LatentRefusal: Latent-Signal Refusal for Unanswerable Text-to-SQL Queries

Xuancheng Ren **Shijing Hu** **Zhihui Lu*** **Jiangqi Huang**
Fudan University
{xcren25, sjhu24, 25213050189}@m.fudan.edu.cn, lzh@fudan.edu.cn

Qiang Duan
Pennsylvania State University Abington College
qduan@psu.edu

Abstract

In LLM-based Text-to-SQL systems, unanswerable and underspecified user queries may generate not only incorrect text but also executable programs that yield misleading results or violate safety constraints, thus posing a major barrier to safe deployment. Existing refusal strategies for such queries either rely on output-level instruction following, which is brittle due to model hallucinations, or on estimating output uncertainty, which adds complexity and overhead. To address this challenge, we first formalize safe refusal in Text-to-SQL systems as an answerability-gating problem, and then propose LATENTREFUSAL, a latent-signal refusal mechanism that predicts query answerability from intermediate hidden activations of an LLM. We introduce the Tri-Residual Gated Encoder (TRGE), a lightweight probing architecture, to suppress schema noise and amplify sparse, localized question–schema mismatch cues that indicate unanswerability. Extensive empirical evaluations across diverse ambiguous and unanswerable settings, together with ablations and interpretability analyses, demonstrate the effectiveness of the proposed scheme and show that LATENTREFUSAL provides an attachable, efficient safety layer for Text-to-SQL systems. Across four benchmarks, LATENTREFUSAL achieves an average F1 of 88.5% and 88.8% on Llama-3.1-8B and Qwen-3-8B respectively, while adding $\sim$2ms probe overhead.

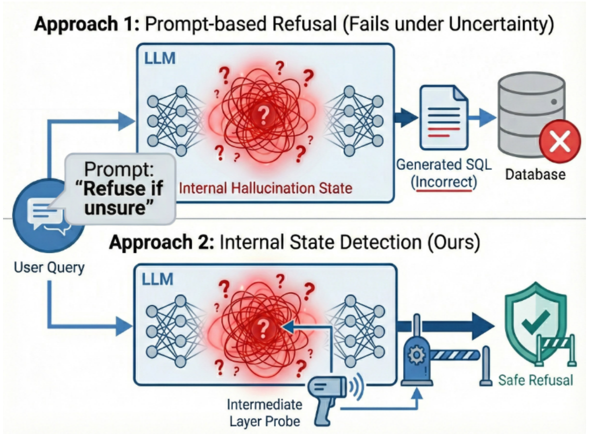

Figure 1: Comparison of refusal paradigms in Text-to-SQL systems. **Top:** Traditional prompt-based methods rely on the LLM's decoded output to decide refusal, which is brittle under uncertainty and can fail when the model hallucinates plausible but incorrect SQL. **Bottom:** LATENTREFUSAL detects refusal signals directly from the frozen LLM's intermediate hidden states *before* any SQL generation, enabling a single-pass, deterministic, and low-latency safety gate without generating or executing potentially harmful queries.

1 Introduction

Large language models (LLMs) have broadened access to data analytics by translating natural language questions into executable SQL (Text-to-SQL) (Sun et al., 2024; Gao et al., 2024; Liu et al., 2024). However, in real deployments (e.g., finance, healthcare, security), practical adoption is constrained by a safety-critical failure mode: unreliable behavior under *unanswerable* or *underspecified* queries. Such queries may require nonexistent schema elements, admit multiple plausible interpretations, fall outside the database scope, or depend on subjective criteria. When optimized for helpfulness, LLMs can still produce plausible-looking SQL that is semantically incorrect. Unlike open-ended dialogue, where hallucinations mainly yield incorrect text, Text-to-SQL hallucinations yield *executable programs*, which can silently corrupt reports, trigger privacy violations, or cause costly operational incidents (Huang et al., 2023; Ji et al., 2023).

Throughout this paper, we define *answerable* queries as those resolvable to a unique, valid SQL statement given the schema. Conversely, we use *unanswerable* to broadly encompass queries suffering from missing schema elements, out-of-scope

* Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 20162–20179
July 2-7, 2026 ©2026 Association for Computational Linguistics

requests, subjectivity, or linguistic ambiguity. In these contexts, *Text-to-SQL hallucination* refers to the generation of executable but semantically unfaithful SQL, a frequent risk when models are forced to answer unanswerable inputs.

Accordingly, a production Text-to-SQL system must be able to *refuse* when it cannot answer safely and faithfully. Existing refusal strategies, however, face a persistent trade-off between reliability and efficiency. *Prompt-based* methods instruct the model to abstain (e.g., "refuse if unsure"), but refusal behavior is brittle under uncertainty and can fail precisely when the model hallucinates. In contrast, *uncertainty-based* methods (e.g., self-consistency or semantic-entropy style scoring) can be more robust, yet often require multiple samples and substantial inference overhead. Worse, in Text-to-SQL, assessing agreement among candidate programs frequently depends on execution to resolve semantic equivalence—but executing questionable SQL to *estimate* uncertainty undermines the goal of safety gating.

We observe that semantic information is implicit in the intermediate layers, which can be leveraged for refusal judgment. Meanwhile, experiments show that the middle layers contain the most accurate refusal direction information (Skean et al., 2025). Accordingly, we propose LatentRefusal, a refusal mechanism that makes the safety decision *before generation* by detecting refusal signals directly from a frozen LLM's internal hidden states. As illustrated in Figure 1, rather than relying on the model's final output (top), LatentRefusal attaches a lightweight intermediate-layer probe to predict answerability in a single forward pass (bottom). This design yields a deterministic, low-latency refusal gate that avoids sampling, avoids generating potentially harmful SQL, and avoids executing any SQL during the decision process.

A key challenge is that naive probing over pooled hidden states is often insufficient for schema-conditioned Text-to-SQL: inputs can be dominated by lengthy schema descriptions, while refusal cues (e.g., a missing column, absent join path, or underspecified constraint) are subtle and localized. To address this, we introduce the Tri-Residual Gated Encoder (TRGE), a probe architecture designed to suppress schema noise and amplify question–schema mismatch signals indicative of unanswerability.

Our contributions are as follows:

- **Problem and constraint formulation for safe refusal in Text-to-SQL.** We formalize refusal as an *answerability gating* problem under a strict safety constraint: the system must decide whether to answer *before* generating or executing any SQL, avoiding execution-based uncertainty estimation.
- **LatentRefusal: Single-pass latent-signal refusal.** We propose a mechanism that predicts answerability directly from a frozen LLM's hidden activations. This enables deterministic, low-latency refusal decisions in a single forward pass, avoiding the overhead of sampling or unsafe query execution.
- **TRGE: A probe for sparse refusal cues in schema-heavy prompts.** We introduce the Tri-Residual Gated Encoder (TRGE), a lightweight SwiGLU-gated probing architecture designed to suppress schema noise and amplify localized question–schema mismatch signals that indicate unanswerability.
- **Empirical validation and analysis.** We evaluate LatentRefusal on diverse unanswerable and ambiguous Text-to-SQL settings, demonstrating improved refusal reliability at near-instruction-following cost, and provide ablations and interpretability analyses that isolate where refusal signals emerge in the latent space.

2 Related Work

Unanswerability and ambiguity in Text-to-SQL. Real-world Text-to-SQL must handle *ambiguous* questions (multiple valid interpretations) and *unanswerable* questions (cannot be grounded to the available schema/database). Prior work studies intention types and fine-grained unanswerability categories (Zhang et al., 2020; Wang et al., 2022), extends the setting to multi-turn conversations (Dong et al., 2024), and benchmarks linguistic ambiguity with paired ambiguous/clarified inputs (Saparina and Lapata, 2024). In this paper, we target a stricter system requirement: a **low-latency, pre-generation gate** that decides whether *any* SQL should be produced.

Prompt-based refusal and output-based uncertainty. Prompting an LLM to abstain is convenient and training-free, but refusal is prompt-sensitive and can fail exactly when the model hallucinates, after the system has already entered executable-code decoding. Output-based uncertainty uses sampling and disagreement signals (e.g., SelfCheckGPT and semantic entropy) (Manakul et al., 2023; Farquhar et al., 2024; Kuhn et al., 2023; Wang et al., 2023), while self-evaluation and selective generation use single-pass confidence, OOD-style scores, or verbalized uncertainty (Kadavath et al., 2022; Ren et al., 2023; Lin et al., 2022; Xiong et al., 2024). These methods are broadly applicable but often conflict with Text-to-SQL deployment constraints: multi-sampling is costly, and comparing SQL candidates can require execution or expensive reasoning, which is misaligned with **pre-execution** safety gating.

Latent/internal-state reliability signals. Recent work shows that reliability and truthfulness are reflected in latent space and internal activations, including latent truth directions (CCS) (Burns et al., 2023; Marks and Tegmark, 2024), generation-driven hallucination membership estimation (HaloScope) (Du et al., 2024), and internal-state predictors of hallucination (Azaria and Mitchell, 2023; Chen et al., 2024). These approaches motivate **single-pass, pre-generation** decisions when internal states are accessible. Our work follows this line but addresses a Text-to-SQL-specific challenge: schema-conditioned prompts are dominated by long schema tokens while refusal cues are sparse and localized, motivating a probe that suppresses schema noise and amplifies question-schema mismatch evidence (Kossen et al., 2024).

3 LatentRefusal

LatentRefusal is a *latent-signal* refusal mechanism for Text-to-SQL that decides whether to answer *prior to generation* by reading refusal cues from a frozen LLM's internal representations (Figure 2). The core novelty is to treat refusal as a *representation-level* decision problem rather than an *output-level* instruction-following behavior: instead of asking the LLM to say "I don't know" (which can fail under hallucination), we learn a compact detector on intermediate hidden states and insert it as a deterministic gate.

3.1 Refusal Gating Framework

Why a pre-generation gate? Text-to-SQL differs from open-ended generation in that the model output is executable code. When the query is unanswerable or underspecified (e.g., missing columns/tables, ambiguous constraints, non-existent entities, or out-of-scope requests), "helpful" decoding can still yield plausible SQL that executes successfully but answers the *wrong* question. Existing approaches face two limitations: (i) **Prompt-based refusal** relies on the model's decoded text to abstain, which is prompt-sensitive and may collapse precisely when the model enters a hallucination state. (ii) **Uncertainty-based refusal** often requires multi-sampling; in Text-to-SQL, comparing candidate programs can require execution or expensive equivalence checking, which conflicts with safety gating and increases latency.

Design goal. We aim for a refusal mechanism that is (i) **single-pass** (one forward pass through the base LLM), (ii) **pre-generation** (no SQL tokens are generated unless permitted), (iii) **execution-free** (no database interaction during gating), and (iv) **architecture-independent** (applicable to any frozen LLM with accessible intermediate activations). Concretely, "architecture-independent" means we do not update the base LLM parameters, while the probe is trained with lightweight supervision. The method requires white-box access to intermediate hidden states, which aligns with our target application (privacy-sensitive financial QA) where models are typically deployed locally rather than via closed APIs.

Gating rule. We implement a deterministic gate that intercepts the generation process based on the probe's output. Specifically, the system returns a refusal response if the predicted answerability probability $\hat{p} = g_\phi(\mathbf{H})$ falls below a calibrated threshold τ (i.e., $\hat{p} < \tau$), and proceeds to SQL generation otherwise. This mechanism decouples safety judgment from generation, ensuring unanswerable queries are blocked before any potentially harmful SQL is produced. This setup is consistent with the **reject option** and **selective classification** frameworks (Chow, 1970; El-Yaniv and Wiener, 2010; Geifman and El-Yaniv, 2017, 2019), where a model abstains from answering under uncertainty to achieve a better safety–utility trade-off.

What signal do we use? A frozen LLM encodes rich consistency information between question and schema during the forward pass, even when its final generation may hallucinate. LatentRefusal exploits this by operating directly on internal hidden states rather than on decoded tokens. Concretely, let $\mathcal{M}$ be the frozen base LLM and let $\mathbf{H}^{(l)} \in \mathbb{R}^{T\times d}$ denote the hidden states at layer l. We choose one layer index $l^\star$ (validated on

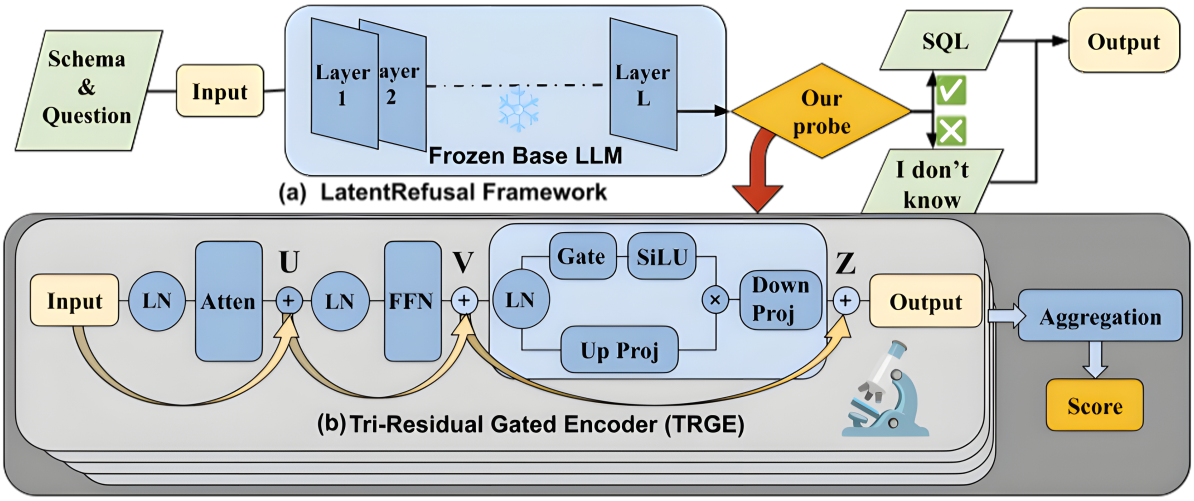

Figure 2: Overview of the LatentRefusal framework. **(a) Refusal gating pipeline:** given a natural-language question and a database schema, a frozen base LLM (e.g., Qwen-3-8B or Llama-3.1-8B) produces hidden states at a selected intermediate layer $l^\star$; a lightweight TRGE probe predicts answerability probability $\hat{p}$ *before* any SQL tokens are generated, and a binary gate (threshold τ) either triggers SQL generation or returns a safe refusal response. **(b) TRGE probe architecture:** each Tri-Residual Gated Encoder layer augments a standard Transformer block (self-attention + FFN) with a third SwiGLU-gated residual branch, designed to suppress schema noise and amplify sparse, localized question–schema mismatch cues indicative of unanswerability.

development data) and feed $\mathbf{H}^{(l^\star)}$ to the probe:

$$\hat{p} = g_\phi\left(\mathbf{H}^{(l^\star)}\right), \qquad \mathbf{H}^{(l^\star)} = \mathcal{M}^{(l^\star)}(x). \quad (1)$$

where ϕ represents the trainable parameters of the probe, and x is the input sequence. Using a single intermediate layer is intentionally minimal: it keeps the gate lightweight and avoids entanglement with the base model's decoder dynamics.

3.2 TRGE Probe

Challenge: schema-heavy prompts hide sparse refusal cues. In Text-to-SQL prompting, the schema tokens frequently dominate context length. However, answerability evidence is often localized: a single missing column mention, an unmatched entity, or an impossible join path. Simple pooling with a linear classifier can underfit because the relevant signal is sparse and can be overwhelmed by irrelevant schema content. A vanilla Transformer probe helps, but without an inductive bias to *suppress* schema noise it may still spend capacity modeling schema regularities rather than detecting mismatches.

Our probe design. We propose the **Tri-Residual Gated Encoder (TRGE)**, a small Transformer-style encoder that adds a *third* residual branch consisting of a SwiGLU gating module. The motivation is explicit: a content-aware gate can act as a soft feature selector that down-weights irrelevant schema patterns while amplifying mismatch features correlated with unanswerability.

Given an input representation $\mathbf{Z}^{(k-1)} \in \mathbb{R}^{T\times d}$ to TRGE layer k (where $d = 512$), we compute:

$$\begin{aligned} \mathbf{U} &= \mathbf{Z}^{(k-1)} + \mathrm{Attn}\left(\mathrm{LN}(\mathbf{Z}^{(k-1)})\right), \\ \mathbf{V} &= \mathbf{U} + \mathrm{MLP}(\mathrm{LN}(\mathbf{U})), \\ \mathbf{Z}^{(k)} &= \mathbf{V} + \mathrm{SwiGLU}(\mathrm{LN}(\mathbf{V})), \end{aligned} \quad (2)$$

where LN is layer normalization and Attn is multi-head self-attention with 8 heads. Here, $\mathbf{U}$ and $\mathbf{V}$ denote the intermediate residual states after the attention and MLP blocks, respectively. The gating branch utilizes the SwiGLU activation:

$$\mathrm{SwiGLU}(\mathbf{x}) = \mathbf{W}_d\left(\mathrm{SiLU}(\mathbf{W}_g\mathbf{x}) \odot (\mathbf{W}_u\mathbf{x})\right), \quad (3)$$

with learnable matrices $\mathbf{W}_g, \mathbf{W}_u, \mathbf{W}_d$ and element-wise product $\odot$. Here, $\mathbf{x}$ is the input vector, and SiLU is the Sigmoid Linear Unit activation function.

Why tri-residual gating helps. The extra residual branch provides a direct pathway for "refusal-relevant" features to accumulate across layers without being washed out by attention/MLP mixing. Intuitively, the gate $\mathbf{g} = \mathrm{SiLU}(\mathbf{W}_g\mathbf{x})$ acts as a soft mask that is *input-conditioned*: it can suppress schema boilerplate while preserving tokens/positions carrying mismatch evidence. This is particularly suited to schema-conditioned prompts where the majority of tokens are irrelevant to the answerability decision.

After L_p TRGE layers, we aggregate token representations into a single vector and predict a scalar score (Eq. 4):

$$\mathbf{r} = \mathrm{Agg}\Big(\mathbf{Z}^{(L_p)}\Big), \qquad s = \mathbf{w}^\top \mathbf{r} + b, \qquad \hat{p} = \sigma(s). \tag{4}$$

where $\mathbf{r}$ is the pooled representation, $\mathbf{w}$ and b are the linear classifier's weight and bias, s is the scalar logit, and σ denotes the sigmoid function. We use a simple aggregation operator $\mathrm{Agg}(\cdot)$ (mean pooling unless otherwise noted), keeping the probe lightweight; Section 4 studies alternatives.

3.3 Training and Implementation Details

We freeze $\mathcal{M}$ entirely and train only the probe parameters ϕ. This yields three practical benefits: (i) stable behavior of the underlying generator, (ii) minimal additional compute and memory, and (iii) easy attachment to different base LLMs.

3.3.1 Supervision on hidden states

Given a labeled dataset $\mathcal{D} = \{(S_i, Q_i, y_i)\}_{i=1}^N$ where $y_i \in \{0, 1\}$ indicates whether the query is answerable under schema S_i, we build the model input via a template $\mathcal{T}$:

$$\mathbf{x}_i = \mathcal{T}(S_i, Q_i). \tag{5}$$

where $\mathbf{x}_i$ is the tokenized input sequence corresponding to schema S_i and question Q_i. We run a single forward pass through the frozen LLM and extract hidden states from layer $l^\star$:

$$\mathbf{H}_i = \mathbf{H}_i^{(l^\star)} = \mathcal{M}^{(l^\star)}(\mathbf{x}_i) \in \mathbb{R}^{T_i \times d}. \tag{6}$$

where T_i denotes the sequence length of the i-th sample. The probe predicts $\hat{p}_i = g_\phi(\mathbf{H}_i)$ and is trained on $\{(\mathbf{H}_i, y_i)\}_{i=1}^N$.

3.3.2 Numerical stability

During mixed-precision inference or offline hidden-state extraction, rare NaN values can destabilize optimization. We apply sanitization operator $\Psi(\cdot)$:

$$\tilde{\mathbf{H}}_i = \Psi(\mathbf{H}_i), \quad \Psi(h) = \begin{cases} 0, & h \text{ is NaN}, \\ C, & h = +\infty, \\ -C, & h = -\infty, \\ h, & \text{otherwise}, \end{cases} \tag{7}$$

with a large clipping constant C (e.g., 10^4), followed by token-wise normalization:

$$\mathbf{H}_i^{\text{safe}} = \mathrm{LN}(\tilde{\mathbf{H}}_i). \tag{8}$$

where $\tilde{\mathbf{H}}_i$ is the sanitized hidden state matrix, and $\mathbf{H}_i^{\text{safe}}$ denotes the final stabilized input for the probe. We feed $\mathbf{H}_i^{\text{safe}}$ to the probe in both training and evaluation for consistent behavior.

3.3.3 Optimization objective and thresholding

Let $s_i = f_\phi(\mathbf{H}_i^{\text{safe}})$ be the probe logit and $\hat{p}_i = \sigma(s_i)$. We optimize binary cross-entropy:

$$\mathcal{L}(\phi) = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)\Big). \tag{9}$$

where N is the number of samples in the training set. For a single example, the derivative w.r.t. the logit is:

$$\frac{\partial \ell}{\partial s} = \sigma(s) - y. \tag{10}$$

where ℓ represents the loss for a single instance. At deployment, we select the threshold τ on a development set to satisfy a desired safety–utility operating point (e.g., high refusal recall under a bounded false-refusal rate). This yields an explicit, auditable trade-off suitable for production gating.

LATENTREFUSAL is novel in (i) making refusal a *pre-generation* decision using *latent* signals from a frozen LLM, enabling single-pass, execution-free gating; and (ii) introducing TRGE, a tri-residual SwiGLU-gated probe tailored to schema-heavy Text-to-SQL prompts, designed to suppress schema noise and amplify sparse mismatch cues that drive unanswerability.

4 Experiments

We evaluate our TRGE Transformer probe for detecting unanswerability across four datasets and compare against representative baselines.

4.1 Experimental Setup

Datasets. We evaluate on four benchmarks: (1) **TriageSQL** (Zhang et al., 2020): converted into a binary refusal task focusing on medical intent; (2) **AMBROSIA** (Saparina and Lapata, 2024): testing sensitivity to linguistic ambiguity via paired clear vs. ambiguous questions; (3) **SQuAD 2.0** (Rajpurkar et al., 2018): used to evaluate cross-task generalizability from Text-to-SQL to machine reading comprehension; and (4) **MD-Enterprise**[1]: a Chinese vertical industrial benchmark spanning six domains (Stock, HR, etc.) with expert-annotated answerability labels based on business logic and safety constraints.

Models and Inference. We use Qwen-3-8B and Llama-3.1-8B (Yang et al., 2025; Grattafiori et al., 2024) as local backbones, loaded in bfloat16. For sampling-based baselines (Semantic Entropy, Eigenscore), we sample Top-$K = 10$ outputs at temperature $T = 0.7$; single-pass baselines (Self-evaluation, CCS, TSV, HaloScope, SAPLMA) use greedy decoding. For LATENTREFUSAL, we extract hidden states from a single greedy forward pass ($T = 0$). The probe is trained with a learning rate of 10^{-5}, batch size 8, no warm-up, and a fixed random seed of 42.

Training Protocol and Splits. We train one probe per dataset group. When multiple datasets are available within a single deployment, we merge them and train jointly. For instance, our internal MD-Enterprise dataset spans six domains (Stock, HR, Loan, Retail, Risk Control, Supervise), and joint training performs well across all of them; AMBROSIA likewise includes multiple ambiguity types and is trained jointly. In Table 1, each dataset is trained and evaluated separately. We split training/validation as 8:2 and sample 300 examples as the held-out test set for each dataset.

Baselines. We compare against three categories of methods: (1) **Output-based Uncertainty**: *Self-evaluation* (Kadavath et al., 2022) prompts the model to estimate its own correctness; *Semantic Entropy* (Farquhar et al., 2024) measures uncertainty via agreement across sampled generations. (2) **Internal-State Methods**: We evaluate unsupervised approaches *CCS* (Burns et al., 2023), *Eigenscore* (Chen et al., 2024), and weakly-supervised *HaloScope* (Du et al., 2024), alongside supervised probes *SAPLMA* (Azaria and Mitchell, 2023) and *TSV* (Park et al., 2025) which predict truthfulness from hidden states or steering vectors. (3) **Prompting**: *DeepSeek-Chat* serves as a zero-shot instruction-following baseline.

[1]This is an internal dataset and cannot be released due to privacy reasons.

4.2 Main Results

Table 1 summarizes the refusal detection performance across four benchmarks. We report F1 as the primary metric, computed at a fixed decision threshold tuned on development data.

Overall performance. LATENTREFUSAL achieves the highest average F1 on both backbone models: **88.5%** on Llama-3.1-8B and **88.8%** on Qwen-3-8B, outperforming all baselines by substantial margins. Specifically, it surpasses the strongest baseline—SAPLMA (84.6% on Qwen-3-8B) and TSV/SAPLMA (82.9% on Llama-3.1-8B)—by +4.2 and +5.6 points respectively. Notably, against the internal-state baseline Eigenscore, the gains exceed +14 points, demonstrating that the TRGE probe architecture more effectively distills refusal-relevant signals from hidden states than unsupervised spectral methods.

Robustness to Linguistic Ambiguity. The AMBROSIA dataset evaluates sensitivity to underspecified queries with subtle linguistic ambiguity. Here, LATENTREFUSAL achieves **80.2%** (Llama), outperforming Semantic Entropy by **18.1 points**. This significant gap may partly stem from a known challenge of sampling-based uncertainty in Text-to-SQL, where syntactic diversity can mask semantic overconfidence. In our implementation, we approximate equivalence using syntactic agreement (Appendix C.1) for safety and cost reasons, which may underestimate the best-case performance of semantic clustering. Our latent-signal approach avoids this limitation by detecting the underlying representation-level confusion before it manifests as overconfident decoding.

Training Efficiency. Instead of relying on zero-shot transfer, LATENTREFUSAL attains its high performance through highly efficient supervised adaptation. Our method can be trained on any dataset using only $\sim$300 samples, completing in just 10 minutes on a single A100-80G GPU. This efficiency allows custom refusal gates to be deployed rapidly for new domains (e.g., reaching **88.6%** F1 on SQuAD 2.0 and **87.1%** F1 on TriageSQL) with minimal data annotation and computational cost.

Comparison with API-based prompting. The DeepSeek-Chat API baseline uses zero-shot prompting to elicit refusal. While it achieves reasonable performance (83.2% avg.), it underperforms LATENTREFUSAL by 5+ points

LLM	Method	MD-Enterprise	AMBROSIA	SQuAD	TriageSQL	Avg.F1
Llama-3.1-8B	Semantic Entropy	66.1	62.1	82.3	66.7	69.3
	CCS	53.1	54.1	62.6	62.9	58.2
	Self-evaluation*	66.7	64.6	74.2	67.2	68.2
	Eigenscore	82.5	63.2	72.4	77.8	73.8
	TSV	97.4	74.3	74.7	85.2	82.9
	HaloScope	97.0	73.7	66.1	82.8	79.9
	SAPLMA*	97.5	77.7	75.4	81.0	82.9
	LATENTREFUSAL	**99.6**	**80.2**	**86.6**	**87.7**	**88.5**
Qwen-3-8B	Semantic Entropy	72.7	58.0	82.4	66.6	70.0
	CCS	55.4	47.0	82.3	73.4	64.5
	Self-evaluation*	68.1	60.2	87.4	56.3	68.0
	Eigenscore	80.0	59.1	70.0	77.8	71.7
	TSV	98.9	73.3	78.0	85.0	83.8
	HaloScope	98.0	72.8	80.1	80.0	82.7
	SAPLMA*	97.8	81.2	76.8	82.6	84.6
	LATENTREFUSAL	**98.8**	**80.9**	**88.6**	**87.1**	**88.8**
DeepSeek-Chat (API)	Prompt-based	97.2	70.3	87.4	77.8	83.2

Table 1: Refusal detection performance (F1 %) across four benchmarks: MD-Enterprise (Chinese financial QA, 6 domains), AMBROSIA (linguistic ambiguity), SQuAD 2.0 (cross-task reading comprehension), and TriageSQL (medical intent). Local backbone models (Llama-3.1-8B and Qwen-3-8B) are loaded in bfloat16 with max sequence length 2048; DeepSeek-Chat is accessed via API. Sampling-based baselines (Semantic Entropy, Eigenscore) use K=10 samples at T=0.7; Self-evaluation uses a single-pass logit score; LATENTREFUSAL uses a single greedy forward pass. Best results per backbone are in **bold**. Methods marked with * are reproduced by us; all others use official implementations.

and lacks controllability—prompt-based refusal is sensitive to instruction phrasing and can fail precisely when the model hallucinates. Our internal-signal approach provides a more reliable and deterministic safety gate.

4.3 Efficiency Analysis

Table 2 compares inference latency (Qwen-3-8B backbone). LATENTREFUSAL adds negligible overhead (**2ms**), achieving 54ms total latency—**13.7× faster** than Semantic Entropy (740ms). While sampling methods are prohibitive for real-time use, LATENTREFUSAL achieves a favorable accuracy–latency trade-off and lies on the Pareto frontier among the evaluated baselines (Table 1). Specifically, it attains the highest F1 (88.8%) at near-minimal latency, superior to both computationally expensive spectral methods and similarly fast but less accurate supervised baselines (TSV).

Architectural Efficiency. Efficiency stems from three factors: (1) **Zero-redundancy extraction**: reusing mandatory forward-pass states adds no LLM-level compute; (2) **Parameter efficiency**: the 19M-parameter TRGE probe is $< 0.3\%$ of the backbone size; (3) **Early-exit capability**: deciding refusal *before* decoding saves generation costs for unanswerable queries.

Method	# Runs	Main Latency(ms)
Semantic Entropy	N=10	52*10
CCS	2	100
Self-evaluation*	1	53
Eigenscore	$N = 5/10$	$51.7 * N$
TSV	1	52
HaloScope	1	50
SAPLMA*	1	53
LATENTREFUSAL	1	54

Table 2: Inference latency comparison on the Qwen-3-8B backbone (single A100-80G GPU, bfloat16, sequence length 2048). "# Runs" denotes the number of LLM forward passes required. Sampling-based methods (Semantic Entropy, Eigenscore) scale linearly with the number of samples N. LATENTREFUSAL requires only one forward pass plus a 2ms probe, achieving the best accuracy–latency trade-off on the Pareto frontier.

4.4 Analysis

The Failure of Spectral Uncertainty. Eigenscore relies on spectral statistics of hidden states across multiple samples. However, in Text-to-SQL, "confident hallucinations" are common: the model may generate syntactically diverse SQL (e.g., varying 'JOIN' orders or alias names) that are semantically identical. This diversity inflates eigenvalue spread, causing spectral methods to misinterpret syntactic variance as epistemic uncertainty. TRGE avoids this by learning to identify the *source* of mismatch in the prompt-schema representation, which is invariant to the downstream decoding path.

Cross-Backbone Consistency. The comparable performance between Llama-3.1-8B (88.5%) and Qwen-3-8B (88.8%) is consistent with the hypothesis that refusal signals may be encoded in a structurally similar manner across modern Transformer architectures. This suggests that the TRGE architecture is not overfitted to a specific model's quirks, and implies that the probe may capture a task-general signal regarding how LLMs process unanswerable context.

Error Analysis and Future Work. Qualitative inspection reveals two primary failure modes: (1) **Semantic Near-Misses**: when a column name is semantically similar but logically incorrect (e.g., 'revenue' vs. 'gross_profit'), the probe occasionally underestimates uncertainty. (2) **Deep Reasoning Chains**: queries requiring complex multi-hop joins sometimes exhibit weak mismatch signals in the selected layer. Future work could explore *multi-layer fusion* or *schema-aware attention* to better capture these high-order logical inconsistencies.

4.5 Ablation Studies

We conduct comprehensive ablation studies on Qwen-3-8B using the TriageSQL dataset to validate the architectural decisions of LATENTREFUSAL. Additional ablations on probe depth and training strategies are provided in Appendix B.

4.5.1 Architecture and Layer Selection

We validate the TRGE probe design and investigate the optimal source of refusal signals.

Efficacy of the TRGE Architecture. Table 3 isolates the impact of the Tri-Residual Gated Encoder. The full TRGE model achieves an F1 score of **87.1%**. Removing the gating branch (*w/o SwiGLU*) degrades performance to 85.4%, but the most critical insight comes from replacing the SwiGLU gate with a standard MLP (−4.1% F1) or a Linear Probe (−16.7% F1). The failure of the Linear Probe (70.4% F1) confirms that refusal detection requires modeling complex non-linear interactions between question and schema. Moreover, replacing SwiGLU with a standard MLP drops performance to 83.0%, demonstrating that the *gating mechanism*—which selectively suppresses schema noise—is essential. Among gating variants, SwiGLU outperforms GLU and GeGLU, likely due to its smoother optimization landscape.

Variant	F1 (%)	Time (ms)
TRGE (Full)	**87.1**	2.6
w/o SwiGLU	85.4	2.3
SwiGLU → MLP	83.0	2.2
SwiGLU → GLU	75.5	2.3
SwiGLU → GeGLU	85.1	2.0
Linear Probe	70.4	**0.8**

Table 3: Architecture ablation on Qwen-3-8B / TriageSQL (layer −16, 4 probe layers, dropout 0.2). We compare the full TRGE against five variants: removing the SwiGLU branch, replacing it with MLP/GLU/GeGLU gates, and a linear probe baseline. "Time" reports the probe-only latency (excluding LLM forward pass). The SwiGLU gate contributes +16.7% F1 over the linear probe and +4.1% over the MLP replacement.

Layer	Acc	Prec	Rec	AUC	F1
−1	84.4	76.7	99.0	87.6	86.5
−8	84.5	**77.4**	97.8	**88.7**	86.4
−16	**85.0**	77.2	**99.8**	88.4	**87.1**
−24	84.4	76.4	**99.8**	87.7	86.5
−32	82.9	74.7	**99.8**	88.4	85.4

Table 4: Hidden state layer selection ablation on Qwen-3-8B / TriageSQL. Layer indices are relative to the final layer (−1 = last, −32 = first). The TRGE probe (4 layers, dropout 0.2) is trained separately for each source layer. Layer −16 (middle of the 32-layer backbone) achieves the best F1 (87.1%) and near-perfect recall (99.8%), supporting the hypothesis that refusal signals peak at intermediate layers before being collapsed by later processing.

Locating Refusal Signals. Table 4 shows that the optimal refusal signal resides in the middle-to-late layers (Layer −16), achieving the highest Accuracy (85.0%) and F1 (87.1%). While Layer −8 offers slightly higher precision, Layer −16 provides a superior balance with near-perfect recall (99.8%). Notably, the final layer (−1) exhibits lower recall (99.0%) than deeper layers. This finding supports the "mechanistic interpretability" hypothesis that LLMs encode epistemic uncertainty about unanswerable queries in intermediate processing stages, which may be resolved—or collapsed into confident hallucinations—by the time representations reach the final output layer (Marks and Tegmark, 2024; Kossen et al., 2024).

Probe Depth and Capacity. Table 5 reveals an inverted-U relationship between probe depth and detection accuracy. Performance peaks at 4 layers (87.1% F1). Shallower probes (1–2 layers) underfit the data, while deeper probes (8–12 layers) show diminishing returns and signs of overfitting. This suggests that a compact 4-layer probe is sufficient to extract the refusal signal without memorizing dataset-specific schema artifacts, validating our design goal of a lightweight, low-latency module.

Layers	Params	F1 (%)	Time (ms)
1	9.6M	82.03	1.04
2	12.7M	83.87	1.64
4	19.0M	**87.09**	2.60
6	25.3M	85.28	3.40
8	31.7M	84.61	4.45
12	44.3M	83.02	6.46

Table 5: Probe depth ablation on Qwen-3-8B / TriageSQL (layer -16, SwiGLU gate, dropout 0.2). We vary the number of TRGE layers from 1 to 12 and report F1 and probe-only inference time. Performance peaks at 4 layers (19.0M params, 2.60ms), with deeper probes showing diminishing returns due to overfitting on the 300-sample training set.

Robustness to Training Strategy. Our method is robust to hyperparameter variations (Table 6 and 7 in Appendix). Label Smoothing ($\epsilon = 0.1$) achieves optimal performance (87.1% F1) by mitigating overconfidence. Additionally, stability across varying dropout rates (0.0–0.3) indicates the probe learns distributed features rather than brittle artifacts.

Summary of ablations. Our ablation studies confirm that: (1) the SwiGLU gating mechanism is critical, with a 16.7% F1 gap between TRGE and linear probes; (2) intermediate LLM layers provide the best refusal signals; and (3) training is robust to standard hyperparameter variations.

5 Conclusion

We address the critical safety risk of unanswerable queries in Text-to-SQL, where hallucinations yield harmful executable code. We formalize refusal as a strict *pre-generation gating* problem and propose LATENTREFUSAL, a single-pass mechanism that detects answerability directly from a frozen LLM's internal states. By introducing the Tri-Residual Gated Encoder (TRGE), we effectively amplify sparse refusal signals amidst schema noise, avoiding the latency and risks of execution-based uncertainty estimation. Empirical results confirm that LATENTREFUSAL provides a robust, low-latency safety layer essential for production-grade Text-to-SQL systems.

Limitations

While LATENTREFUSAL achieves high detection accuracy, its generalization capability across disparate domains remains a limitation; the probe currently requires fine-tuning to adapt to specific deployment scenarios. However, this need for adaptation is offset by the method's exceptional training efficiency. The probe introduces minimal computational overhead, with training converging in approximately 10 minutes on a single NVIDIA A100-80G GPU. This low-cost training profile makes it practical to re-train or fine-tune the refusal mechanism for new verticals without significant resource expenditure. Future research will explore cross-dataset generalization and universal safety gating—where a single probe can reliably gate any Text-to-SQL deployment without domain-specific re-tuning—through techniques such as domain-invariant representation learning and meta-learning.

Ethics Statement

This work does not involve human subjects or personally identifiable information. The MD-Enterprise dataset is used under institutional agreement and cannot be released due to privacy constraints. All other datasets are publicly available benchmarks.

Acknowledgements

This work was supported by the Yangtze River Delta Science and Technology Innovation Community Joint Research Project (YDZX20233100004031), the "Internet Financial Customer Service Vertical Large Model Construction Project" (Contract Registration No. 2026310031000213), and the Gientech Enterprise Cooperation Project on Financial AI Product Safety Monitoring and Intelligent Processing of Massive Transaction Data (GIFD20222025).

References

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. *arXiv preprint arXiv:2304.13734*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering Latent Knowledge in Language Models without Supervision. In *International Conference on Learning Representations (ICLR)*.

Chen Chen, Kexin Liu, Zepu Chen, Yanzhe Gu, Yijie Wu, Ming Tao, Zhaowei Fu, and Jieping Ye. 2024. INSIDE: LLMs' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*.

Qian Cheng, Ting Sun, Xin Liu, Wenge Zhang, Zhibin Yin, Si Li, Lizhen Li, Zhilin He, Kai Chen, and Xipeng Qiu. 2024. Can AI assistants know what they don't know? *arXiv preprint arXiv:2401.13275*.

C. K. Chow. 1970. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46.

Mingwen Dong, N. Anand Kumar, Yushi Hu, Saurabh Kaushik, Dongmei Song, Cho-Jui Hsieh, and Jinyang Li. 2024. PRACTIQ: A practical conversational text-to-SQL dataset with ambiguous and unanswerable queries. *arXiv preprint arXiv:2410.11076*.

Xiang Du, Chen Xiao, and Yang Li. 2024. HaloScope: Harnessing Unlabeled LLM Generations for Hallucination Detection. *arXiv preprint arXiv:2409.17504*.

Ran El-Yaniv and Yair Wiener. 2010. On the Foundations of Noise-free Selective Classification. *Journal of Machine Learning Research*, 11:1605–1641.

Sebastian Farquhar, Jannik Kossen, Lennart Kuhn, and Yarin Gal. 2024. Detecting Hallucinations in Large Language Models Using Semantic Entropy. *Nature*, 630(8017):625–630.

Jing Gao, Biye Wang, Yuchen Li, Youtao Zhang, and Rui Wang. 2024. Text-to-SQL empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, 17(11):2842–2855.

Yonatan Geifman and Ran El-Yaniv. 2017. Selective Classification for Deep Neural Networks. In *Advances in Neural Information Processing Systems*.

Yonatan Geifman and Ran El-Yaniv. 2019. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. In *International Conference on Machine Learning*.

Alexandre Grattafiori, Ayush Dubey, Ayush Jauhri, Ayush Pandey, Ayush Kadian, Alex Letman, Ankit Mathur, Armen Schelten, Arthur Vaughan, Bowen Yang, Angela Fan, Ankit Goyal, Asa Hartshorn, and Zheng Ma. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihua Zhong, Zhenxin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng He, Sakit Li, Tianyun Zhu, and Binghuai Lin. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv preprint arXiv:2311.05232*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38.

Saurav Kadavath, Thomas Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language Models (Mostly) Know What They Know. *arXiv preprint arXiv:2207.05221*.

Jannik Kossen, Jiarui Han, Mohammed Razzak, Lisa Schut, Shreshth A Malik, and Yarin Gal. 2024. Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs. *arXiv preprint arXiv:2406.15927*.

Lennart Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation. In *International Conference on Learning Representations (ICLR)*.

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

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2024. A survey of Text-to-SQL in the era of LLMs. *arXiv preprint arXiv:2408.05109*.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Samuel Marks and Max Tegmark. 2024. The Geometry of Truth: Emergent Linear Structure in Large Language Model Representations of True/False Datasets. In *Conference on Language Modeling (COLM)*.

Sung-Min Park, Xue-Ying Du, Ming-Chang Yeh, Hsin-Wei Wang, and Yu-Chiang Li. 2025. Steer LLM Latents for Hallucination Detection. *arXiv preprint arXiv:2503.01917*.

Bing Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binyi Li, Ruiying Geng, Rong Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A survey on Text-to-SQL parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jie Ren, Jing Luo, Yuxin Zhao, Kalpesh Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J. Liu. 2023. Out-of-Distribution Detection and Selective Generation for Conditional Language Models. In *International Conference on Learning Representations (ICLR)*.

P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. *arXiv preprint arXiv:2402.07927*.

Irina Saparina and Mirella Lapata. 2024. AMBROSIA: A benchmark for parsing ambiguous questions into database queries. *arXiv preprint arXiv:2406.19073*.

O. Skean, M. R. Arefin, D. Zhao, N. Patel, J. Naghiyev, and Y. LeCun. 2025. Layer by Layer: Uncovering Hidden Representations in Language Models. *arXiv preprint arXiv:2502.02013*.

Ruoxi Sun, Soumya Sanyal, Raymond Cheng, Li Bo, and Xifeng Yan. 2024. SQL-PaLM: Improved large language model for Text-to-SQL. *Transactions on Machine Learning Research*.

Y. M. Tsai, T. Cojean, and H. Anzt. 2020. Evaluating the performance of NVIDIA's A100 Ampere GPU for sparse linear algebra computations. *arXiv preprint arXiv:2008.08478*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*.

Bowen Wang, Yuxuan Gao, Zhen Li, and Jian-Guang Lou. 2022. Know What I don't know: Handling Ambiguous and Unanswerable Questions for Text-to-SQL. *arXiv preprint arXiv:2212.08902*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations (ICLR)*.

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

An Yang, An Li, Bin Yang, Botao Zhang, Binyuan Hui, Bosheng Zheng, Bowen Yu, Chao Gao, Chen Huang, Cheng Lv, Chuan Zheng, Dai Liu, Feng Zhou, Fei Huang, Fu Hu, Hao Ge, Hong Wei, Hongyi Lin, Jian Tang, and Zhaoye Qiu. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yutao Zhang, Xinyi Dong, Shuaichen Chang, Tao Yu, Peng Shi, and Rui Zhang. 2020. Did You Ask a Good Question? A Cross-Domain Question Intention Classification Benchmark for Text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xinyang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shubhangi Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and Dan Hendrycks. 2023. Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv preprint arXiv:2310.01405*.

Appendix

Appendix Contents

A Prompt Template

We include the prompt templates used for answerability judgment (Chinese and English).

Prompt Template (Chinese)

现在是2025年，你是一个会写sql的机器，你不具备数据分析能力。不要对开放问题给出你的看法。
你的任务是以最严格的标准判断：用户的问题是否为SQL查询需求，是否有确定的计算逻辑，能否通过生成并执行sql计算，在给定的数据库范围内，得出确定的答案。
{database_meta}
User Query:
{user_query}
要求：仔细分析用户输入和数据库信息，按照以下例子进行回答。
可以回答的问题具备以下特征：
1. 查询具体的历史数据点，如某个时间点的价格、交易量等
2. 对历史数据进行简单的统计计算，如平均值、总和、最大/最小值等
3. 查找满足特定条件的记录，如超过某个阈值的数据
4. 查询实体的基本属性或标识信息，如代码、名称等
5. 在特定时间范围内进行以上操作
不能回答的问题特征：
1. 没有具体、直接的计算逻辑的。比如：如何分析员工的职位晋升?
2. 涉及未来预测或趋势判断，比如：未来员工会不会离职?
3. 需要主观分析或评估，比如：员工的工作能力如何?
4. 需要数据库之外的其他信息
5. 开放性问题或建议性问题，比如：如何评价员工的绩效情况?
6. 涉及决策指导
7. 需要解释因果关系
8. 需要实时或动态数据
9. 需要深度分析或复杂模型
判断时应该考虑：问题是否有明确的答案，以及是否能仅通过已有的历史数据得出这个答案。如果问题模糊或需要额外信息和分析，则应判定为不能回答。
以json格式输出：

```
{
    "label": boolean,
    "tables": [
        {
            "table_name": string,
            "fields": [string]
        }
    ],
    "reason": string
}
```

English version.

Prompt Template (English)

It is 2025. You are a machine that writes SQL. You do not have data-analysis capability. Do not provide opinions for open-ended questions.
Your task is to judge, with the strictest standard, whether the user's question is a SQL-query request, whether it has a well-defined computational logic, and whether a definite answer can be obtained

by generating and executing SQL within the given database scope.
{database_meta}
User Query:
{user_query}
Instructions: Carefully analyze the user query and the database information, and answer following the examples below.
Answerable questions usually have the following characteristics:
1. Query a specific historical data point, e.g., price or volume at a certain time.
2. Perform simple statistical computations on historical data, e.g., average, sum, max/min.
3. Retrieve records that satisfy specific conditions, e.g., values above a threshold.
4. Query basic attributes or identifiers of an entity, e.g., code or name.
5. Perform the above within a specified time range.
Unanswerable questions usually have the following characteristics:
1. No concrete and direct computational logic (e.g., How to analyze employees' promotion paths?).
2. Future prediction or trend judgment (e.g., Will the employee resign in the future?).
3. Subjective analysis or evaluation (e.g., How is the employee's work capability?).
4. Require information beyond the database.
5. Open-ended or advice-seeking questions (e.g., How to evaluate the employee's performance?).
6. Decision-making guidance.
7. Require causal explanation.
8. Require real-time or dynamic data.
9. Require deep analysis or complex models.
When judging, consider whether the question has a clear answer and whether the answer can be derived solely from the existing historical data. If the question is vague or requires additional information and analysis, it should be judged as unanswerable.
Output in JSON format:

```
{
    "label": boolean,
    "tables": [
        {
            "table_name": string,
            "fields": [string]
        }
    ],
    "reason": string
}
```

B Additional Ablation Studies

B.1 Robustness to Training Strategy

Loss Function	F1 (%)	AUC (%)
Label Smoothing ($\epsilon = 0.1$)	**87.1**	**88.7**
Label Smoothing ($\epsilon = 0.05$)	85.1	88.4
Focal Loss ($\gamma = 2$)	85.3	88.3
Focal Loss ($\gamma = 1$)	85.2	86.9
BCE Loss	84.8	87.8

Table 6: Loss function ablation on Qwen-3-8B / TriageSQL (4-layer TRGE, layer −16, dropout 0.2). Label smoothing with ϵ=0.1 yields the best F1 (87.1%) and AUC (88.7%), outperforming standard BCE by +2.3% F1, likely by mitigating overconfident predictions near the decision boundary.

Dropout Rate	F1 (%)	AUC (%)
Dropout = 0.0	85.4	88.0
Dropout = 0.1	86.5	88.4
Dropout = 0.2 (default)	**87.1**	**88.7**
Dropout = 0.3	85.5	88.0

Table 7: Dropout rate ablation on Qwen-3-8B / TriageSQL (4-layer TRGE, layer -16, label smoothing ϵ=0.1). Performance is stable across rates 0.1–0.3, peaking at 0.2 (87.1% F1). The narrow variance ($\pm$1.6% F1) indicates the probe learns distributed rather than brittle features.

C Baseline Experimental Settings

We evaluate all baselines on Qwen-3-8B and Llama-3.1-8B backbones using bfloat16 precision, with a maximum sequence length of 2048 tokens.

C.1 Implementation Details

- **Self-Evaluation (P(True)):** We convert the question and schema into a declarative statement and prompt the model to judge its correctness using a True/False format. The logit of the "True" token is used as the answerability score. We use 4-shot prompting (2 answerable, 2 unanswerable) for calibration.
- **Semantic Entropy:** We sample $K = 10$ outputs ($T = 0.7$) and cluster them into semantic equivalence classes using an NLI model. For Text-to-SQL, refusal responses are treated as equivalent, while SQL queries are compared based on syntactic agreement. The entropy over equivalence classes serves as the uncertainty score.
- **CCS (Contrast-Consistent Search):** We construct contrast pairs (e.g., "Is correct? Yes/No") and extract hidden states from the last token of the final layer. An unsupervised linear probe is trained to satisfy consistency and informative constraints.
- **SAPLMA:** We extract the hidden states of the last token across all layers. A multi-layer perceptron (MLP) classifier ($4096 \to 256 \to 128 \to 64 \to 1$) is trained on a specific layer, selected via grid search on the validation set.
- **HaloScope:** We leverage an exemplar set to construct a hallucination-related subspace using Singular Value Decomposition (SVD). Answerability is predicted by training a classifier on the projection scores within this subspace.
- **TSV (Truthfulness Separator Vector):** We learn a separator direction in the latent space through consistency constraints. The answerability score is derived from prototype similarity after activation intervention.
- **Eigenscore:** We analyze the spectral statistics of hidden states across multiple samples ($N \geq 5, T = 0.7$). The eigenvalue distribution of the activation covariance matrix is used to estimate epistemic uncertainty.
- **API-based Prompting:** We use DeepSeek-Chat with zero-shot instructions to directly judge query answerability. Refusal signals are extracted from the text output.

D Algorithm Pseudocode

Note: Algorithms 4 and 5 describe optional extensions for combined hallucination detection and multi-domain adaptation, which are not part of the main results reported in Table 1.

D.1 Hidden State Extraction Framework

D.2 TRGE Probe Architecture

D.3 Training Procedure

Algorithm 1 Hidden State Extraction for Answerability/Refusal Detection

Require: Dataset $\mathcal{D} = \{(q_i, c_i, y_i)\}_{i=1}^N$, LLM $\mathcal{M}$, Layer index ℓ
Ensure: Hidden state dataset $\mathcal{H} = \{(\mathbf{H}_i, y_i)\}_{i=1}^N$

Initialize empty dataset $\mathcal{H} \leftarrow \emptyset$
for each $(q_i, c_i, y_i) \in \mathcal{D}$ **do**
  $\mathbf{x}_i \leftarrow \textsc{Tokenize}(\textsc{Prompt}(q_i, c_i))$
  $\{\mathbf{H}^{(l)}\}_{l=0}^L \leftarrow \mathcal{M}(\mathbf{x}_i)$ {Forward pass with hidden states}
  $\mathbf{H}_i \leftarrow \mathbf{H}^{(\ell)} \in \mathbb{R}^{T \times d}$ {Extract layer ℓ}
  if $\textsc{HasNaN}(\mathbf{H}_i)$ **then**
    $\mathbf{H}_i \leftarrow \textsc{NanToNum}(\mathbf{H}_i)$ {Numerical stability}
  end if
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\mathbf{H}_i, y_i)\}$
end for
return $\mathcal{H}$

Algorithm 2 TRGE Probe for Answerability/Refusal Detection

Require: Hidden state $\mathbf{H} \in \mathbb{R}^{T \times d_{in}}$, Padding mask $\mathbf{M} \in \{0, 1\}^T$
Ensure: Answerability probability $\hat{p} \in [0, 1]$

$\mathbf{H} \leftarrow \textsc{LayerNorm}(\mathbf{H})$ {Input Projection}
$\mathbf{Z} \leftarrow \text{GELU}(\textsc{LayerNorm}(\mathbf{H}\mathbf{W}_{proj} + \mathbf{b}_{proj}))$ {$\mathbf{Z} \in \mathbb{R}^{T \times d}$}
$\mathbf{Z} \leftarrow \mathbf{Z} + \textsc{LearnablePE}(T)$ {Positional Encoding}
for $n = 1$ to N **do**
  $\mathbf{U} \leftarrow \mathbf{Z} + \textsc{MultiHeadAttn}(\text{LN}(\mathbf{Z}), \text{LN}(\mathbf{Z}), \text{LN}(\mathbf{Z}), \mathbf{M})$ {Residual 1: Attention}
  $\mathbf{V} \leftarrow \mathbf{U} + \text{GELU}(\text{LN}(\mathbf{U})\mathbf{W}_1)\mathbf{W}_2$ {Residual 2: FFN}
  $\mathbf{Z} \leftarrow \mathbf{V} + \textsc{SwiGLU}(\text{LN}(\mathbf{V}))$ {Residual 3: Gating}
end for
$L_{valid} \leftarrow \sum_{t=1}^T (1 - M_t)$ {Mean Pooling}
$\mathbf{z}_{pool} \leftarrow \frac{1}{L_{valid}} \sum_{t=1}^T (1 - M_t)\, \mathbf{Z}_t$
$\hat{p} \leftarrow \sigma(\text{GELU}(\mathbf{z}_{pool}\mathbf{W}_{c1})\mathbf{W}_{c2})$ {Classification Head}
return $\hat{p}$

Algorithm 3 Distributed Training with F1-based Selection

Require: Training set $\mathcal{H}_{train}$, Validation set $\mathcal{H}_{val}$, Epochs E
Ensure: Trained parameters θ^*

Initialize $\theta, F_1^{best} \leftarrow 0, \theta^* \leftarrow \theta$
for $e = 1$ to E **do**
  for each mini-batch $\{(\mathbf{H}_i, y_i)\}_{i=1}^B \in \mathcal{H}_{train}$ **do**
    $\mathcal{L} \leftarrow -\frac{1}{B} \sum_{i=1}^B [y_i \log f_\theta(\mathbf{H}_i) + (1 - y_i) \log(1 - f_\theta(\mathbf{H}_i))]$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
  end for
  $F_1 \leftarrow \text{F1}\textsc{Score}(\textsc{Evaluate}(f_\theta, \mathcal{H}_{val}))$
  if $F_1 > F_1^{best}$ **then**
    $F_1^{best} \leftarrow F_1, \theta^* \leftarrow \theta$
  end if
end for
return θ^*

D.4 Refusal-Aware Hallucination Detection

Algorithm 4 Refusal-Aware Query Classification

Require: Query q, Context c, LLM $\mathcal{M}$, Probe f_θ, Thresholds τ_r, τ_h
Ensure: Decision $\in \{\text{ANSWER}, \text{REFUSE}, \text{HALLUCINATION}\}$

$p_{ans} \leftarrow f_\theta^{ref}(\mathcal{M}(\text{TOK}(\text{REFPROMPT}(q, c)))^{(\ell)})$
if $p_{ans} < \tau_r$ **then**
  return REFUSE
end if
$r \leftarrow \mathcal{M}.\text{GENERATE}(\text{TOK}(\text{ANSPROMPT}(q, c)))$
$p_{hal} \leftarrow f_\theta^{hal}(\mathcal{M}(\text{TOK}(\text{ANSPROMPT}(q, c)) \oplus r)^{(\ell)})$
if $p_{hal} > \tau_h$ **then**
  return HALLUCINATION
else
  return ANSWER
end if

D.5 Multi-Domain Refusal Training

Algorithm 5 Multi-Domain Refusal Training

Require: Domain datasets $\{\mathcal{D}_k\}_{k=1}^K$, LLM $\mathcal{M}$, Layer ℓ
Ensure: Domain-specific probes $\{f_{\theta_k}\}_{k=1}^K$

for each domain $k \in \{1, \ldots, K\}$ **do**
  $\mathcal{H}_k \leftarrow \text{EXTRACTHIDDENSTATES}(\mathcal{D}_k, \mathcal{M}, \ell)$
  $\theta_k \leftarrow \text{TRAINPROBE}(\mathcal{H}_k)$ {Algorithm 3}
  $F_1^{(k)} \leftarrow \text{EVALUATE}(f_{\theta_k}, \mathcal{H}_k^{test})$
end for
return $\{f_{\theta_k}\}_{k=1}^K$

D.6 Model Configuration

Table 8: TRGE Probe Hyperparameters

Hyperparameter	Value
Input dimension (d_{in})	4096
Model dimension (d)	512
Number of attention heads	8
Number of TRGE layers (N)	4
SwiGLU intermediate dimension	2048
Feed-forward dimension	4096
Dropout rate	0.2
Maximum sequence length	8192
Epochs	20
Optimizer	AdamW

E Extended Case Study

We provide an extended case study to demonstrate LATENTREFUSAL's behavior in a realistic financial analysis scenario. Figure 3 illustrates the model's response to two distinct types of user queries.

The two queries shown in Figure 3 are translated as follows:

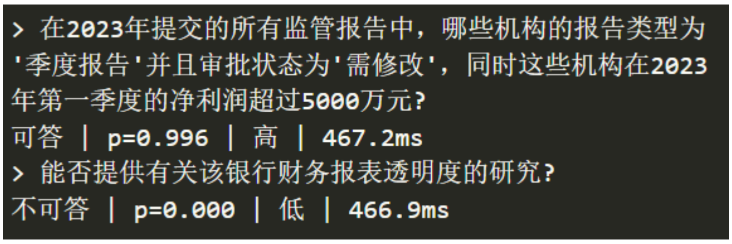

Figure 3: Running screenshot of LatentRefusal in a production Chinese financial QA deployment (NVIDIA RTX 4090, Qwen-3-8B backbone; hyperparameters optimized for production). **Top:** a multi-constraint regulatory query is correctly identified as answerable (p=0.996). **Bottom:** a subjective, out-of-scope request about "financial statement transparency" is correctly refused (p=0.000). End-to-end inference latency is stable at ≈467ms regardless of the gating decision. English translations of the Chinese queries are provided below.

- **Query 1 (Answerable,** p=0.996**):** "Among all regulatory reports submitted in 2023, which institutions have a report type of 'Quarterly Report' and an approval status of 'Requires Revision', whilst also having a net profit exceeding 50 million yuan in Q1 2023?"
- **Query 2 (Unanswerable,** p=0.000**):** "Can you provide information regarding the transparency of the bank's financial statements?"

In the first example (top), the user asks a highly specific question with multiple filtering conditions ("quarterly reports", "status needs revision", "net profit > 50 million"). Despite the syntactic complexity, LatentRefusal detects strong grounding between the query constraints and the database schema, assigning a high answerability probability ($p = 0.996$). This demonstrates that the probe is not easily confused by query length or logical depth.

In the second example (bottom), the user asks for "research on financial statement transparency" regarding a specific bank. This is a subjective, open-ended request that cannot be resolved by a structured SQL query against the bank's transactional or reporting database. Baseline models often hallucinate SQL queries that retrieve loosely related text fields (e.g., `remarks` columns). In contrast, LatentRefusal identifies the lack of schema alignment for the abstract concept of "research" and correctly predicts the query as unanswerable ($p = 0.000$), effectively serving as a safety gate.

Notably, the inference latency remains consistent ($\approx$ 467ms) regardless of the decision, confirming the efficiency of the single-pass architecture.