Confidence Scoring for LLM-Generated SQL in Supply Chain Data Extraction

Jiekai Ma
AWS Supply Chain
Seattle, Washington, USA
jiekal@amazon.com

Yikai Zhao
AWS Supply Chain
Austin, Texas, USA
yikai@amazon.com

Abstract

Large Language Models (LLMs) have recently enabled natural language interfaces that translate user queries into executable SQL, offering a powerful solution for non-technical stakeholders to access structured data. However, one of the limitation that LLMs do not natively express uncertainty makes it difficult to assess the reliability of their generated queries. This paper presents a case study that evaluates multiple approaches to estimate confidence scores for LLM-generated SQL in supply chain data retrieval. We investigated three strategies: (1) translation-based consistency checks; (2) embedding-based semantic similarity between user questions and generated SQL; and (3) self-reported confidence scores directly produced by the LLM. Our findings reveal that LLMs are often overconfident in their own outputs, which limits the effectiveness of self-reported confidence. In contrast, embedding-based similarity methods demonstrate strong discriminative power in identifying inaccurate SQL.

ACM Reference Format:
Jiekai Ma and Yikai Zhao. 2025. Confidence Scoring for LLM-Generated SQL in Supply Chain Data Extraction. In *Proceedings of the 1st Workshop on "AI for Supply Chain: Today and Future" @ 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 Introduction

Supply chain analysis frequently requires querying large-scale databases to extract insights related to inventory, demand trends, logistics, and supplier performance. Recent advances in Large Language Models (LLMs) have enabled natural language interfaces that allow non-technical users to pose complex questions and receive SQL queries to query databases automatically. While this represents a significant leap in usability and accessibility, it introduces a critical challenge: trust. In the supply chain context, user queries often involve intricate constraints and domain-specific semantics. For example, a question like *"Which supplier had the fastest average delivery time for Category X items in the past year?"* requires reasoning and accurate mapping to structured schema elements. LLMs operating in this setting must translate natural language into syntactically valid and semantically precise SQL. These challenges are compounded by the dynamic nature of supply chain databases, where schema variations or out-of-scope queries (e.g., asking for metrics not tracked) are common. Errors in generated SQL or misinterpretation of user intent can result in significant business costs, underscoring the importance of robust reliability mechanisms [10]. To mitigate this risk, systems must be able to estimate and communicate confidence in their outputs. Confidence estimation enables selective deployment: the model may choose to present results only when sufficiently confident and defer uncertain queries for human verification. This could reduce the likelihood of misinformation. This paper investigates the novel and practical challenge of estimating confidence for LLM-generated SQL queries in the context of supply chain data extraction. We evaluated three strategies for confidence scoring:

(1) **Translation-based consistency checks**, where SQL is re-expressed in natural language and compared to the original question;
(2) **Embedding-based semantic similarity**, which measures the alignment between the user question and the generated question;
(3) **Self-reported confidence**, where the LLM assigns its own certainty score during or after generation.

Our goal is to assess the effectiveness of these approaches in identifying unreliable queries and to inform more trustworthy LLM-driven data access systems in supply chain environments.

2 Methodology

In this section, we first explain the baseline Text-to-SQL system we used and the evaluation method of the in-distribution text-to-SQL accuracy, and then describe three strategies for estimating confidence in LLM-generated SQL queries, along with relevant supporting literature. We assume an LLM-based text-to-SQL system that takes a user's natural language question as input and produces an SQL query for a supply chain database. The strategies are applied on top of the base system without altering the underlying LLM answer.

2.1 Baseline Text-to-SQL System Simulation

To support accurate SQL generation and evaluation, we first constructed a synthetic question bank by prompting a large language model to generate domain-specific business question templates spanning key supply chain functions such as inventory management, order tracking, demand trends, and supplier performance. Each question is paired with a reference SQL query, accompanying reasoning steps, and verified execution results. All the data is reviewed by domain experts to ensure accuracy. As shown in Table 1,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
KDD '25, Toronto, ON, Canada.
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1454-2/25/08
https://doi.org/10.1145/XXXXXX.XXXXXX

this high-fidelity dataset serves as a reference for SQL generation and trusted ground truth for evaluating text-to-SQL performance.

To improve SQL generation accuracy, our baseline system adopts a retrieval-augmented generation (RAG) framework, which has been shown to significantly enhance factual correctness and response quality in LLM outputs [2, 9]. Given a customer query, the system first performs entity extraction to identify relevant supply chain concepts (e.g., product ID, Order ID). The These sentence-level embedding model is used to extract entities, the most relevant database schema and a set of five semantically similar example questions from the synthetic question bank. The retrieved schema and question-SQL pairs are provided as contextual prompts to the large language model. This targeted retrieval ensures the model is guided by structurally and semantically aligned examples, promoting syntactic correctness, schema consistency, and domain fidelity in the generated SQL.

2.2 Evaluation Method

To evaluate the model's performance and simulate realistic variability, we generated an evaluation dataset by introducing controlled paraphrasing and context variation. Specifically, for each original customer question in the question bank, we used an LLM to generate semantically equivalent rephrasings by altering word choices and entity references [6, 8]. For example, the question *"What's the sales order of Apple_1 in the last week?"* may yield variants like *"Could you tell me the sales order of Olive_3 over the last 7 days?"*. This paraphrasing protocol introduces linguistic and contextual diversity while preserving the original intent, thereby testing the model's generalization to in-distribution examples with increased surface-level complexity [3]. It reflects real-world variability in how business users may phrase similar questions. We define in-distribution accuracy as the proportion of evaluation queries for which the LLM-generated SQL query yields the correct result upon execution—i.e., when the output of the generated query matches the ground truth label Y. This reflects whether the generated SQL is both syntactically correct and semantically aligned with the input question. To assess the utility of different confidence scoring strategies, we report the Area Under the Receiver Operating Characteristic Curve (AUROC). AUROC evaluates the model's ability to discriminate between correct and incorrect SQL predictions based on their associated confidence scores [5]. Intuitively, it measures the probability that a randomly chosen correct prediction is assigned a higher confidence score than a randomly chosen incorrect one. A perfectly calibrated scoring system yields an AUROC of 1.0, indicating flawless ranking. In contrast, an AUROC of 0.5 implies that the confidence scores are no better than random guessing.

The AUROC is defined as:

$$\text{AUROC} = \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{x_p \in \mathcal{P}} \sum_{x_n \in \mathcal{N}} I(\cdot) \left(S(x_p) > S(x_n) \right)$$

$S(x)$ denotes the confidence score assigned to input x, and $\mathcal{P}$ and $\mathcal{N}$ represent the sets of positive (correct) and negative (incorrect) examples. $I(\cdot)$ is the indicator function, returning 1 if the argument is true and 0 otherwise.

2.3 Confidence Scoring Method

LLM-Generated Confidence. Large Language Models (LLMs) can be prompted to articulate or quantify their confidence, either directly within a single response or through a follow-up query. The self-probing approach was introduced first by Kadavath et al. [7]. After generating an answer, the model is asked a binary question such as "Is the previous answer correct?" The predicted probability of a positive response serves as a proxy of confidence. This method demonstrated up to 30% improvement in identifying incorrect answers compared to relying solely on raw token probabilities. Subsequent studies have built on the concept showing that prompting the model to first explain its reasoning and then report a numerical confidence score led to better-calibrated "predictions" (Tian et al. [11]). The experiments indicated that the model's self-reported confidence was more closely aligned with actual correctness than its conditional likelihood, outperforming log-probability-based by approximately 50% in calibration metrics across question-answering tasks. These findings underscore the potential of prompting strategies for eliciting reliable confidence estimates from LLMs.

Translation-based Consistency Check. This approach leverages the intuition that if the SQL truly captures the user's intent, its natural language rendition should semantically match the input question. A self-validation framework called CycleSQL was presented to implement the translation-based consistency check idea Fan et al. [4]. In this approach, after an LLM generates a SQL query, the system executes the query and translates the results back into natural language, essentially producing an answer explanation. It then uses a textual entailment model to compare the natural language explanation with the original question, treating it as a premise-hypothesis pair for natural language inference. If the explanation does not entail the question (indicating a discrepancy in semantics), the SQL is flagged as wrong and the model attempts to refine it. This round-trip verification process significantly boosts accuracy on benchmarks, as the model catches subtle errors where the SQL, though syntactically valid, fails to answer the questions.

Embedding Similarity Approach. Another approach is using embeddings to find if a new input question is semantically similar to some known questions with verified answers. If a closely similar question has appeared before in the question bank with verified answer, one can infer that the LLM's answer to the new question is more likely correct. Conversely, if the question is unlike anything seen before, the model's output should be treated with lower confidence. This technique has been observed in real-world deployments of QA systems to improve consistency. For instance, an LLM-augmented search application might remember that it correctly answered *"How many passengers fit in a Boeing 747?* before, so when a semantically equivalent question *"What is the seating capacity of a 747 jumbo jet?* is asked, the model's answer can be cross-checked against the previous answer for agreement. If they align, the confidence in the answer increases; if not, it may indicate uncertainty. Such semantic caching exploits question-to-question similarity to avoid repeat mistakes and to provide calibrated confidence based on prior successes.

Table 1: Example entry from the synthetic question bank used to evaluate SQL generation

Customer Question	What is the total quantity of `Apple_1` ordered?
Context	`{ 'entities': [{ 'table': 'product', 'id': 'Apple_1', 'description': 'Apple' }, { 'topic': 'inbound_order_line' }] }`
Reasoning	Based on the input question and context: - `Apple_1` refers to a product ID in the `product` table. - This ID is used to filter the `orders` table. - The goal is to sum the `quantity` for this product.
SQL (SQLite)	`SELECT SUM(quantity) FROM orders WHERE product_ID = 'Apple_1'`
Execution Result	`[(40,)]`

3 Experiments

In this section, we present the experimental setup and confidence score accuracy results obtained. The experiments were designed to evaluate the effectiveness of our proposed confidence estimation strategies within a controlled, privacy-preserving environment. These strategies were integrated into our base SQL generation system to assess their applicability and performance.

3.1 Experimental Setup

To ensure the customer data privacy, we demonstrated our method evaluation using a synthetic dataset that we generated. The synthetic dataset consisting of 988 supply chain related questions [12]. There are 731 simpler questions and 257 complex business questions. Simpler questions typically map to SQL queries with a well-defined and consistent structure. For example, the question *"What's the total quantity of item X sold yesterday?"* corresponds to a straightforward aggregation query with minimal variation. In contrast, business questions often involve domain-specific terminology and admit multiple valid SQL formulations. Consider the question *"How many sites have stock-out risk for item Y?"* The term *"stock-out risk"* is a supply chain concept with a fixed SQL definition, typically derived from business logic. Such queries increase semantic ambiguity and structural variability, posing greater challenges for SQL generation systems. All experiments were conducted using Claude Sonnet 3, the most advanced model available at the time of experimentation, accessed through Amazon Bedrock API [1] with default hyperparameter settings. The same model was used for both SQL generation and downstream confidence estimation tasks. Unless otherwise stated, no fine-tuning or task-specific training was performed.

Our evaluation comprises three stages, each exploring a different approach to confidence score generation for LLM-produced SQL queries.

Stage 1: Prompt-Based Self-Reported Confidence. In the first stage, we evaluated the ability of the LLM to self-report its confidence in the generated SQL query. We experimented with multiple prompting strategies to elicit confidence scores in binary (e.g., True or False) and scalar (e.g., score 0-100) formats. Details of the prompt templates used are provided in Table 2. This setup assesses whether the model's self-assessed confidence aligns with actual correctness.

Table 2: Prompt strategies for eliciting self-reported confidence scores from the LLM

Prompt Strategy	Prompt Template
Subtracting from 100	Aim for a wide distribution of scores that accurately reflects the varying levels of confidence you have in the generated SQL.
Money Betting	Aim to place bets that accurately reflect your true confidence in the generated SQL, as if real money were at stake.
SQL Component Weighting	Calculate the overall confidence score by taking the weighted average of the component scores, with weights determined by the importance of each component in the given context.
Straight Instructions	After generating the SQL query, please take a moment to critically analyze your output and generate your own confidence score, between 0 and 100, about the generated SQL.
Binary Classification	Generate the binary judgment of "confident" or "not confident"; explain why you gave that classification within the `<conf_reason>...</conf_reason>` XML tags.

Stage 2: Translation-Based Consistency Scoring. In the second stage, we implemented the translation-based validation framework. For each LLM-generated SQL query, we prompt a second LLM to generate a natural language question that the SQL is likely to answer, given the SQL statement, reasoning trace, and execution result. The inferred question is then compared to the original input question using a third LLM acting as a natural language judge. If the inferred question is semantically equivalent to the original, the SQL is considered high-confidence.

Stage 3: Embedding-Based Similarity Scoring. In the third stage, we adopted an embedding-based approach to estimate confidence through semantic alignment. Specifically, we compute the cosine similarity between the input customer question and the five semantically relevant examples retrieved from the question bank for in-context learning. The average similarity score across these five examples is used as the confidence estimate for the generated SQL

query. This method assumes that if the target question closely resembles high-quality examples, the model is more likely to produce reliable SQL.

3.2 Results

Stage 1. As shown in Table 3, the binary classification prompt strategy achieved the highest AUROC among all evaluated methods, outperforming direct scalar score generation approaches. However, the overall performance across all strategies remains close to the 0.5 threshold, indicating limited discriminative power. While analyzing the mismatch between confidence scores and actual accuracy, we observed that the LLM tends to assign consistently high confidence scores—often above 0.9—to its own generated SQL queries, particularly for simpler questions. This suggests that the model's self-reported confidence scores are poorly calibrated and often overconfident—offering little improvement over random guessing in identifying correct versus incorrect SQL outputs.

Table 3: AUROC performance of different prompt strategies for self-reported confidence estimation in stage 1

Prompt Strategy	AUROC
Subtracting from 100	0.544
Money Betting	0.509
SQL Component Weighting	0.481
Straight Instructions	0.523
Binary Classification	**0.553**

Table 3 shows the classification prompt strategy will provide better AUROC than direct score generation. Overall, the results show that it's over-confident for this method. All of the AUROC are near 0.5 borderline, meaning the confidence score is not much better than a random guess.

Stage 2. Table 4 presents the distribution of confident versus non-confident predictions across accurate and inaccurate SQL generations. The binary confidence classifier achieved an AUROC of 0.524, indicating performance only marginally better than random chance. These results suggest that the strategy remains overconfident and its confidence scores fail to reliably distinguish between correct and incorrect SQL outputs. The translation-based approach demonstrated lower performance on complex questions, particularly due to hallucinations around domain-specific terminology. For instance, the LLM often translates the term "quantity" as "volume" during SQL interpretation. While both terms can imply a numerical measure in general language, they carry distinct meanings in the supply chain domain: quantity typically refers to the number of units ordered, whereas volume refers to the physical size or space occupied by the items. This kind of domain-specific ambiguity in natural language is difficult for the model to detect and disambiguate, often resulting in overconfident yet incorrect SQL generation.

Stage 3. The similarity-based confidence score achieved an AUROC of 0.57 across both simple questions and complex questions, the highest among all methods evaluated. Figure 1 illustrates the relationship between SQL execution accuracy, return rate (i.e., the proportion of queries with confidence above a given threshold), and average confidence score for simple questions. Figure 2 presents the same analysis for complex business questions.

Table 4: Confusion matrix of binary confidence classification vs. actual SQL correctness in stage 2

Confidence Accuracy	Confident	Non-Confident
Accurate	754	12
Non-Accurate	208	14

Both figures reveal a clear pattern: when the confidence score exceeds a specific threshold, empirically observed around 0.85, the majority of generated SQL queries attain significantly higher execution accuracy. This indicates that the similarity score serves as a reliable proxy for SQL correctness and can effectively be used to filter out low-confidence outputs, thereby improving the overall trustworthiness of the system.

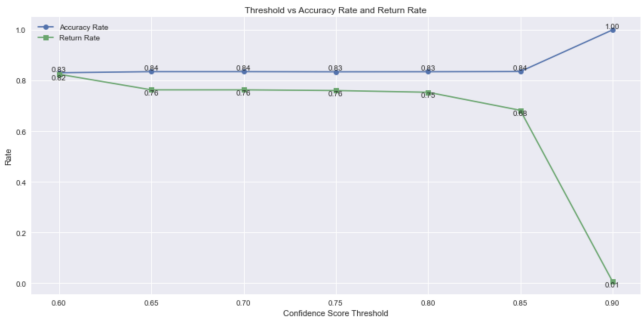

Figure 1: Simple Questions: Accuracy and Returning Rate by Confidence Level

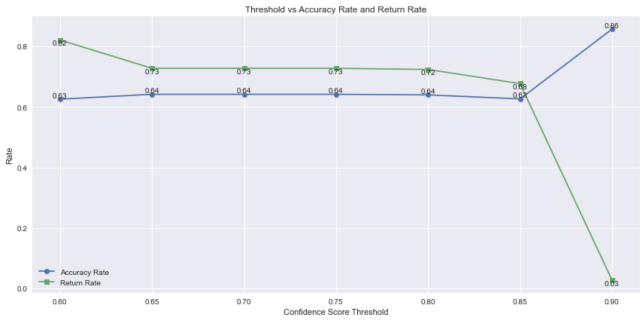

Figure 2: Complex Questions: Accuracy and Returning Rate by Confidence Level

4 Dicussion

In this study, we evaluated three distinct approaches for estimating confidence levels in LLM-generated SQL queries: (1) self-reported scores via direct prompting, (2) translation-based consistency checks, and (3) embedding-based similarity scoring.

Our experimental results reveal a consistent trend: large language models tend to be overconfident in the SQL generation to extract supply chain data, regardless of the prompting strategy employed. Among the self-assessment methods, the binary classification prompt outperformed scalar score generation, though still with limited discriminative ability. In contrast, the embedding-based similarity score demonstrated the stronger correlation with SQL correctness, achieving the higher AUROC and showing clear utility in separating high-quality from low-quality generations.

As a practical insight for real-world deployment, we find that the similarity score offers a promising mechanism for thresholding LLM outputs—particularly in complex supply chain data extraction scenarios, where schema diversity and business logic can increase error risk. By filtering out low-confidence queries based on semantic alignment, systems can significantly improve overall reliability and user trust.

Future work will explore the generalizability of the similarity-based confidence strategy across diverse supply chain datasets, schemas, and query intents. In addition, integrating confidence-driven post-filtering into end-to-end decision support systems may offer further gains in production robustness.

5 Limitation and Future Work

One limitation of this study is that all experiments were conducted on a synthetic dataset due to customer data privacy concerns. While the synthetic data allowed us to prototype and evaluate the confidence scoring approach in a controlled environment, it may not fully capture the complexities of real-world supply chain data. In future work, we aim to extend our evaluation to public supply chain datasets, such as the M5 competition dataset, to validate the generalizability and robustness of our approach under more realistic conditions. Additionally, Large Language Models are evolving rapidly. Although this study used Claude Sonnet 3, the latest model available to us at the time, it is important to evaluate the proposed confidence scoring strategy across a broader set of LLMs. Future work will explore the performance of the method on newer and more diverse models, including GPT-4, DeepSeek, and Claude Sonnet 3.5 or 3.7. As different models, particularly those optimized for reasoning, may demonstrate varying levels of confidence reliability, such comparison is essential to assess the method's transferability and performance consistency. Lastly, our current confidence scoring framework is built on a baseline text-to-SQL system that leverages a retrieval-augmented generation pipeline. As more domain-specific supply chain databases are developed, incorporating fine-tuning strategies on top of this foundation may further improve both SQL generation accuracy and the reliability of the associated confidence scores. Future exploration into fine-tuning will allow for deeper model alignment with domain-specific language and schema patterns, thereby enhancing practical utility in enterprise settings.

References

[1] Amazon Web Services. 2023. Amazon Bedrock — Build and scale generative AI applications with foundation models. https://aws.amazon.com/bedrock/. Accessed: 2025-05-14.

[2] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Kate Millican, George van den Driessche, Bogdan Damoc, Aidan Clark, Andrew Brock, et al. 2022. Improving language models by retrieving from trillions of tokens. *Nature* 610, 7930 (2022), 593–603.

[3] Li Dong and Mirella Lapata. 2020. Data augmentation for multilingual semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7335–7346.

[4] Yuankai Fan, Tonghui Ren, Can Huang, Zhenying He, and X. Sean Wang. 2024. Grounding Natural Language to SQL Translation with Data-Based Self-Explanations. *arXiv preprint arXiv:2411.02948* (2024). under review.

[5] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.

[6] Ziniu Hu, Xinyan Li, Jimmy Lin, Mu Li, and Li Deng. 2022. Paraphrasing for all NLP tasks via semi-supervised prompt learning. In *Findings of ACL*.

[7] Chethan Kadavath, Shashank Bansal, Jacob Schaeffer, Andrew Lovitt, Deep Ganguli, David Krueger, Christopher Manning, and Geoffrey Irving. 2022. Language Models "Know" When They're Guessing. *arXiv preprint arXiv:2207.05221* (2022).

[8] Ritesh Kumar, Arvind Singh, and Vasudeva Varma. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kulkarni, Devang Chaudhary, Tim Rocktäschel, Angela Fan, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 9459–9474.

[10] Beibin Li, Konstantina Mellou, Bo Zhang, Jeevan Pathuri, and Ishai Menache. 2023. Large Language Models for Supply Chain Optimization. *arXiv preprint arXiv:2307.03875* (2023).

[11] Yujia Tian, Suzanna Sia Chia, Ledell Wu, Ganesh Ramakrishnan, and Zachary C Lipton. 2023. Justifications Improve Model Calibration in Natural Language Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

[12] Rakshit Trivedi, Kai-Wei Chang, Niket Awasthi, and Mo Yu. 2023. Can Large Language Models Automate Supply Chain Analytics? An Evaluation on Text-to-SQL Tasks. *arXiv preprint arXiv:2308.12980* (2023).