Reliable End-to-End Text-to-SQL Generation

Kaiwen Chen
University Of Toronto
Toronto, Ontario, Canada
kckevinchen@cs.toronto.edu

Yueting Chen
Seattle University
Seattle, Washington, United States
ychen24@seattleu.edu

Nick Koudas
University Of Toronto
Toronto, Ontario, Canada
koudas@cs.toronto.edu

Xiaohui Yu
York University
Toronto, Ontario, Canada
xhyu@yorku.ca

Abstract

While Large Language Models (LLMs) excel at text-to-SQL, their outputs lack reliability guarantees, leading to costly errors from incorrect queries. Existing validation and ranking methods fail to provide these formal guarantees. We propose RTS++ , a novel uncertainty-aware pipeline that ensures query reliability. RTS++ introduces execution entropy, a new measure that quantifies uncertainty by clustering candidates based on their execution results, capturing semantic ambiguity that token-level metrics miss. We integrate this measure into a conformal prediction (CP) framework to provide formal statistical correctness guarantees. This allows our system to reliably abstain from low-confidence queries, flagging them for human review. We evaluate RTS++ on the Spider and BIRD benchmarks, demonstrating that our human-in-the-loop approach achieves state-of-the-art execution accuracy of 92.3% and 92.5%, respectively. RTS++ offers a principled framework for building trustworthy natural language database interfaces.

1 Introduction

Text-to-SQL systems have emerged as a transformative technology, enabling users to query structured databases using natural language, eliminating the need for SQL expertise. This capability is particularly valuable in domains such as business intelligence, healthcare, and scientific research, where analysts and domain experts often lack formal training. Recent advancements in large language models (LLMs) have significantly improved the performance of text-to-SQL systems [6, 14], achieving state-of-the-art selective results on benchmarks such as Spider [23] and BIRD [14].

Despite these improvements, ensuring the reliability of generated SQL queries remains a major challenge. In real-world applications, incorrect SQL generation can lead to misleading business insights, flawed scientific conclusions, or operational failures. Unlike traditional database systems, text-to-SQL models lack explicit guarantees of query accuracy. Errors such as incorrect table joins, faulty aggregations, and misaligned schema references remain prevalent. These issues are further exacerbated by schema heterogeneity, where variations in database structures make generalization across different schemas difficult [11].

Existing methods for improving query accuracy primarily rely on post-generation validation and ranking techniques [7, 17, 20, 27]. Validation-based approaches detect syntactic anomalies but are limited in ensuring semantic correctness, as queries may still run without error but return plausible yet incorrect results.

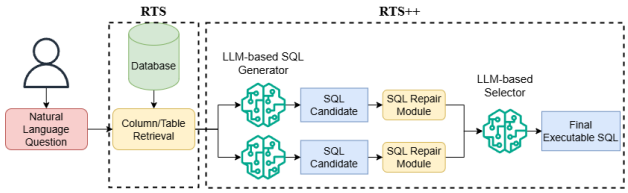

Figure 1: A high-level overview of an LLM-enhanced text-to-SQL pipeline with schema linking, candidate generation and repair, and final query selection.

Ranking-based approaches attempt to select the most likely correct query but are inherently vulnerable to model biases and the difficulty of defining a universal ranking function for diverse database schemas [16, 24].

These limitations highlight a critical gap: a principled approach to quantifying and ensuring query reliability. Recent work, Reliable Text-to-SQL (RTS) [2], made significant progress by focusing on the schema linking bottleneck. RTS ensures natural language references are correctly mapped to database elements and uses a human-in-the-loop approach to abstain from low-confidence links, prompting user review to resolve ambiguities.

While RTS enhances reliability at the schema-linking stage, it does not provide a formal statistical framework for quantifying uncertainty in the final output, namely the generated SQL query itself. A query can be generated from perfectly linked schema and still be semantically incorrect. To address this, we propose RTS++ , a novel uncertainty-aware text-to-SQL pipeline that integrates conformal prediction (CP), a statistical framework that quantifies uncertainty in the final generated queries. A core component of our framework is execution entropy, a new uncertainty measure that clusters query candidates by their execution results, capturing semantic ambiguity that token-level metrics miss.

Our CP-based approach provides formal guarantees: queries that meet a predefined confidence threshold are executed directly, while low-confidence queries are flagged for refinement or expert review. By incorporating this statistical measure, our approach ensures that only high-confidence SQL queries are used in decision-making.

Our contributions are summarized as follows: (1) We introduce an execution-aware uncertainty measure, **execution entropy**, that incorporates feedback from query execution; (2) We propose a full uncertainty-aware framework, **RTS++** , that integrates conformal prediction to provide statistical correctness guarantees; and (3) We evaluate RTS++ on Spider and BIRD, demonstrating that our human-in-the-loop approach achieves state-of-the-art reliability, with a **selective execution accuracy** (on the answered subset) of **92.3%** and **92.5%** respectively.

EDBT '26, Tampere (Finland)

10.48786/edbt.2026.44

2 Preliminaries

To set the context for our method, we first briefly review the standard text-to-SQL pipeline and then define the critical challenges of applying uncertainty quantification to SQL.

2.1 The Standard Text-to-SQL Pipeline

Text-to-SQL refers to the task of transforming a natural language question Q and a relational database schema S into an executable SQL query. Formally, the task aims to generate a query $f_\theta(Q, S)$ such that when executed on a database instance D conforming to S, it retrieves the user's intended answer.

Large language models (LLMs) have emerged as the dominant approach, leveraging their pre-trained knowledge to bridge natural language and structured schemas [13, 14, 14, 23]. As noted in prior work [7, 15, 17], the most effective LLM-based systems adopt a structured, three-stage pipeline to solve this task:

(1) **Schema Linking:** The model first analyzes the query Q and schema S to map natural language references (e.g., "highest population") to the corresponding database components (e.g., the `Population` column in the `Cities` table). Accurate linking is a critical prerequisite for correct query generation.
(2) **Candidate Generation:** Using the linked schema, an LLM generates one or more SQL query candidates. This stage often involves specialized prompt engineering or fine-tuning to produce syntactically valid queries that conform to the schema's constraints.
(3) **Candidate Selection:** Finally, if multiple candidates are generated, a selection or re-ranking model chooses the single most promising query. This is often another LLM call that evaluates the candidates based on likelihood or other heuristics [15].

While this pipeline has proven effective, the final output typically lacks a formal reliability guarantee. This motivates the need for a principled uncertainty quantification framework, which we discuss next.

2.2 Uncertainty Quantification

Uncertainty quantification (UQ) is critical for building reliable AI systems. In text generation, uncertainty is often measured using model internals. A common method is token-level entropy [5, 22], which quantifies the model's confidence in its probability distribution over the next token. While effective for capturing local uncertainty, this metric struggles with semantic variation (i.e., different but valid tokens can express the same meaning). **Semantic entropy** [12] was proposed to address this by measuring variation in meaning rather than just syntactic differences.

This distinction is critical. Consider the following queries generated for the question: *"List the unique names of employees who work in the sales department"*:

Golden Query

```
SELECT DISTINCT(name) FROM employees WHERE
department = 'sales';
```

First Generated Query (Correct)

```
SELECT DISTINCT(name) FROM employees WHERE
id IN ( SELECT id FROM employees WHERE
department = 'Sales' );
```

Second Generated Query (Incorrect)

```
SELECT DISTINCT(name) FROM employees WHERE
department != 'sales';
```

A UQ method based on textual similarity (like edit distance or token overlap) would heavily penalize the **First Generated Query**. It is syntactically complex and looks very different from the Golden Query. However, it is functionally identical and semantically correct. This is an example of syntactic variability: multiple, different-looking SQL queries can express the same intent and produce the same correct result. A reliable UQ method must be robust to this.

Conversely, the **Second Generated Query** is syntactically very similar to the Golden Query, differing by only one operator (`!=` vs. `=`). A text-based UQ method would likely assign it a high confidence score. However, this query is semantically disastrous and returns the exact opposite of the intended result. This highlights the primary challenge of execution correctness: syntactic similarity is a poor and unreliable proxy for semantic correctness.

This example demonstrates that uncertainty in text-to-SQL cannot be measured at the surface level. Any reliable framework must go beyond textual analysis and incorporate **execution behavior**. This motivates our proposal for *execution entropy*, a novel uncertainty measure tailored for database querying, which we introduce in Section 3.3.

2.3 Conformal Prediction for Text-to-SQL

Conformal Prediction (CP) is a statistical framework that provides formal, model-agnostic reliability guarantees [1]. For a user-defined confidence level α, CP constructs a prediction set $C(x)$ that is guaranteed to contain the true correct prediction y^* with a probability of at least $1 - \alpha$, such that:

$$P(y^* \in C(x)) \geq 1 - \alpha$$

This guarantee is achieved by using a small, held-out calibration set of examples with known correct labels. This set is used to learn a confidence threshold τ_α for a chosen nonconformity measure $c(y, x)$. This measure scores how "unusual" or "non-conforming" a given prediction y is, relative to the model's behavior on the calibration data. The final prediction set for a new input x is then defined as all predictions whose nonconformity score is below this threshold:

$$C(x) = \{y \mid c(y, x) \leq \tau_\alpha\}$$

The entire validity of the CP framework hinges on the choice of this nonconformity measure. This presents the critical gap, as Section 2.2 demonstrated that standard, text-based metrics are unreliable for SQL. If the nonconformity measure is based on syntactic similarity (like edit distance), the framework will fail. It would incorrectly penalize the functionally identical First Generated Query (high nonconformity) and incorrectly accept the semantically disastrous Second Generated Query (low nonconformity).

Therefore, a reliable CP framework for SQL is not possible with standard, text-based nonconformity scores. It requires a new measure that is aware of execution correctness. This motivates our core contribution: an execution-aware nonconformity score built on our novel *execution entropy* measure.

3 The RTS++ Framework

To address the challenges of uncertainty and reliability for text-to-SQL generation, we introduce RTS++, our framework for reliable end-to-end text-to-SQL generation. RTS++ enhances the standard 3-stage pipeline by integrating statistical guarantees at each critical step. As depicted in Figure 1 , the framework first leverages the existing RTS [2] system for reliable schema linking, then

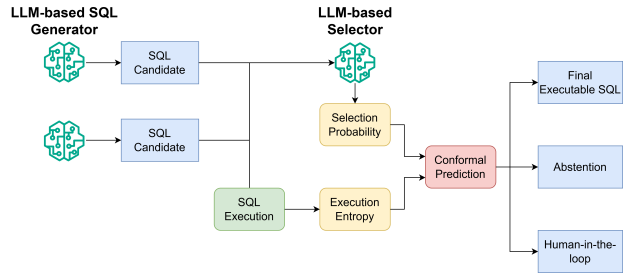

Figure 2: Overview of the RTS++ SQL Generator. This component enhances generation and selection with reliable signals, including execution entropy and conformal prediction.

passes the trusted schema to our novel Reliable SQL Generator to produce a final, high-confidence query.

3.1 Reliable Schema Linking

The first stage of our pipeline, schema linking, is handled by the Reliable Text-to-SQL (RTS) framework [2]. RTS integrates Branching Point Prediction (BPP), a form of conformal prediction, to estimate the confidence of each individual schema link (e.g., mapping a natural language phrase to a specific column). When the model identifies a high-uncertainty link, it does not silently proceed. Instead, it either abstains or triggers a human-in-the-loop interaction, proactively asking the user to resolve the ambiguity. As shown in [2], this strategy is highly effective: it achieves near-perfect linking accuracy across diverse benchmarks while keeping the number of user interventions low. This strikes an effective balance between autonomy and correctness, providing a high-confidence, trusted set of linked schema elements for the downstream components.

3.2 Reliable SQL Generator

After the high-confidence schema links are acquired, they are passed to our novel Reliable SQL Generator, as illustrated in Figure 2. This component is responsible for the subsequent generation and selection stages and contains our core contributions.

The process begins with the Candidate Generation module, which produces multiple diverse SQL queries. These candidates are then evaluated to generate two key reliability signals. First, our novel Execution Entropy measure quantifies uncertainty based on execution results. Second, a Candidate Selection model assigns a Selection Probability score. These two reliability signals—probability and entropy—are fed into the final Conformal Prediction module, which decides if a query is reliable enough to execute or if the system must abstain and defer to a human.

3.2.1 Candidate Generation The candidate generation stage produces multiple SQL queries that could potentially answer the user's question. To do this, we prompt an LLM with the natural language query and a serialized representation of the trusted schema. Following prior work [17], we use multiple prompt templates to encourage the generation of m diverse SQL candidates for the question Q, $S = \{s_1, ..., s_m\}$. Generating a diverse set increases the likelihood that at least one correct query is present.

The only requirement for this stage is that we can access the model's decoding logits to derive the generated sequence likelihood $p(s_i|Q)$. This probability information is essential for computing the selection probability and our execution entropy measure in the following stages.

3.3 Execution Entropy

As discussed in Section 2.2, standard text-based uncertainty metrics fail to capture semantic stability. To address this, we introduce **Execution Entropy**, a measure rooted in self-consistency that quantifies uncertainty over the space of execution results rather than surface-form tokens.

Given m generated candidates $S = \{s_1, \ldots, s_m\}$ with sequence probabilities $p(s_i \mid Q)$, we first execute all candidates against the target database. To group these executions meaningfully, we map each candidate's output to a canonical form $\Phi(\cdot)$. This is necessary because raw SQL results often differ in non-semantic ways. Our canonicalization process applies three transformations:

- **Set Normalization:** This transformation handles unordered results by sorting the rows and columns of the result set. For example, $\{A, B\} \equiv \{B, A\}$.
- **Type Standardization:** We unify numerical precision to prevent insignificant formatting differences from creating distinct clusters. This ensures values such as `42` and `42.0` are treated as identical.
- **Null Handling:** We treat `NULL` values as distinct, explicit tokens to ensure they are not conflated with empty strings or zeros.

We define two candidates s_i and s_j as belonging to the same cluster r if and only if their canonicalized results match:

$$\Phi(\text{Exec}(s_i)) = \Phi(\text{Exec}(s_j))$$

For each identified cluster r, let S_r be the subset of candidates belonging to that cluster. We calculate the cluster probability $P(r \mid Q)$ by marginalizing over S_r:

$$P(r \mid Q) = \frac{1}{\mathcal{Z}} \sum_{s_j \in S_r} p(s_j \mid Q) \tag{1}$$

where $\mathcal{Z} = \sum_{s_k \in S} p(s_k \mid Q)$ is the normalization factor. We then assess the overall semantic confusion using cluster entropy $H(Q)$:

$$H(Q) = -\sum_{r \in \mathcal{R}} P(r \mid Q) \log P(r \mid Q) \tag{2}$$

Finally, to score an individual candidate s_i belonging to result cluster r, we calculate the Execution Entropy Score H_{exec} by combining cluster entropy with the negative log-probability of its cluster:

$$H_{\text{exec}}(s_i) = \underbrace{H(Q)}_{\text{Global Confusion}} \quad \underbrace{-\log P(r \mid Q)}_{\text{Local Outlier}} \tag{3}$$

To understand the utility of this formulation, consider the distinction between *syntactic* and *semantic* variance. If a model generates ten distinct SQL queries that all produce the same result, standard token-based entropy would register high uncertainty. In contrast, our metric registers zero uncertainty. Conceptually, $H_{\text{exec}}(s_i)$ identifies two specific failure modes:

- **Global Confusion** (High $H(Q)$): The model spreads probability mass across many conflicting answers (e.g., results A, B, C, D all have ≈25% probability), suggesting it does not know the answer.
- **Local Outliers** (Low $\log P(r \mid Q)$): The model is globally confident in a dominant result, but the specific candidate s_i produces a rare execution result (an outlier). Because s_i

links to this unlikely outcome rather than the consensus, it is penalized as untrustworthy.

We employ $H_{\text{exec}}(s_i)$ as the nonconformity score in our conformal prediction framework, ensuring that the constructed credible set prioritizes candidates that belong to the dominant consensus within a stable distribution.

3.3.1 Execution-Guided Conformal Prediction The final stage of our framework, candidate selection, uses the reliability signals from the previous steps to build the formal Conformal Prediction (CP) framework. As motivated in Section 2.3, the key to a reliable CP framework is an execution-aware nonconformity score. We now show how we construct this score by combining our *execution entropy* with a selection probability.

Our nonconformity score is built from two signals. The first is the **Selection Probability (P_{sel})**. For each candidate s_i in our set S, we use a reranker model M to assign a score $P_{\text{sel}}(s_i \mid Q)$, representing the model's likelihood of s_i being the correct query. The second signal is our **Execution Entropy (H_{exec})**, $H_{\text{exec}}(s_i)$, which we defined in Section 3.3. This score represents the uncertainty of the *execution result* that s_i produces.

We combine these two signals into a single **Execution-Aware Score**, $F(s_i)$, for each candidate:

$$F(s_i) = P_{\text{sel}}(s_i \mid Q) \cdot \exp(-\lambda H_{\text{exec}}(s_i))$$

Here, λ is a hyperparameter that controls the trade-off. This score is intuitively powerful: it is high only for candidates that both have a high individual probability (P_{sel}) and belong to a low-entropy, high-confidence result cluster.

To integrate this into the CP framework, we define our nonconformity score $A(s_i)$ as the negative of our score (since high scores are good, they should represent low nonconformity):

$$A(s_i) = -F(s_i)$$

Using a held-out calibration set, we compute this score for all calibrated examples and find the $(1-\alpha)$ quantile, which gives us our reliability threshold τ_α. The final credible set C_α is then all candidates whose nonconformity score is at or below this threshold:

$$C_\alpha = \{s_i \in S \mid A(s_i) \leq \tau_\alpha\}$$

This credible set C_α is guaranteed to contain the correct SQL query with a probability of at least $1-\alpha$. The RTS++ system then makes its final decision based on the contents of this set:

- **Abstention:** If $C_\alpha = \emptyset$ (the set is empty), it means no candidate met the reliability threshold. The system abstains and returns no query, preventing a potential error.
- **Single Candidate:** If $|C_\alpha| = 1$ (the set has exactly one member), that single query is selected as the final, trusted prediction.
- **Multiple Candidates:** If $|C_\alpha| > 1$, it means the system is confident in multiple candidates. The system abstains due to ambiguity or, in a human-in-the-loop setting, presents the credible set to the user for disambiguation.

4 Experimental Evaluation

We now present our experimental evaluation, designed to assess the effectiveness and reliability of the RTS++ framework. We structure this section as follows: First, we detail the **Experimental Setup**, including our datasets, baselines, and metrics. We then present our **Main Results** on the Spider and BIRD benchmarks, focusing on the trade-off between execution accuracy and coverage. Following this, we provide in-depth **Ablation Studies** to validate the contribution of key components like execution entropy. We conclude this section by analyzing the **Computational Overhead** and summarizing broader insights.

4.1 Experimental Setup

We conduct experiments to evaluate the effectiveness and reliability of RTS++ in the text-to-SQL task. Our goal is twofold: (1) to assess whether RTS++ can maintain high execution accuracy while offering probabilistic correctness guarantees, and (2) to examine the benefit of abstention and human-in-the-loop mechanisms in handling ambiguous queries.

Datasets. [1] We report results on **Spider** [23] for cross-domain generalization and **BIRD** [14], which reflects complex, real-world database scenarios. For both datasets, we use the respective training sets to construct calibration datasets. For the BIRD benchmark, we report evaluation results on the development set. For the Spider benchmark, we report results on the official test set.

Implementation Details. We implement our system using two different large language models: GPT-4.1 and DeepSeek. For schema linking, we adopt the RTS framework. In all experiments, we assume users provide accurate feedback during the interaction loop. We fix the user-defined error tolerance parameter α to 0.1 (implying a target reliability of 90%). Regarding the weighting coefficient λ, we performed a grid search on the validation set and empirically set $\lambda = 1$, as equal weighting between selection probability and entropy yielded the most stable calibration

4.2 Models and Baselines

We compare our approach against the following models:

- **Reranking Pipeline**: A standard generation pipeline that produces multiple SQL candidates and selects the top-1 based on likelihood, without any abstention mechanism.
- **RTS++ & Abstention**: Our proposed method using Conformal Prediction to abstain when the confidence set is empty or contains multiple distinct result clusters.
- **RTS++ & Human Feedback**: Our method extended with a human-in-the-loop component, where the credible set is presented to a user for disambiguation instead of abstaining.
- **Best Reported Method**: The strongest-performing system previously reported for each benchmark. For Spider, we report **MiniSeek** [23]; for BIRD, we report **AskData + GPT-4o** [14]. Note that these baselines operate at 100% coverage (no abstention).

Evaluation Metrics. To ensure a fair comparison regarding the reliability-coverage trade-off, we report:

(1) **Selective Execution Accuracy (Reliability):** The percentage of correct queries among the *non-abstained* subset. This measures the trustworthiness of the system's answers.
(2) **Abstention Rate:** The percentage of queries where the system refuses to answer.
(3) **Effective Error Rate:** The frequency of incorrect answers explicitly delivered to the user. Unlike standard accuracy

[1] Note that while other text-to-SQL datasets (e.g., WikiSQL[26], Spider2[13], BEAVER[3]) are available, they either (1) lack the complex schema and query variety needed to evaluate probabilistic guarantees, or (2) do not include a development set suitable for calibrating reliability-aware systems. Thus, we focus on Spider and BIRD, which offer the most rigorous and representative settings for our evaluation.

(which penalizes abstention as error), this metric penalizes *silent failures*—incorrect answers that the user might blindly trust.

4.3 Results

Model	LLM	Selective Acc.	Abst. Rate
Best Reported Method	–	91.2%	0.0%
Reranking Pipeline	GPT-4.1	82.1%	0.0%
RTS++ & Abstention	GPT-4.1	89.8%	18.4%
RTS++ & Human Feedback	GPT-4.1	91.4%	18.4%
Reranking Pipeline	DeepSeek	84.2%	0.0%
RTS++ & Abstention	DeepSeek	90.7%	16.1%
RTS++ & Human Feedback	DeepSeek	92.3%	16.1%

Table 1: Performance on Spider. "Selective Acc." denotes accuracy on the non-abstained subset.

Model	LLM	Selective Acc.	Abst. Rate
Best Reported Method	GPT-4-o	77.6%	0.0%
Reranking Pipeline	GPT-4.1	62.6%	0.0%
RTS++ & Abstention	GPT-4.1	78.4%	31.2%
RTS++ & Human Feedback	GPT-4.1	85.2%	31.2%
Reranking Pipeline	DeepSeek	70.2%	0.0%
RTS++ & Abstention	DeepSeek	88.5%	30.6%
RTS++ & Human Feedback	DeepSeek	92.5%	30.6%

Table 2: Performance on BIRD. The high abstention rate (≈30%) reflects the higher ambiguity of BIRD queries.

4.3.1 Spider Benchmark Results Table 1 presents results on the Spider benchmark. To enable a fair comparison, we report **Selective Execution Accuracy**, which is computed only over non-abstained predictions.

On GPT-4.1, the reranking pipeline baseline achieves 82.1% accuracy. When equipped with our abstention mechanism, the selective accuracy rises to 89.8%, while abstaining on 18.4% of inputs. Crucially, this significantly lowers the **Effective Error Rate** (from 17.9% to ≈10%), meaning fewer incorrect queries are shown to the user. With human feedback incorporated, performance improves to 91.4%, surpassing the best reported result from MiniSeek (91.2

Using the stronger DeepSeek model, the reranking pipeline reaches 84.2% accuracy. Our method with abstention increases selective accuracy to 90.7%, and human feedback pushes it to 92.3%. While the baseline MiniSeek achieves 91.2% with 100% coverage, RTS++ achieves comparable or superior accuracy on the trusted subset (84% coverage), effectively filtering out potential errors before they reach the user.

4.3.2 BIRD Benchmark Results Table 2 summarizes results on the BIRD benchmark, where the higher query ambiguity makes reliability even more critical.

On GPT-4.1, the reranking pipeline baseline achieves only 62.6% accuracy, meaning nearly 40% of generated queries are incorrect. With RTS++ , the selective accuracy improves substantially to 78.4% (+15.8 points), despite abstaining on 31.2% of examples. This high abstention rate reflects the difficulty of the dataset; however, it ensures that the answers provided are trustworthy. Incorporating human feedback leads to a further gain, reaching 85.2%.

Stronger results are observed with the DeepSeek backend. The reranking pipeline achieves 70.2% accuracy. Our method with abstention improves selective accuracy to 88.5%, and human feedback further boosts it to 92.5%. Critically, comparing RTS++ to the best reported method (AskData, 77.6%) requires considering the coverage trade-off. While AskData answers every query, it has an effective error rate of ≈22.4%. In contrast, RTS++ (with abstention) has a selective error rate of only 11.5%. This demonstrates that RTS++ is far safer for deployment in high-stakes environments where silent errors are costly.

4.4 Ablation Studies

We further investigate the contribution of individual components in our system through a series of ablation experiments. Unless otherwise stated, all results in this section are reported on the BIRD benchmark using the DeepSeek backend with human feedback.

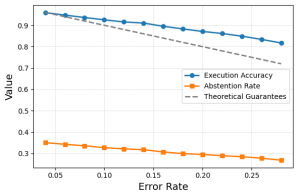

Figure 3: Effect of conformal threshold on accuracy and abstention.

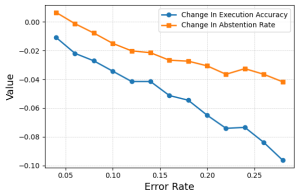

Figure 4: Effect of removing execution entropy from uncertainty estimation.

4.4.1 Conformal Prediction Error Tolerance Figure 3 presents how execution accuracy and abstention rate vary as we adjust the conformal prediction error tolerance α. As expected, lower values of α—corresponding to stricter reliability targets—lead to higher execution accuracy but increased abstention.

Interestingly, we determine that our method consistently maintains execution accuracy well above the theoretical lower bound guaranteed by conformal prediction (dashed line), indicating that our calibration procedure is effective in identifying genuinely uncertain cases without being overly cautious. The abstention rate declines from 35% to 27% as the error rate increases, allowing practitioners to tune the system for the desired balance between reliability and coverage.

4.4.2 Execution Entropy. We conduct an ablation to isolate the contribution of execution entropy. Specifically, we remove the entropy signal (setting $\lambda = 0$) during confidence calibration and analyze the downstream impact.

Figure 4 shows that removing execution entropy consistently leads to reduced execution accuracy and a simultaneous drop in abstention rate. Execution accuracy degrades by up to 9.5%, while abstention rate decreases by up to 4%. This indicates that without the entropy signal, the system becomes **overconfident**, failing to flag ambiguous queries that result in execution errors. This confirms that execution entropy is a critical signal for detecting semantic ambiguity that simple probability metrics miss.

4.4.3 Computational Overhead A key concern in execution-based approaches is latency. We analyzed the wall-clock time per query on the BIRD development set. The average total latency per query ranges from approximately **3.7s to 30s**. This variance is driven primarily by the LLM generation overhead and API queuing delays. Crucially, the computational cost introduced by RTS++ itself—executing m candidates and computing clusters—is marginal ($\approx$ **0.1s** per query) for standard OLTP workloads. Thus, the reliability mechanism adds negligible overhead to the standard generation pipeline.

4.5 Key Insights and Limitations

Our experiments demonstrate that RTS++ transforms text-to-SQL from a stochastic "best-effort" task into a reliable service. Current SOTA methods on BIRD plateau at approximately 78-82% accuracy[14], implying that nearly one in five queries results in a silent error. This high failure rate highlights the inherent ambiguity in real-world database inquiries; whereas existing models effectively "guess" on these uncertain inputs, RTS++ utilizes execution entropy to detect the ambiguity and makes the safe choice to abstain. This effectively converts dangerous silent errors into safe refusals, bridging the critical reliability gap required for deployment.

Limitations. The primary limitation is the coverage-precision trade-off; to gain safety, users must tolerate abstention. Additionally, our reliance on execution signals introduces deployment constraints:

- **Scalability:** While overhead is negligible ($\approx 0.1s$) for standard benchmarks, executing multiple candidates on large-scale data warehouses may require sampling or Approximate Query Processing to maintain low latency.
- **Safety:** Execution-based validation necessitates a read-only or sandboxed environment to prevent unintended side effects during the candidate generation phase.

5 Related Work

Text-to-SQL Generation and Benchmarks. The evolution of text-to-SQL has been driven by a succession of increasingly challenging benchmarks. Early approaches employed sketch-based decoders on simpler datasets like WikiSQL [26], while subsequent architectures such as IRNet [9] and RAT-SQL [20] introduced intermediate representations to tackle the complex schema linking found in Spider [23]. The recent shift toward Large Language Models (LLMs) has unlocked remarkable generalization capabilities through in-context learning [11]. However, this paradigm shift has also exposed a critical reliability gap: on the rigorous BIRD benchmark [14]—which reflects real-world semantic ambiguity—even state-of-the-art methods have plateaued at approximately 77–82% execution accuracy. This performance ceiling highlights that accuracy maximization alone is insufficient for production deployment, necessitating the robust uncertainty estimation and abstention mechanisms we propose. While newer benchmarks like Spider 2.0 [25] and BEAVER [3] have emerged, we focus our evaluation on Spider and BIRD to strictly assess high-confidence, single-turn generation capabilities.

Uncertainty Estimation in Structured Prediction. Estimating uncertainty is critical for deploying structured prediction models in high-stakes environments. Traditional entropy-based measures, while effective for classification, are limited in text-to-SQL by the problem of *semantic variability*: different SQL queries can be syntactically distinct yet functionally equivalent. Recent advancements like Semantic Entropy [12] address this by clustering outputs based on meaning, while Conformal Prediction [1] provides a rigorous statistical framework for constructing valid confidence sets. Our work builds upon these foundations but diverges by introducing *execution-based* uncertainty measures. Furthermore, recent work has extended conformal guarantees to non-exchangeable text generation settings [19], ensuring applicability even when distribution shifts occur between training and deployment.

Selective Prediction. Selective prediction and abstention have been explored in classification [8] and more recently in NLP [21]. In the text-to-SQL domain, RTS [2] introduces abstention specifically for schema linking. Our work generalizes this mechanism across the full text-to-SQL pipeline, integrating execution entropy and conformal confidence to detect failures in the generation logic itself, not just the linking phase.

Human-in-the-Loop Feedback. Incorporating user feedback improves parser reliability. Prior work includes clarification-based systems [10], interactive editing [18], and feedback-aware training [4]. We adopt a similar approach, triggering lightweight user input only for abstained queries.

6 Conclusion

We present a principled and practical framework for reliable text-to-SQL generation that integrates uncertainty estimation, conformal prediction, and selective abstention. By quantifying the model's confidence through execution entropy and conformal calibration, our method enables the system to abstain from uncertain predictions and engage human feedback only when necessary. This approach bridges the gap between model autonomy and reliability, addressing critical limitations in existing reranking and validation-based strategies.

Through extensive experiments on Spider and BIRD, we demonstrate that our method significantly improves execution accuracy and robustness across both standard and large-scale settings. Moreover, the framework allows practitioners to adjust the system's error tolerance to control when the model should abstain, providing flexibility for deployment in risk-sensitive applications. While our work focuses on single-turn text-to-SQL, the core ideas of uncertainty-aware abstention and human-in-the-loop feedback are broadly applicable to other structured prediction tasks.

7 Artifacts

The artifacts for this paper are available in our GitHub repository: https://github.com/kckevinchen/RTS-SQL.

References

[1] Anastasios N Angelopoulos and Stephen Bates. 2021. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511* (2021).

[2] Kaiwen Chen, Yueting Chen, Nick Koudas, and Xiaohui Yu. 2025. Reliable Text-to-SQL with Adaptive Abstention. *Proc. ACM Manag. Data* 3, 1, Article 69 (Feb. 2025), 30 pages. doi:10.1145/3709719

[3] Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2024. BEAVER: an enterprise benchmark for text-to-sql. *arXiv preprint arXiv:2409.02038* (2024).

[4] Ahmed Elgohary, Xinyi Lin, and Mohit Iyyer. 2020. Speak to your parser: Interactive text-to-SQL with natural language feedback. In *Proceedings of ACL*.

[5] Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, and Maxim Panov. 2024. Fact-Checking the Output of Large Language Models via Token-Level Uncertainty Quantification. *arXiv preprint arXiv:2403.04696* (2024). https://arxiv.org/abs/2403.04696

[6] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *CoRR* abs/2308.15363 (2023). https://arxiv.org/abs/2308.15363

[7] Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, Jinyang Gao, Liyu Mou, and Yu Li. 2024. A Preview of XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL. *arXiv preprint arXiv:2411.08599* (2024). https://arxiv.org/abs/2411.08599

[8] Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[9] Jiaqi Guo, Zecheng Sun, Lanjun Wang, Yujia Fan, Xiaoyan Zhang, Jiannan Tang, Daxin Zhao, and Xipeng Qiu. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of ACL*.

[10] He He, Percy Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop Parsing: Syntactic Parsing with Partial Feedback. In *Proceedings of ACL*.

[11] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2025. Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL. *IEEE Transactions on Knowledge and Data Engineering* 37, 12 (2025), 7328–7345. doi:10.1109/TKDE.2025.3609486

[12] Jonas Kuhn, Colin Raffel, Jason Phang, Angelina McMillan-Major, and Samuel R Bowman. 2023. Semantic entropy: A measure of uncertainty for language models. *arXiv preprint arXiv:2310.02237* (2023).

[13] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2024. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. arXiv:2411.07763 [cs.CL] https://arxiv.org/abs/2411.07763

[14] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. 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 (2024). https://bird-bench.github.io/

[15] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O Arik. 2025. CHASE-SQL: Multi-Path Reasoning and Preference Optimized Candidate Selection in Text-to-SQL. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=CvGqMD5OtX

[16] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *arXiv preprint arXiv:2304.11015* (2023).

[17] Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. CHESS: Contextual Harnessing for Efficient SQL Synthesis. *arXiv preprint arXiv:2405.16755* (2024).

[18] Yuanhe Tian, Qian Liu, Tao Yu, and Dragomir Radev. 2023. Explanation-Based Editing for Text-to-SQL Semantic Parsing. In *Proceedings of ACL*.

[19] Dennis Ulmer, Chrysoula Zerva, and Andre Martins. 2024. Non-Exchangeable Conformal Language Generation with Nearest Neighbors. In *Findings of the Association for Computational Linguistics: EACL 2024*, Yvette Graham and Matthew Purver (Eds.). Association for Computational Linguistics, St. Julian's, Malta, 1909–1929. https://aclanthology.org/2024.findings-eacl.129/

[20] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Yu Su. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of ACL*.

[21] Jia Xin, Mohit Iyyer Yu, et al. 2021. The art of abstention: Selective prediction and error regularization for NLP. In *Proceedings of ACL*.

[22] Jiacheng Xu, Shrey Desai, and Greg Durrett. 2020. Understanding Neural Abstractive Summarization Models via Uncertainty. *arXiv preprint arXiv:2010.07882* (2020). https://arxiv.org/abs/2010.07882

[23] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, Xi Victoria Ma, James Li, William Yao, Shanelle Zhang, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of EMNLP*.

[24] Jichuan Zeng, Xi Victoria Lin, Steven C.H. Hoi, Richard Socher, Caiming Xiong, Michael Lyu, and Irwin King. 2020. Photon: A Robust Cross-Domain Text-to-SQL System. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Asli Celikyilmaz and Tsung-Hsien Wen (Eds.). Association for Computational Linguistics, Online, 204–214. doi:10.18653/v1/2020.acl-demos.24

[25] Victor Zhong et al. 2023. Spider 2.0: Towards Open-domain Text-to-SQL Semantic Parsing. *arXiv preprint arXiv:2306.06063* (2023).

[26] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.00103 (2017).

[27] Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Seq2SQL: Generating Structured Queries From Natural Language Using Reinforcement Learning. https://openreview.net/forum?id=Syx6bz-Ab