B o o l e a n w o r k   T e c h n o l o g i e s   L L P
sales@booleanwork.co

Natural language query to SQL

Introduction

This document describes an advanced design for converting natural language text into SQL queries using Large Language Model (LLM) agents. In addition to SQL validation and Chain of Thought (CoT) reasoning, this updated design introduces a ***Human-in-the-Loop (HITL)*** component to confirm ambiguous column names or resolve other query uncertainties. This approach further increases the robustness and reliability of the system by involving a human to validate critical aspects when needed, ensuring accurate query generation.

System Overview

Key Components

- **LLM Agent:** A large language model trained to interpret natural language questions, apply structured reasoning via CoT, and convert them into SQL queries.
- **Chain of Thought (CoT) Module:** Guides the LLM through a logical reasoning process, helping it break down complex queries step-by-step for accurate SQL generation.
- **SQL Validator:** A module that checks the generated SQL query's syntax, schema compliance, and overall validity.
- **Human-in-the-Loop (HITL) Interface:** An interface that requests user confirmation when the LLM is uncertain about column names, tables, or ambiguous terms.
- **Database:** SQL-compliant database like PostgreSQL, MySQL, etc., where data is queried.
- **Agent Manager:** Handles the workflow, interactions with the LLM, SQL Validator, and Human-in-the-Loop, as well as error handling.

Workflow

User Input

The user enters a natural language question, such as *"Show the top customers by purchase amount and region."*

Chain of Thought Reasoning (CoT)

The Chain of Thought (CoT) module allows the LLM to break down complex queries into logical steps. This process makes the LLM's reasoning more transparent and guides its response to complex or multi-step queries.

For example:
  - Step 1: Identify metrics: **top customers**, **purchase amount**, and **region**.
  - Step 2: Determine necessary aggregations and sorting.

SQL Query Generation with Ambiguity Detection
Using CoT guidance, the LLM generates an SQL query based on its understanding of the schema. If it detects ambiguity, such as unfamiliar or similar column names, it flags these terms for verification.

Example SQL:

```
SELECT customer_name, region, SUM(purchase_amount) AS total_purchases
FROM orders
GROUP BY customer_name, region
ORDER BY total_purchases DESC
LIMIT 10;
```

SQL Validator

The SQL Validator module checks for:

- **Syntax Validation:** Ensures the SQL query adheres to SQL syntax.
- **Schema Validation:** Confirms that tables and columns referenced in the query exist in the database schema.
- **Execution Testing:** Performs a dry run to validate the query without modifying the database.

If the SQL Validator detects a schema discrepancy or ambiguity, it triggers the ***Human-in-the-Loop (HITL) module*** to confirm.

Human-in-the-Loop (HITL) Interaction

When an ambiguity in column names or tables is detected, the HITL module engages a human user to clarify.

- **HITL Trigger:** The LLM or SQL Validator identifies ambiguous terms or mismatched columns and flags the query for user review.
- **HITL Interface Notification:** The user is prompted with a message, such as:

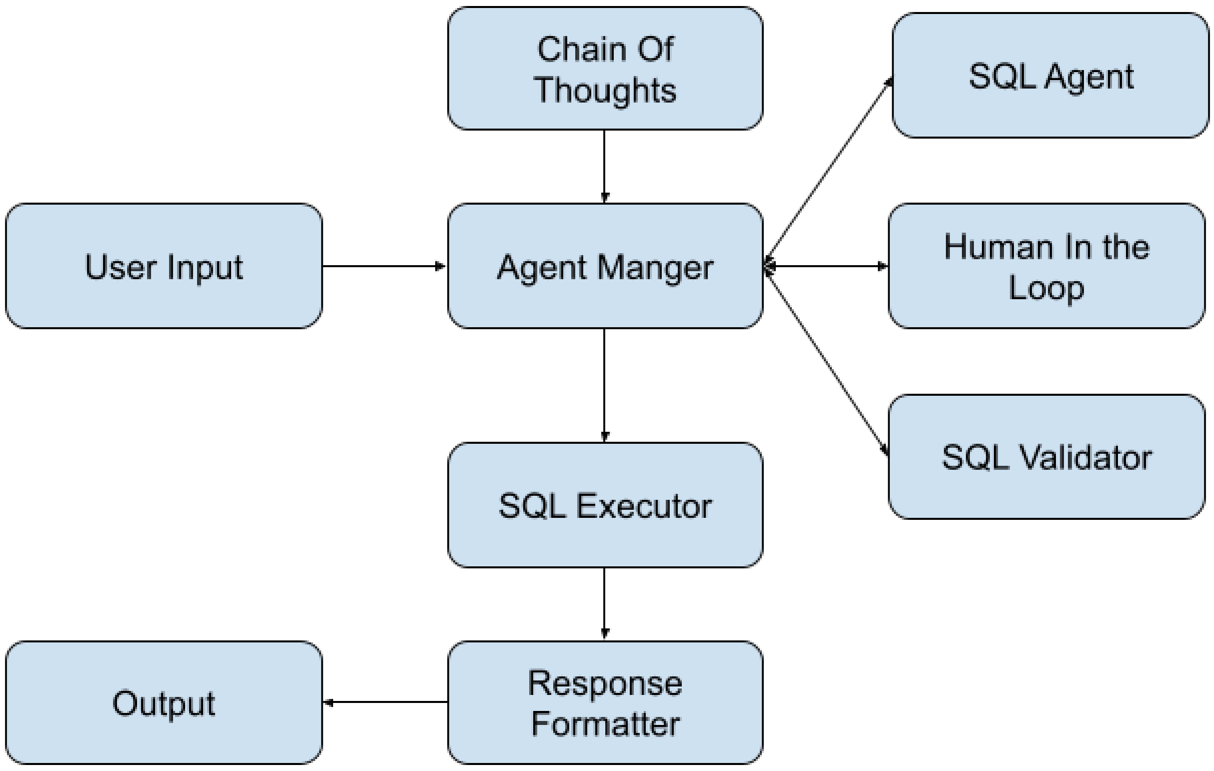
- - *"Could you confirm if 'purchase_amount' refers to 'total_amount' or 'net_amount' in the 'orders' table?"*
- **User Selection:** The user selects or confirms the correct column names from a suggested list or manually inputs them.
- **Query Regeneration:** The LLM refines the SQL query based on the user's feedback.

This step ensures that critical ambiguities are resolved before executing the query, minimizing the risk of incorrect outputs.

Query Executor
Once validated, the SQL query is executed on the database. If no issues are identifed, the database returns the results based on the user's request.

Response Formatting
The query results are formatted for a user-friendly presentation, such as a table or chart, and displayed in the user interface.

System Architecture



Example Walkthrough

User Query: ***List the top 10 products by sales in Q1 2024, including revenue and profit.***

1. **Chain of Thought:**
   - Step 1: Identify key metrics: ***top products, sales, revenue, profit.***
   - Step 2: Define time range for Q1 2024: *2024-01-01 to 2024-03-31*.
   - Step 3: Prepare query to aggregate ***revenue*** and ***profit*** for each product.

2. **SQL Query Generation:**

```
SELECT product_name, SUM(revenue) AS total_revenue, SUM(profit) AS total_profit
FROM sales_data
WHERE sale_date BETWEEN '2024-01-01' AND '2024-03-31'
GROUP BY product_name
ORDER BY total_revenue DESC
LIMIT 10;
```

3. **SQL Validator and HITL Trigger:**
   - **Validation:** The SQL Validator detects a potential ambiguity—whether **revenue** and ***profit*** refer to columns ***gross_revenue*** and ***net_profit***.
   - **HITL Request:** The system prompts the user with: "*Could you confirm if 'revenue' and 'profit' should refer to 'gross_revenue' and 'net_profit' in the 'sales_data' table?*"
   - **User Response:** The user confirms, and the SQL query is adjusted based on feedback.

4. **Execution:** The validated query is executed, and results are returned.

Security and Privacy Considerations

- **SQL Injection Prevention:** Input sanitization and parameterization prevent SQL injection attacks.
- **Data Privacy:** Sensitive data is masked according to role permissions. HITL interactions are carefully logged to prevent unintentional exposure.
- **HITL Audit Logs:** Records of user confirmations are stored for troubleshooting and query auditing.

Previous Projects:

- **SQL Generator for Financial Databases:** This project automates the creation of SQL queries tailored for financial databases by leveraging table schemas and domain-specific knowledge. It ensures accurate, schema-compliant query generation for tasks like transaction analysis, account reporting, and auditing, catering to both technical and non-technical users.

- **User Events DB Interactor:** This system focuses on retrieving and analyzing user event data to gain insights into user engagement, sentiment, and journey. It provides advanced filtering and querying capabilities to extract meaningful patterns from event data, supporting applications like behavioural analytics and customer journey mapping.