Agentic Diagnostic Backtracking for NL2SQL on Federated Database Systems

Joel Hudgens, B.S. Computer Science
Mentor: Doctor Jia Zou, Assistant Professor
School of Computing and Augmented Intelligence

Introduction

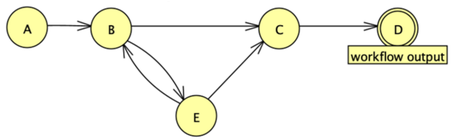
Natural Language to Structured Query Language (NL2SQL) systems have seen significant imporvements with the rapid advancement of Large Language Models (LLMs). Given a database schema and a natural language query, modern LLMs are capable of consistently outputting highly accurate SQL queries. However, this capability has largely been demonstrated on single, well-defined databases. Real-world enterprise systems are rarely this simple: operational data is routinely partitioned across multiple specialized systems that share no common schema or identifier format. Querying across these federated systems requires decomposing a single question into coordinated sub-queries, transforming data across incompatible representations, and synthesizing results, a challenge that current NL2SQL benchmarks do not address.

Database Setup

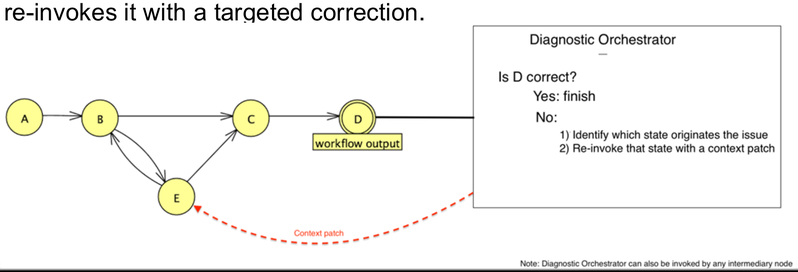
The experimental database was constructed by extracting patient records from MedAgentBench[1], an existing medical benchmark, and partitioning the data into separate patient and insurance claims tables with applied schema transformations to simulate real-world federated system incompatibilities.

Approach

Complex multi-step generation tasks are commonly structured as directed pipelines where discrete LLM stages are placed on "rails" to constrain the agent's solution space. LangGraph[2], a graph-based orchestration framework, formalizes this pattern by allowing developers to define nodes (LLM stages) and edges (control flow) that route execution toward a final output.



Initial querying trials exposed an inherent limitation of this framework: recovery is strictly local. When a later stage fails, the pipeline can retry that stage or step back one node, but it has no mechanism to identify which earlier stage introduced the error. For example, if stage D yields an incorrect result, the true cause may lie in a flawed decision made at stage B. Rather than retrying D or blindly restarting, the diagnostic orchestrator reads the full execution trace, identifies the originating stage, and re-invokes it with a targeted correction.

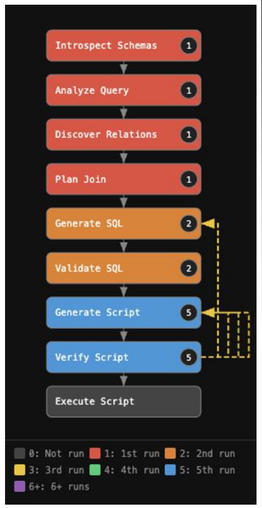


Benchmarking

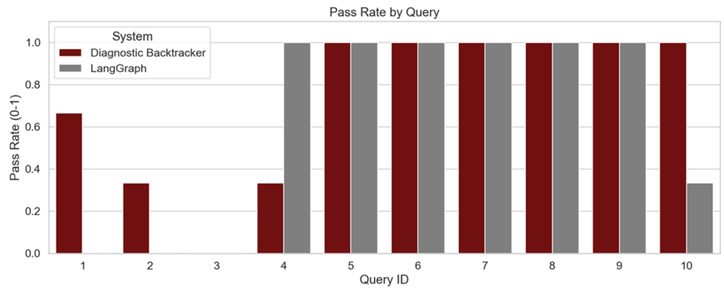
Ten natural language queries spanning both the patient and claims databases were constructed, each paired with a ground truth result set.

To evaluate the diagnostic orchestrator, we benchmark it against a LangGraph baseline across all ten queries, running each three times to account for LLM nondeterminism. Pass rate and token consumption are tracked as primary metrics.

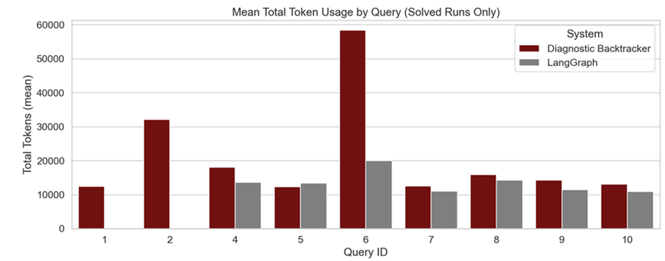
A visualization interface was also developed to inspect individual pipeline executions. Shown at right, each node represents a defined stage of the NL2SQL pipeline defined for this particular goal, with yellow edges indicating diagnostic orchestrator invocations.



Query Accuracy Results



Token Consumption Results



Conclusion

The Diagnostic Orchestrator achieved a pass rate of 22/30 compared to 19/30 for LangGraph, showing a promising improvement. However, repeated diagnostic invocations can drive token consumption two to three times higher than LangGraph. These results suggest diagnostic backtracking is a viable direction, though “human in the loop” remains more practical for complex multi-step reasoning tasks.

Future Direction

Next steps include expanding the query set, improving benchmark speed via multithreading, and testing whether earlier versus later diagnostic interventions affect accuracy and token cost. Additional experiments will explore whether node-specific criteria or more targeted diagnostic prompts, informed by domain-specific correctness criteria, can sharpen the diagnoser's reasoning and improve correction precision. Longer term, the framework will be tested on problems beyond NL2SQL

References

[1] Jiang, Y., Black, K. C., Geng, G., Park, D., Zou, J., Ng, A. Y., & Chen, J. H. (2025). MedAgentBench: A virtual EHR environment to benchmark medical LLM agents. *NEJM AI*, *2*(9). https://doi.org/10.1056/AIdbp2500144
[2] LangChain Inc. "LangGraph Overview." *Docs by LangChain*, 2025, docs.langchain.com/oss/python/langgraph. Accessed 4 Apr. 2026.

FURI
This work would not have been possible without the mentorship and support of Professor Jia Zou.
Thank you!

Ira A. Fulton Schools of Engineering
Arizona State University