following steps internally and return ONLY the specified JSON.

--------------------------------
STEP 1: STRICT RESULT VALIDATION
--------------------------------

Determine whether the result is a correct and complete answer to the
question. Evaluate the result against the QUESTION, not merely whether
the values appear plausible.

Check all relevant aspects:

1. COLUMN / ROW VALIDITY
- Are the required columns present and relevant?
- Are required rows missing?
- Are extra or invalid rows/columns present?

2. OPERATION / AGGREGATION ALIGNMENT
Verify that the result reflects the operation explicitly requested:
COUNT, SUM, AVG, MIN, MAX, percentage, ratio, rate, difference, total,
most, least, highest, lowest, etc.
A mismatch such as AVG vs SUM or COUNT vs returning individual rows is
incorrect.

3. FILTER / OPERATOR ALIGNMENT
Verify every condition and operator in the question:
>, >=, <, <=, =, !=, BETWEEN, before, after, at least, at most,
exactly, categorical filters, dates, entities, AND/OR, exclusions, etc.
The result must respect the exact semantics and boundaries.

4. ORDER / LIMIT / TOP-K ALIGNMENT
If the question specifies top/bottom, first/latest/oldest, ranking,
sorting, or a number of results, verify both:
- correct ordering/ranking criterion
- correct number of returned records
For example, "top 5 by revenue" requires the five highest-revenue
records, not merely any five records.

5. GROUPING / GRANULARITY
Verify that the result has the requested level of aggregation:
e.g. overall total vs total by company, average by year, results for
each sector, etc.

6. DATE / TIME ALIGNMENT
Verify requested years, dates, periods, before/after semantics, ranges,
```

```
latest/earliest requirements, and other temporal constraints.

7. VALUE / RESULT CONSISTENCY
Verify that returned scalar, aggregate, ranking, and derived values are
consistent with the question.

8. COMPLETENESS / EMPTY RESULTS
Verify that all parts of the question are answered. Empty, NULL, or zero
results are not automatically wrong; determine whether they are
consistent with the question and applied conditions.

If ANY substantive requirement is violated, the result is incorrect.

evaluation_score:
- 1 = result is correct AND complete.
- 0 = result is incorrect, incomplete, or violates any substantive
      requirement.

evaluation_confidence:
Confidence (0-100) in the above binary evaluation.
- 90-100: clear evidence and no meaningful ambiguity
- 70-89: strong evidence with minor uncertainty
- 40-69: meaningful uncertainty
- 1-39: substantial uncertainty
- 0: cannot reliably determine correctness

A clearly violated explicit requirement (aggregation, operator, filter,
ordering, LIMIT, grouping, date, etc.) should produce high confidence
that evaluation_score = 0.

--------------------------------
STEP 2: REVERSE QUESTION GENERATION
--------------------------------

Ignore the original question.

Using ONLY the SQL RESULT and schema, generate the most precise question
that the result actually answers.

Do not repair, idealize, or infer missing filters, aggregations,
conditions, ordering, or other requirements.

--------------------------------
STEP 3: ALIGNMENT AND SIMILARITY
--------------------------------

Compare the generated question with the original question.

alignment_score (0-100) measures how well the RESULT itself answers the
original question, considering:
- requested information
- filters and operators
- aggregation
- ordering
- LIMIT/top-k
- grouping/granularity
- dates/time
- completeness

question_similarity (0-100) measures semantic similarity between the
ORIGINAL QUESTION and the GENERATED QUESTION.

Important:
- A result can be topically related but operationally incorrect.
- Do not give high alignment merely because the returned data is related.
```

```
- If evaluation_score = 0, alignment_score should normally be <= 30
  for a substantive correctness failure.
- If the result only partially answers the question, alignment_score
  should normally be <= 50.
- If evaluation_score = 0, question_similarity MUST NOT exceed 60.

---------------------------------
OUTPUT
---------------------------------

Return EXACTLY ONE valid JSON object with these five fields:

{
  "evaluation_score": <0 or 1>,
  "evaluation_confidence": <integer between 0 and 100>,
  "alignment_score": <number between 0 and 100>,
  "generated_question": "<generated question>",
  "question_similarity": <number between 0 and 100>
}

No markdown.
No explanations.
No reasoning.
No additional fields.

---------------------------------
INPUT
---------------------------------

Original Question:
<INSERT_QUESTION_HERE>

Database Schema:
<INSERT_DB_SCHEMA_HERE>

Generated SQL Query:
<INSERT_SQL_QUERY_HERE>

SQL Result:
<INSERT_RECORDS_HERE>
```

B Full Feature List

Table 7 lists all 41 features of $\mathbf{x}$ with the identifiers used in our code. All token statistics use the generator's probabilities π_t of the tokens of $\hat{s}$, from the first SELECT token to the terminating semicolon. *Keyword* tokens are upper-case SQL keywords, and *non-keyword* tokens are the rest (identifiers, literals, operators). The family column gives the grouping used in the ablation (Table 5).

Table 7: All 41 features. Core features appear in the main text (Table 1) and in the density plots of Appendix C.

Family	Symbol	Code name	Definition	Core
Verbalized	c_{gen}	`Generation_conf.`	Generator's verbalized confidence, 0–100 (Stage 1)	✓
	v_{sql}	`Evaluation_Score`	Critic's binary verdict on the result (Stage 2)	✓
	c_{val}	`Evaluation_Conf.`	Critic's confidence in its verdict, 0–100 (Stage 2)	✓
Alignment	a	`alignment_score`	How well the result answers q, 0–100 (Stage 2)	✓
	s_{sim}	`question_similarity`	Similarity of q and reconstructed $\tilde{q}$, 0–100 (Stage 2)	✓
Logit (P_{feats})	μ_{geom}	`geo_sql_all`	Geometric mean of π_t, all tokens	✓
	FTC	`ftc_sql_all`	Product of π_t, all tokens	✓
	μ_{arith}	`mean_sql_all`	Arithmetic mean of π_t, all tokens	✓
	$\pi_{\min}$	`min_sql_all`	Minimum π_t, all tokens	✓
	H	`entrop_sql_all`	$-\sum_t \pi_t \log \pi_t$, all tokens	✓
	σ^2	`sql_token_variance`	Variance of π_t	✓
	ℓ_{CLM}	`clm`	$-\sum_t \log \pi_t$ ($= -\log \text{FTC}$)	✓
		`*_sql_clause`	`geo`, `ftc`, `mean`, `min`, `entrop` over keyword tokens (5 features)	
		`*_non_sql`	The same five over non-keyword tokens (5 features)	
		`min`	Minimum π_t after dropping whitespace tokens (near-duplicate of $\pi_{\min}$)	
		`sql_token_margin`	Probability range $\max_t \pi_t - \min_t \pi_t$	
		`bin_1`–`bin_13`	Histogram over 13 probability bins, $[0, 0.3)$, $[0.3, 0.4)$, $[0.4, 0.5)$, then width-0.05 bins to 1. Each value is the mean of the min–max-normalised probabilities of the tokens in the bin (0 if empty) (13 features)	
SQL-structural (S_{feats})	SAC_{geom}	`sac`	Geometric mean over clause groups of the within-group mean π_t (Eq. 11)	✓
	SLC	`slc`	Geometric mean of π_t over schema-element tokens (Eq. 11)	✓
		`schema_geo_mean`	Geometric mean of π_t over tokens matching schema table names	
		`c3s`	Weakest clause geometric mean $\times\, e^{-\text{Var}_{\text{inter}}} \times e^{-\overline{\text{Var}}_{\text{intra}}}$ over clause groups	
Total: 5 LLM-derived + 32 logit + 4 structural				**41** (14 core)

Some features are constant in some configurations and are then never used by the forest. For example, `bin_1` (probabilities below 0.3) is always empty for generator GPT-OSS-120B.

C Feature Distributions Across Generator Backbones

Figures 2–5 show kernel density estimates for 16 features per generator backbone: the 14 core features of Table 7, and FTC and the minimum probability computed on non-keyword tokens (`ftc_non_sql`, `min_non_sql`). The two non-keyword variants are included because identifiers and literals are where schema-grounding errors appear. We omit the other 25 features for readability. They recompute a core statistic on a token subset, are histogram bins, are the near-duplicate `min` or the range `sql_token_margin`, or are the structural variants `schema_geo_mean` and `c3s`. In every panel, correct queries (Z=1) are shown in orange and incorrect queries (Z=0) in blue. Densities are normalised jointly, so each curve's area is proportional to its class size. The script `plot_feature_kde.py` reproduces the figures from the saved feature tables.

C.1 GPT-OSS-120B

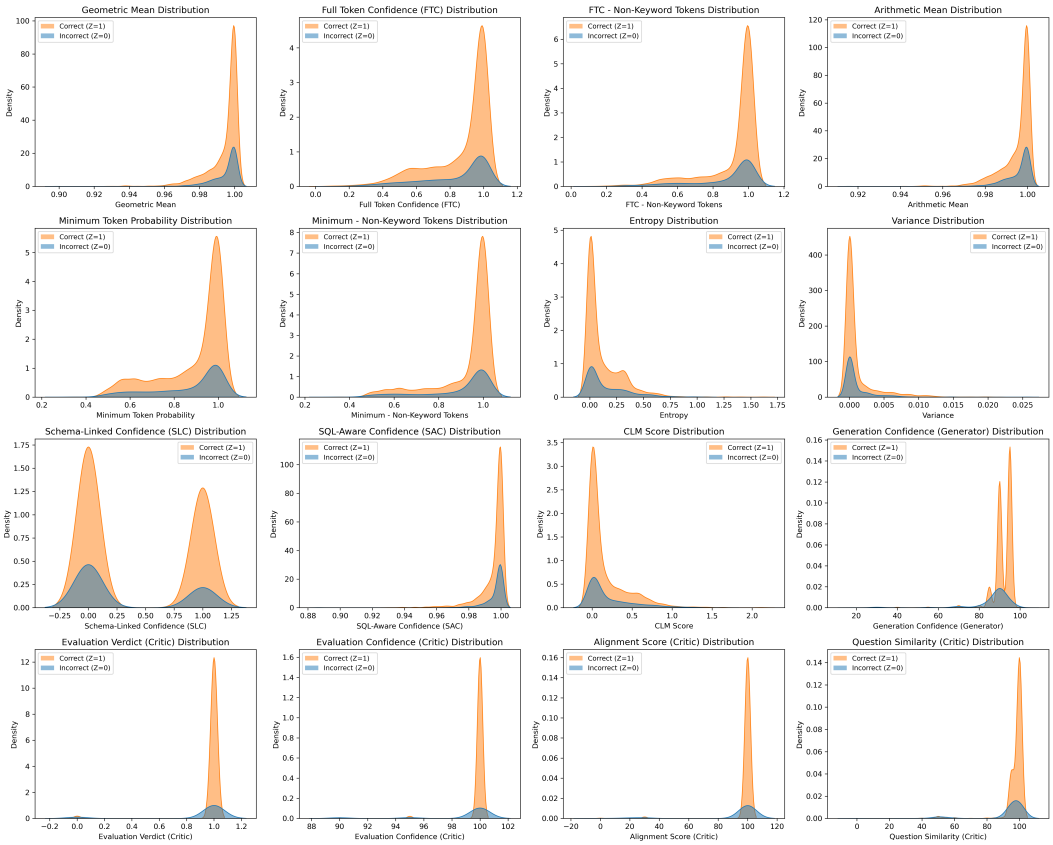

Figure 2: Kernel density estimates of the 14 core features and two non-keyword variants for generator GPT-OSS-120B with the Gemma-4-31B critic, separated by execution outcome (orange: correct, blue: incorrect). Feature selection is described at the start of Appendix C.

C.2 OLMO-3-32B-THINK

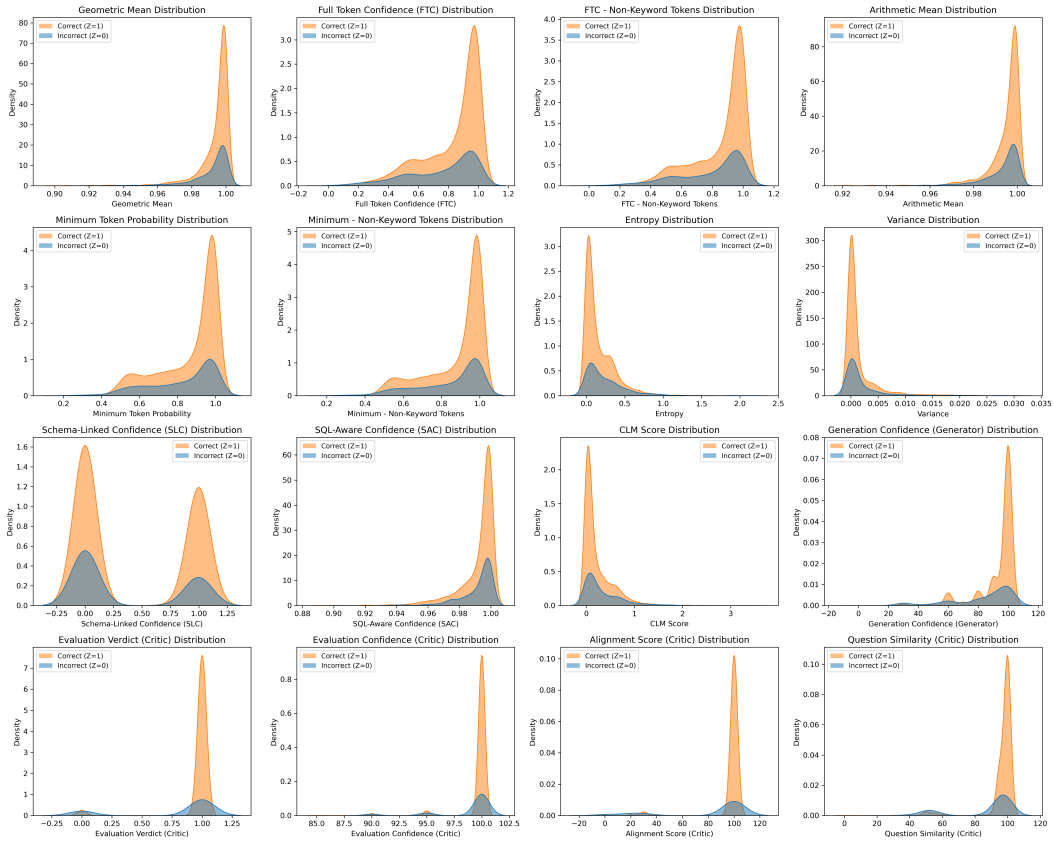

Figure 3: Kernel density estimates of the 14 core features and two non-keyword variants for generator Olmo-3-32B-Think with the Gemma-4-31B critic, separated by execution outcome (orange: correct, blue: incorrect). Feature selection is described at the start of Appendix C.

C.3 LLAMA-3.3-NEMOTRON-SUPER-49B-V1.5

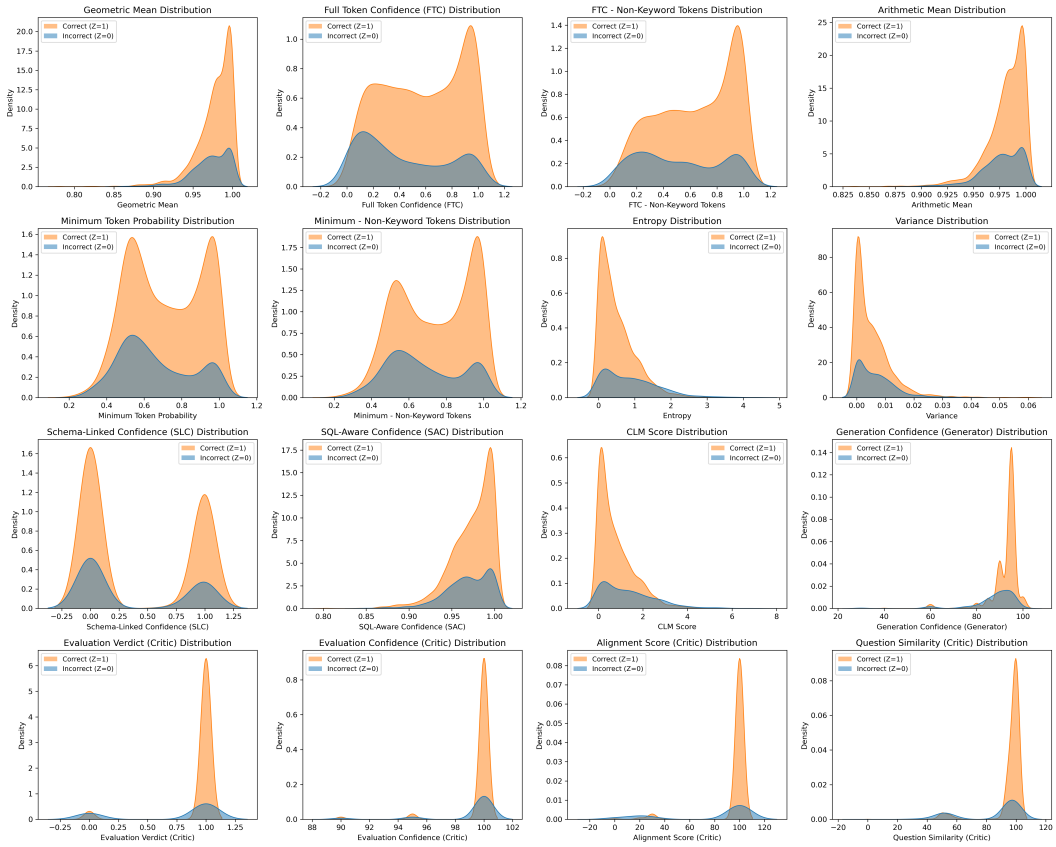

Figure 4: Kernel density estimates of the 14 core features and two non-keyword variants for generator Llama-3.3-Nemotron-Super-49B-v1.5 with the Gemma-4-31B critic, separated by execution outcome (orange: correct, blue: incorrect). Feature selection is described at the start of Appendix C.

C.4 QWEN2.5-72B-INSTRUCT

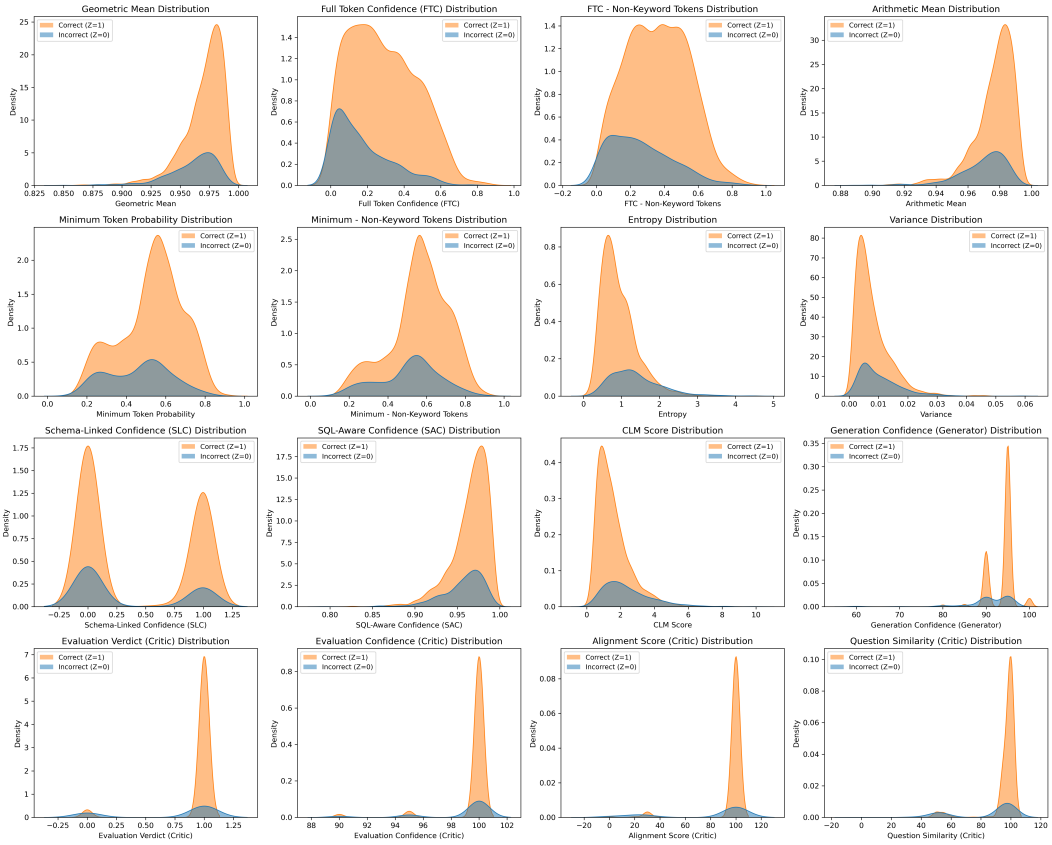

Figure 5: Kernel density estimates of the 14 core features and two non-keyword variants for generator Qwen2.5-72B-Instruct with the Gemma-4-31B critic, separated by execution outcome (orange: correct, blue: incorrect). Feature selection is described at the start of Appendix C.

D Random Forest Training

Hyperparameters. For each configuration, hyperparameters are chosen by randomized search over 40 configurations with 5-fold cross-validation on the training split, maximising ROC-AUC. Bootstrap sampling and `class_weight=balanced_subsample` are always used, and out-of-bag estimation is enabled. Table 8 lists the selected values.

Table 8: Selected Random Forest hyperparameters per configuration. *Generation run of the four-backbone study (Section 5.1).

Configuration	Trees	Max depth	Min split	Min leaf	Max features
Generator GPT-OSS-120B, critic Gemma-4-31B	400	10	5	10	`sqrt`
Generator GPT-OSS-120B, critic Qwen3.8-27B	400	10	5	10	`sqrt`
Generator GPT-OSS-120B, critic DeepSeek-V4.1-Flash	400	10	5	10	`sqrt`
Generator GPT-OSS-120B, critic Nemotron-3-Super	200	50	10	10	all
Generator GPT-OSS-120B*, critic Gemma-4-31B	800	10	10	10	all
Generator Olmo-3-32B, critic Gemma-4-31B	400	15	5	10	`log2`
Generator Qwen2.5-72B, critic Gemma-4-31B	200	20	2	1	`log2`
Generator Llama-49B, critic Gemma-4-31B	400	10	5	10	`sqrt`

Per-configuration training. A separate forest is trained for each configuration. Its training set contains the feature vectors computed from that configuration's outputs on the Spider training split, with labels Z_i obtained by execution matching against gold SQL. We train per backbone because logit-derived features (P_{feats}, S_{feats}) have backbone-specific distributions. Token probability ranges, entropy scales, and SAC values differ across model families and sizes (Figures 2–5). For zero-shot transfer (Section 5.6), the GPT-OSS-120B* forest and calibration scores (Table 8) are applied to GPT-OSS-120B outputs on BIRD. For NL2SQL-BUGs, whose SQL is given, a forest trained on the four critic scores is used. No target-domain labels are used at any stage.

Conformal calibration. Calibration scores $\eta_i = -\log p_i$ are computed on calibration queries with $Z_i = 1$ only. At test time, Eq. 6 is evaluated with ties counted in the numerator ($\eta_i \geq \eta$). Counting ties this way makes $\hat{c}$ larger and the FAR bound conservative. Using $1 - p$ instead of $-\log p$ as the nonconformity score gives the same $\hat{c}$, because the two are monotone transforms of each other.

E Proofs

Proposition 1. Let $\eta_1, \ldots, \eta_{m_1}$ be the scores of the correct calibration queries, and η_{m_1+1} the score of a correct test query. Under exchangeability, conditional on the trained estimator p, the rank of η_{m_1+1} among the $m_1 + 1$ scores is uniform on $\{1, \ldots, m_1 + 1\}$. Ties are counted in the numerator of Eq. 6, which can only increase $\hat{c}$. Hence $\hat{c} = (1 + |\{i \leq m_1 : \eta_i \geq \eta_{m_1+1}\}|)/(m_1 + 1)$ satisfies $\mathbb{P}(\hat{c} \leq t \mid Z = 1) \leq \lfloor t(m_1 + 1) \rfloor / (m_1 + 1) \leq t$. Taking $t = 1 - \alpha$ gives $\text{FAR}(1 - \alpha) = \mathbb{P}(\hat{c} < 1 - \alpha \mid Z = 1) \leq 1 - \alpha$ (Angelopoulos & Bates, 2023, Prop. 3).

Proposition 2. Fix $\lambda \in \Lambda$. Since p is trained on data disjoint from $\mathcal{C}$ and the calibration queries are i.i.d., the queries with $p_i \geq \lambda$ are, conditional on their number $n(\lambda)$, i.i.d. draws from the distribution of queries with $p \geq \lambda$. Their error count therefore satisfies $k(\lambda) \mid n(\lambda) \sim \text{Bin}(n(\lambda), \text{Risk}(\lambda))$. Under $H_\lambda : \text{Risk}(\lambda) > \varepsilon$, binomial stochastic ordering gives $\mathbb{P}(u_\lambda \leq s) \leq s$ for all s, so u_λ is a valid p-value. The rule certifies only thresholds with $u_\lambda < \delta/|\Lambda|$. By the union bound, the probability of certifying any λ with $\text{Risk}(\lambda) > \varepsilon$ is at most δ. On the complementary event, $\hat{\lambda}$ satisfies $\text{Risk}(\hat{\lambda}) \leq \varepsilon$, or no threshold is certified and the system abstains on every query. This is the Learn then Test guarantee (Angelopoulos et al., 2025) with the Bonferroni correction (Angelopoulos & Bates, 2023, Thm. A.1).

F Evaluation Metrics

All metrics are computed on the test split. For selective metrics, a query is *accepted* if its score passes the rule's threshold ($\hat{c} \geq 1 - \alpha$ for the conformal rule, $p \geq \hat{\lambda}$ for the risk-controlled rule) and

abstained otherwise. We treat acceptance as the positive prediction and write TP (correct, accepted), FP (incorrect, accepted), FN (correct, abstained), and TN (incorrect, abstained).

Table 9: Evaluation metrics. Risk is bounded by Proposition 2 (risk-controlled rule), and FAR by Proposition 1 (conformal rule).

Metric	**Definition**	**Better**
ECE	Weighted mean gap between confidence and empirical accuracy over 10 equal-width bins.	↓
ACE	As ECE, with equal-mass bins.	↓
Brier	Mean squared error between confidence and Z.	↓
AUROC	Probability that a random correct query receives a higher score than a random incorrect one.	↑
AUARC	Area under the accuracy–rejection curve: accuracy of the retained queries as the rejection rate varies from 0 to 1. Primary metric.	↑
Prec./Recall/F1 (Table 11)	Classification of $Z{=}1$ by the forest at $p \geq 0.5$.	↑
Error-detection Prec./Recall/F1 (BIRD, NL2SQL-BUGs)	Classification of $Z{=}0$ by flagging queries with $\hat{c} < 1 - \alpha$.	↑
Coverage	Fraction of queries accepted, $(\mathrm{TP} + \mathrm{FP})/n$. Empirical.	—
Risk	Error rate among accepted queries, $\mathrm{FP}/(\mathrm{TP}{+}\mathrm{FP})$. Bounded by ε with probability $1 - \delta$ for the risk-controlled rule under i.i.d. data (Proposition 2).	↓
CondAcc	Accuracy among accepted queries, $1 - \mathrm{Risk} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$. Identical to SSC($\lvert S\rvert{=}1$). Empirical.	↑
TAR	True abstention rate, $\mathrm{TN}/(\mathrm{TN} + \mathrm{FP})$: fraction of incorrect queries withheld. Empirical.	↑
FAR	False abstention rate, $\mathrm{FN}/(\mathrm{FN}{+}\mathrm{TP})$: fraction of correct queries withheld. Bounded by $1 - \alpha$ in expectation under exchangeability.	↓
SSC($\lvert S\rvert{=}0$)	Accuracy among abstained queries, $\mathrm{FN}/(\mathrm{FN} + \mathrm{TN})$.	—

G Additional Experimental Results

G.1 Model Details

Table 10: Language models used in our experiments. The first four serve as generators. Gemma-4-31B-it is the default critic, and Qwen3.8-27B, DeepSeek-V4.1-Flash, and Nemotron-3-Super are additional critics. Qwen2.5-72B-Instruct also serves as a critic. The main configuration pairs generator GPT-OSS-120B with critic Gemma-4-31B-it.

Model	Params (B)	HuggingFace model ID
GPT-OSS-120B	120	`openai/gpt-oss-120b`
Olmo-3-32B-Think	32	`allenai/Olmo-3-32B-Think`
Llama-3.3-Nemotron-Super-49B-v1.5	49	`nvidia/Llama-3_3-Nemotron-Super-49B-v1_5`
Qwen2.5-72B-Instruct	72	`Qwen/Qwen2.5-72B-Instruct`
Gemma-4-31B-it (critic)	31	`google/gemma-4-31B-it`
Qwen3.8-27B (critic)	27	`Qwen/Qwen3.8-27B`
DeepSeek-V4.1-Flash (critic)	—	`deepseek-ai/DeepSeek-V4.1-Flash`
Nemotron-3-Super (critic)	120	`nvidia/NVIDIA-Nemotron-3-Super-120B-A12B-BF16`

G.2 Full Results for Four Generator Backbones

Table 11 reports the four generator backbones, each with the Gemma-4-31B critic. The confusion matrices at α=0.1 reproduce the reported TAR, FAR, and CondAcc. FAR and CondAcc use different denominators. For Olmo-3-32B, FAR is computed over the 470 correct queries (416 withheld, 0.885), whereas CondAcc is computed over the 56 accepted queries (54 correct, 0.964).

Table 11: Four generator backbones with the Gemma-4-31B critic on the Spider test split. Selective metrics use the conformal rule at α=0.1. The GPT-OSS-120B column uses the generation run of the four-backbone study, not the main run of Table 2.

Metric	**GPT-OSS-120B**	**Olmo-3-32B**	**Qwen2.5-72B**	**Llama-49B**
Correct / incorrect queries	499 / 137	470 / 166	503 / 133	475 / 160
Base accuracy ρ	0.785	0.739	0.791	0.748
ECE	0.133	0.126	0.045	0.113
ACE	0.133	0.125	0.041	0.112
Brier	0.163	0.158	0.137	0.153
AUROC	0.791	0.816	0.765	0.814
AUARC	0.918	0.895	0.898	0.902
Precision (Z=1, $p \geq 0.5$)	0.889	0.847	0.815	0.854
Recall (Z=1, $p \geq 0.5$)	0.804	0.851	0.974	0.848
F1 (Z=1, $p \geq 0.5$)	0.844	0.849	0.888	0.851
TP / FP / FN / TN	51 / 1 / 448 / 136	54 / 2 / 416 / 164	50 / 3 / 453 / 130	52 / 1 / 423 / 159
Coverage	0.082	0.088	0.083	0.083
Risk	0.019	0.036	0.057	0.019
CondAcc = 1−Risk	0.981	0.964	0.943	0.981
TAR	0.993	0.988	0.977	0.994
FAR	0.898	0.885	0.901	0.891
SSC ($\lvert S\rvert$=0)	0.767	0.717	0.777	0.727

G.3 Operating Points

Risk-controlled rule on the reported split. Table 12 applies the risk-controlled rule to the split behind Tables 2 and 11. On this split, every certified threshold keeps the test risk below ε. Critic rows use generator GPT-OSS-120B; generator rows use the Gemma critic and the generation runs of the four-backbone study.

Table 12: Risk-controlled rule (δ=0.1) on the reported Spider split. $\hat{\lambda}$: certified threshold on p ("—": none certified, so the system abstains on every query). Cov. and Risk in %.

	$\varepsilon = 0.10$			$\varepsilon = 0.15$			$\varepsilon = 0.20$		
	$\hat{\lambda}$	Cov.	Risk	$\hat{\lambda}$	Cov.	Risk	$\hat{\lambda}$	Cov.	Risk
Critic: Gemma-4-31B	—	0.0	—	0.58	54.4	8.4	0.44	76.4	13.4
Critic: Qwen3.8-27B	0.68	44.3	5.7	0.50	67.5	9.6	0.40	79.7	12.4
Critic: DeepSeek-V4.1-Flash	0.77	29.4	2.7	0.59	54.7	8.9	0.41	81.0	14.4
Critic: Nemotron-3-Super	—	0.0	—	0.65	47.3	7.3	0.46	71.1	13.1
Gen.: GPT-OSS-120B	0.91	20.1	0.8	0.67	47.6	5.9	0.56	61.6	8.7
Gen.: Olmo-3-32B	—	0.0	—	0.77	26.4	3.6	0.51	73.0	14.7
Gen.: Qwen2.5-72B	0.87	42.3	8.6	0.74	72.5	10.8	0.60	89.6	16.5
Gen.: Llama-49B	—	0.0	—	0.67	50.9	9.3	0.57	65.8	11.7

Conformal rule. Table 13 sweeps the operating level α of the conformal rule. By Eq. 5, α fixes the accepted fraction of correct queries at about α, and the estimator determines how many incorrect queries are accepted alongside them ($1 - \mathrm{TAR}$). With the Gemma critic, coverage rises from 7.9% to 100%. Risk stays between 5.0% and 6.8% up to α=0.6 and then rises to the no-abstention error of 22.6%. On this single split, FAR exceeds its bound by up to 0.049 (at α=0.5). All levels share one calibration draw, so their deviations are correlated. The bound holds in expectation over draws.

The 10-seed mean FAR at α=0.1 is 0.907 ± 0.016, within one standard deviation of the bound (Appendix G.5).

Table 13: Coverage–risk trade-off of the conformal rule (Proposition 1) on the Spider test split. Critic rows use generator GPT-OSS-120B. The last rows average the four generator backbones of Table 11 (Gemma critic).

	α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
	FAR bound $1-\alpha$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0
Critic: Gemma-4-31B	FAR	.904	.813	.728	.640	.549	.445	.319	.181	.075	.000
	Coverage (%)	7.9	15.3	22.2	29.6	37.3	46.1	58.2	72.3	85.7	100
	Risk (%)	6.0	5.2	5.0	5.9	6.3	6.8	9.5	12.4	16.5	22.6
Critic: Qwen3.8-27B	Coverage (%)	6.3	14.2	21.2	26.9	35.4	46.7	58.3	69.2	80.0	100
	Risk (%)	0.0	3.3	2.2	2.3	4.0	5.7	6.7	9.3	12.6	22.6
Four generators (average)	Coverage (%)	8.4	15.8	23.4	31.7	39.8	51.2	61.9	72.9	84.2	100
	Risk (%)	3.3	3.3	4.1	4.7	6.3	8.8	10.5	13.1	16.9	23.4

Table 14 gives the confusion matrices behind the critic rows of Table 13, computed from the test-split predictions of each configuration (492 correct and 144 incorrect queries).

Table 14: Accept/abstain confusion matrices across α (generator GPT-OSS-120B, Spider test split, n=636). Positive = accepted.

	Critic: Gemma-4-31B						Critic: Qwen3.8-27B					
α	TP	FP	FN	TN	TAR	FAR	TP	FP	FN	TN	TAR	FAR
0.1	47	3	445	141	.979	.904	40	0	452	144	1.000	.919
0.2	92	5	400	139	.965	.813	87	3	405	141	.979	.823
0.3	134	7	358	137	.951	.728	132	3	360	141	.979	.732
0.4	177	11	315	133	.924	.640	167	4	325	140	.972	.661
0.5	222	15	270	129	.896	.549	216	9	276	135	.938	.561
0.6	273	20	219	124	.861	.445	280	17	212	127	.882	.431
0.7	335	35	157	109	.757	.319	346	25	146	119	.826	.297
0.8	403	57	89	87	.604	.181	399	41	93	103	.715	.189
0.9	455	90	37	54	.375	.075	445	64	47	80	.556	.096
1.0	492	144	0	0	.000	.000	492	144	0	0	.000	.000

Generator Olmo-3-32B. Table 15 gives the confusion matrix at each operating level for generator Olmo-3-32B with the Gemma critic. Recall of correct queries, $1-\mathrm{FAR}$, stays close to α, as Proposition 1 implies, and Risk rises from 3.6% to 26.1% as α grows. Figure 6 plots the corresponding curves for the main configuration.

Table 15: Accept/abstain confusion matrices across α for generator Olmo-3-32B-Think with the Gemma critic (Spider test split, n=636). Positive = accepted.

α	TP	FP	FN	TN	CondAcc	Recall = 1−FAR	Risk
0.1	54	2	416	164	96.4%	11.5%	3.6%
0.2	90	4	380	162	95.7%	19.1%	4.3%
0.3	136	6	334	160	95.8%	28.9%	4.2%
0.4	185	9	285	157	95.4%	39.4%	4.6%
0.5	239	18	231	148	93.0%	50.9%	7.0%
0.6	293	27	177	139	91.6%	62.3%	8.4%
0.7	347	43	123	123	89.0%	73.8%	11.0%
0.8	395	68	75	98	85.3%	84.0%	14.7%
0.9	435	99	35	67	81.5%	92.6%	18.5%
1.0	470	166	0	0	73.9%	100.0%	26.1%

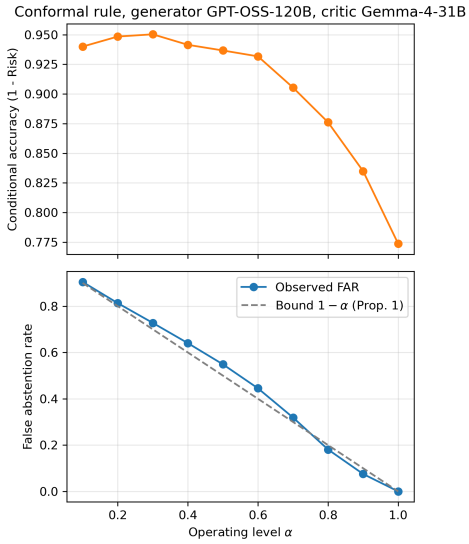

Figure 6: Conditional accuracy (top) and FAR (bottom) of the conformal rule across operating levels α (generator GPT-OSS-120B, critic Gemma-4-31B). The dashed line is the FAR bound $1 - \alpha$ of Proposition 1.

G.4 Partial Critic Configurations

In a *partial* configuration, the generator supplies its own validation verdict and confidence, and the other model supplies only the alignment and similarity scores. Table 16 compares these with self-critique and with the full Qwen2.5-72B critic. AUARC stays between 0.905 and 0.915 across all four.

Table 16: Degrees of generator–critic separation (generator GPT-OSS-120B, Spider, α=0.1).

	Self-critique	Gemma partial	Qwen partial	Qwen critic
AUROC	0.781	0.782	0.821	0.826
AUARC	0.913	0.905	0.913	0.915
ECE	0.149	0.129	0.154	0.166
TAR	0.993	1.000	0.993	0.993
FAR	0.904	0.896	0.902	0.896

G.5 Seed Variability

For each configuration, we resample the stratified train/calibration/test split with seeds 0–9. We refit the forest with hyperparameters fixed to Table 8 and recompute the calibration scores.

Table 17: Mean $\pm$ standard deviation over 10 resampled splits (α=0.1). Critic rows use generator GPT-OSS-120B; generator rows use the Gemma critic and the generation runs of the four-backbone study. The low ECE for generator Qwen2.5-72B reflects its selected forest (Table 8).

Configuration	AUROC	AUARC	ECE	TAR	FAR
Critic: Gemma-4-31B	0.772 ± 0.020	0.900 ± 0.010	0.151 ± 0.007	0.988 ± 0.012	0.907 ± 0.016
Critic: Qwen3.8-27B	0.811 ± 0.019	0.914 ± 0.008	0.136 ± 0.009	0.993 ± 0.006	0.902 ± 0.012
Critic: DeepSeek-V4.1-Flash	0.793 ± 0.021	0.907 ± 0.009	0.145 ± 0.009	0.988 ± 0.006	0.905 ± 0.016
Critic: Nemotron-3-Super	0.766 ± 0.023	0.898 ± 0.011	0.125 ± 0.011	0.984 ± 0.011	0.905 ± 0.018
Generator: GPT-OSS-120B	0.772 ± 0.023	0.909 ± 0.010	0.133 ± 0.011	0.992 ± 0.007	0.899 ± 0.015
Generator: Olmo-3-32B	0.790 ± 0.016	0.885 ± 0.008	0.128 ± 0.006	0.990 ± 0.006	0.889 ± 0.020
Generator: Qwen2.5-72B	0.785 ± 0.015	0.911 ± 0.007	0.032 ± 0.009	0.983 ± 0.012	0.895 ± 0.016
Generator: Llama-49B	0.806 ± 0.012	0.898 ± 0.007	0.131 ± 0.007	0.990 ± 0.008	0.896 ± 0.021

G.6 Reliability of Individual LLM-Derived Scores

Table 18 reports the standalone AUROC of each LLM-derived score, before fusion. The first four columns are generator backbones with the Gemma critic (generation runs of the four-backbone study, so c_{gen} for GPT-OSS differs from the last four columns). The last four use generator GPT-OSS-120B and name the critic. In every configuration, c_{gen} comes from the generator and the other four scores come from the critic. All 40 values exceed 0.5 (range 0.525–0.771).

Table 18: Standalone AUROC of LLM-derived scores on the Spider test split.

	Generator (critic Gemma-4-31B)				Critic (generator GPT-OSS-120B)			
Score	GPT-OSS	Olmo	Qwen2.5	Llama	Gemma	Qwen3.8	DeepSeek	Nemotron
c_{gen} (generation confidence)	0.739	0.697	0.658	0.704	0.715	0.715	0.715	0.715
v_{sql} (evaluation verdict)	0.537	0.607	0.576	0.605	0.563	0.567	0.588	0.596
c_{val} (evaluation confidence)	0.567	0.569	0.554	0.525	0.580	0.771	0.700	0.572
a (alignment)	0.537	0.608	0.579	0.606	0.564	0.753	0.651	0.611
s_{sim} (question similarity)	0.652	0.721	0.665	0.704	0.651	0.737	0.710	0.595

G.7 Error Analysis

The feature distributions (Appendix C) show where individual signals fail. Logit features overlap heavily between correct and incorrect queries, because a wrong query is usually still fluent token by token. The error is semantic, not a decoding hesitation. The critic's verdict and alignment score peak at their "looks correct" value for most correct queries, and most incorrect queries share that peak.

We examined the 144 incorrect test queries of the main configuration, where GPT-OSS-120B generates and Gemma-4-31B critiques. We call an error *recognized* if the critic's verdict v_{sql} and its alignment score a both flag it ($v_{\text{sql}} = 0$ and $a < 50$; 20 queries), and *not recognized* if neither does (124 queries). The critic therefore misreads most errors. Recognized errors receive a mean forest posterior of $p = 0.34$, and at most one is accepted at any operating level up to α=0.5. Unrecognized errors receive a higher mean posterior ($p = 0.47$), yet only 3 of 124 are accepted at α=0.1, because the generator-side features still separate them from most correct queries. As α increases to 0.5, up to 14 of 124 (11%) are accepted. Errors that the critic misreads are thus the main source of accepted errors at larger α.

G.8 Zero-Shot Transfer to BIRD

Table 19: Zero-shot transfer to the BIRD dev set (1,534 queries, generator GPT-OSS-120B, critic Gemma-4-31B, Spider-trained forest and calibration, no retraining or recalibration). Precision, recall, and F1 are for error detection ($\hat{c} < 0.9$). AUARC ranges from 0.343 (random) to 0.709 (perfect ranking) at this base accuracy.

Metric	**Value**
Base accuracy	0.342
ECE	0.091
ACE	0.090
Brier	0.202
AUROC	0.724
AUARC	0.515
Error-detection precision	0.662
Error-detection recall	0.998
Error-detection F1	0.796
CondAcc (α=0.1)	0.833
Risk at Spider-certified $\hat{\lambda}$ (ε=0.15)	0.301

Why ECE is lower on BIRD than on Spider. The lower ECE does not mean better calibration out of domain. It is a base-rate effect (Table 20). On Spider, base accuracy is high, so even low-

confidence predictions are often correct, and empirical accuracy exceeds confidence in every bin. On BIRD, most predictions have low confidence, and in that region accuracy is close to mean confidence, so it contributes little to ECE. Conformal calibration explains neither result. It concerns the threshold, not the binned agreement that ECE measures, and its guarantee does not extend to BIRD, whose queries are not exchangeable with the Spider calibration set.

Table 20: Base-rate comparison behind the ECE difference (generator GPT-OSS-120B, critic Gemma-4-31B).

	Spider (in-domain)	BIRD (zero-shot)
Error rate without abstention	21.5%	65.8%
Share of predictions with $p \leq 0.5$	29%	72%
Accuracy vs. mean confidence there	53% vs. 36%	24% vs. 35%
ECE	0.133	0.091

G.9 Latency Breakdown

Table 21: Per-query wall-clock latency (seconds) of the main configuration. The generator, critic and local stages were timed one query at a time on 40 test queries. Stage 3 can run in parallel with Stage 2.

Stage	**Operation**	**Mean time (s)**
1	SQL generation (GPT-OSS-120B) + execution	1.9 - 2.6
2	Validation + alignment (Gemma-4-31B critic)	0.5 - 1.7
3–4	Features, forest, conformal score (local)	0.03
Total	per query	**2.43 - 4.33**

The generator dominates the cost, because of the query execution. Both LLM calls ran through a shared inference gateway with reasoning enabled. Feature extraction, forest training and inference, and conformal calibration ran on a MacBook M4 Pro without a GPU.

G.10 Published Results of Existing Methods

Table 22 lists results reported in the original papers on Spider. These use different backbones, splits, and evaluation protocols, so they are *not* directly comparable with ours. Three of the four report AUROC but not AUARC, so AUROC is the closest common metric. Table 4 gives the controlled comparison.

Table 22: Reported results of existing confidence estimation methods on Spider, each on its own backbone and protocol.

Method	**Backbone**	**Approach**	**Reported results**
SAC-Product (Entezari Maleki et al., 2026)	GPT-4o-mini	Logit-based	AUROC 79.9, ECE 0.14
Geometric Aggregation (Xiao et al., 2025)	CodeLlama-34B	Black-box	AUROC 76.0, AUARC 35.0
Multivariate Platt Scaling (Liu et al., 2025a)	Llama-3.1-8B	Logit + sampling	AUROC 74.8, Brier 0.163, ECE 0.024, ACE 0.026
Pooled token-level (Ramachandran & Sarawagi, 2024)	Codestral-22B	Logit-based	AUROC 78.8
Ours	Olmo-3-32B-Think + Gemma-4-31B critic	Combined	AUARC 89.5, AUROC 81.6, ECE 0.126, ACE 0.125, Brier 0.158

H Applicability Beyond Text-to-SQL

Parts of the pipeline do not depend on SQL. These are the generator–critic separation, the critic's validation step, the token-level features, the forest fusion, and the conformal calibration. They apply to any task that produces an output with token probabilities and has labeled data on which correctness can be computed. Stage 2's alignment step and the structural features are SQL-specific. The alignment step needs a concrete result, a way to generate the question the result answers, and a check of that result. Code generation (with unit tests as the execution analogue) and mathematical reasoning (with a solver) have such analogues. Open-domain QA does not, and Stage 2 would need a replacement such as an attribution check of whether retrieved evidence supports the answer. We have not evaluated these extensions.