A Conversational Agent Framework for Multimodal Knowledge Retrieval: A Case Study in FHWA InfoHighway Web Portal Queries

Sai Surya Gadiraju, Zijie He, Duoduo Liao
School of Computing
George Mason University
sgadira3@gmu.edu, zhe20@gmu.edu, dliao2@gmu.edu

Abstract

The rapid proliferation of heterogeneous data in government and industry presents increasing challenges for users seeking to retrieve actionable insights across both structured and unstructured sources. To address this, this paper presents *InfoTech Assistant*, a novel multimodal conversational framework that enables natural language interaction with both semantic document retrieval and structured database querying. The system integrates Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) and schema-aware Text-to-SQL capabilities, enabling dual-mode processing of user input for unstructured explanations and relational analytics. The architecture features a modular, locally deployed backend built with Flask and optimized for Graphics Processor Unit (GPU) acceleration, supporting low latency, privacy preserving inference. User queries are dynamically routed through an intent-aware processing pipeline, leveraging sentence embeddings, schema metadata, and prompt engineering strategies. A pilot deployment using infrastructure datasets from the Federal Highway Administration (FHWA) InfoHighway portal demonstrates the system's effectiveness in real-world domain-specific retrieval. The assistant ingests FHWA technology documents and National Bridge Inventory (NBI) text records, tables, and images organized in a hybrid schema supporting both semantic and SQL-driven interaction. Evaluation results show 95% accuracy in RAG-based semantic tasks and 88.6% success in translating natural language into executable SQL queries. These findings underscore the potential of hybrid LLM-based agents for scalable, secure knowledge access in critical public-sector and industrial applications.

1 Introduction

Public infrastructure management increasingly depends on multimodal data sources, including structured databases and unstructured documentation, to support tasks such as maintenance planning, inspection analysis, and policy development. However, deriving actionable insights from these sources often requires technical expertise in Structured Query Language (SQL), relational schema design, and the ability to interpret domain-specific documents. These challenges pose significant barriers to non-technical stakeholders such as policymakers and professionals outside of database or data science domains, including engineers from civil or infrastructure disciplines. This highlights the growing need for unified systems that simplify access to both structured and unstructured data through natural language interaction.

Recent advancements in Natural Language Processing (NLP), particularly with Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) (Vasiliev, 2020; Mohammadjafari et al., 2024), offer promising solutions to bridge this gap. These approaches enable intuitive, conversational interfaces capable of synthesizing information from diverse sources. Yet, most existing systems are tailored for open-domain use cases and primarily focus on either unstructured document retrieval or structured SQL generation, rarely supporting multiple modalities within a unified framework.

To address this limitation, this paper introduces *InfoTech Assistant*, a multimodal knowledge retrieval framework that integrates LLMs within a dual processing architecture. The system supports both document grounded semantic retrieval via RAG (Gadiraju et al., 2024) and structured querying through a schema-aware Text-to-SQL module. This design facilitates seamless interaction with infrastructure data text, tables, and image references through a single natural language interface.

The system is deployed on a locally hosted, GPU-accelerated backend using a modular Flask architecture, enabling low latency and privacy preserving inference. A case study on the Federal Highway Administration's (FHWA) InfoBridge

portal (Federal Highway Administration, 2024a) demonstrates its effectiveness, with high accuracy observed across semantic and SQL driven tasks. Output formats include HTML tables and reference-linked responses, enhancing both usability and domain adaptability.

The main contributions of this paper are as follows:

1. **Unified Conversational Interface:** A single interface supports both unstructured and structured data access, integrating text, tables, and images to streamline user interaction across diverse query types.

2. **Domain-Specific Dual-Mode Retrieval:** A hybrid framework combines RAG with schema aware Text-to-SQL processing to deliver accurate, context-aware responses for infrastructure-related queries.

3. **Locally Deployed, High-Performance System:** The architecture supports GPU accelerated, on-premise inference for real-time, privacy-preserving access to heterogeneous data sources.

4. **Real-World Evaluation:** The system is validated on FHWA bridge data, demonstrating strong performance in both semantic retrieval and structured query tasks.

This work introduces a practical framework for multimodal knowledge retrieval and offers a template for deploying conversational agents in domain-specific public-sector and industrial applications.

2 Related Work

Structured data querying and unstructured knowledge retrieval remain key challenges in domain-specific Question Answering (QA). Prior research in this space generally falls into two categories: Text-to-SQL techniques for translating natural language into executable database queries (Mohammadjafari et al., 2024; Pandey et al.), and LLM-based systems with RAG for grounding responses in external documents (Oreški and Vlahek, 2024; Jeong, 2023). This section reviews representative work across both directions.

2.1 LLM-Based Text-to-SQL Systems

Text-to-SQL models have evolved from early rule-based approaches to neural methods using schema aware prompting and pretrained language models (Mohammadjafari et al., 2024). More recent work introduces retrieval-augmented enhancements to improve alignment between user intent and relational schema (Pandey et al.), while graph-based techniques (Ma et al., 2025) enhance schema linking by modeling table relationships.

However, many existing solutions rely on cloud-based inference or complex multi-stage pipelines, which introduce latency and privacy concerns. To address these limitations, recent approaches have explored local deployment and improved interpretability (Dou et al., 2023; Gadiraju et al., 2024). The approach described in this paper adopts a lightweight, schema-aware Text-to-SQL module that uses prefix-triggered routing and local LLM inference (Jha et al., 2025), optimized for structured infrastructure datasets.

2.2 LLMs with RAG for Domain-Specific QA

RAG frameworks extend LLM capabilities by incorporating external document context during inference, supporting more accurate and context-aware responses (Oreški and Vlahek, 2024; Jeong, 2023). While RAG has shown strong results in open-domain and educational applications (Cabezas et al., 2024), its use in infrastructure domains is still limited especially in systems requiring structured data integration or multimodal output support (Wang et al., 2025)(Zhao et al., 2024).

The proposed system extends prior work in infrastructure-specific RAG applications (Gadiraju et al., 2024) by embedding a schema-aware Text-to-SQL engine within the RAG pipeline. This integration enables seamless access to both unstructured explanations and structured analytics within the same conversational interface.

3 System Architecture Framework

The InfoTech Assistant is a modular, extensible conversational system that integrates RAG and Text-to-SQL capabilities. Built on locally deployed LLMs and GPU acceleration, the system supports high performance, multimodal knowledge retrieval as shown in the Figure. 1 illustrates the overall architecture.

3.1 User Interface and Request Initialization

User interactions begin at the frontend, implemented in HTML/JavaScript, where natural language queries are entered via an interactive chatbot

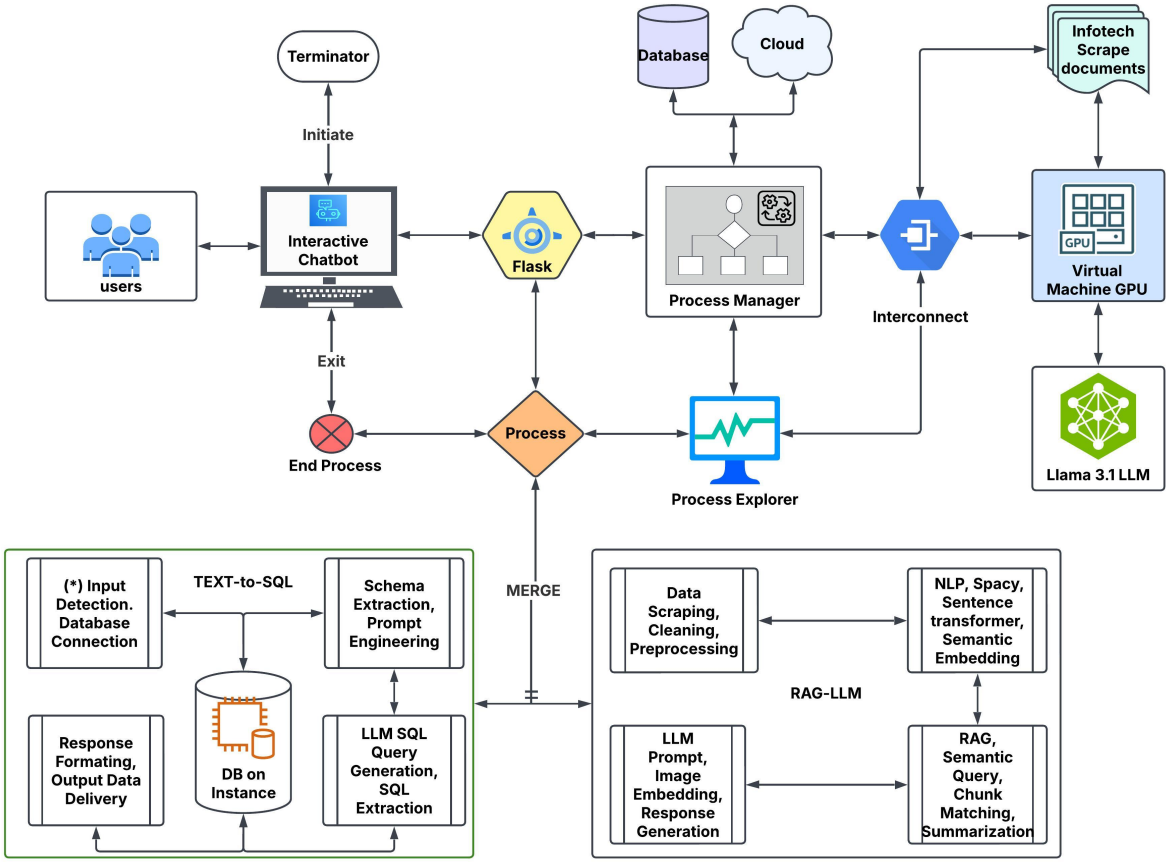


Figure 1: System Architecture of the InfoTech Assistant: A Multimodal Knowledge Retrieval Framework.

interface. These are transmitted to the backend through HTTP endpoints managed by a Flask web server (Relan, 2019). The received query is forwarded to the core processing pipeline for classification and execution.

3.2 Flask Backend and Process Management

The Flask backend functions (Relan, 2019) as the orchestration engine, handling request validation and routing. A *Process Manager* component determines the execution path based on input type. Inputs prefixed with an asterisk (*) are routed to the Text-to-SQL module (Jha et al., 2025), while others are treated as open-domain queries handled by the RAG engine.

3.3 Semantic Retrieval via RAG-LLM

For general information queries, the system activates the RAG pipeline, comprising:

- **Document Preprocessing:** Textual documents are cleaned, segmented, and embedded using the `all-mpnet-base-v2` model from SentenceTransformers (Siino, 2024).
- **Keyword Extraction:** Named entities and noun phrases are identified using spaCy (Srinivasa-Desikan, 2018).
- **Semantic Matching:** Cosine similarity is computed between user query embeddings and document vectors to identify top-matching segments (Alfianto et al., 2023).
- **Prompt Construction and Inference:** The retrieved content and user query are merged into a prompt and passed to a locally hosted LLaMA 3.1 8B Instruct model (Vavekanand and Sam, 2024; Meta AI, 2024) via `llama-cpp-python` for response generation.

This enables multimodal, document-grounded responses, often enriched with source citations and image references.

3.4 Structured Query Handling via Text-to-SQL

For structured inputs (prefixed with *), the system activates the Text-to-SQL pipeline:

- **Schema Extraction:** A local SQLite instance (Dar and Iqra, 2016) hosts structured data. Schema metadata is extracted using PRAGMA commands.
- **Prompt Engineering:** The schema, example templates, and constraints (e.g., SELECT-only rules) are embedded into the system prompt (Chen et al., 2024).
- **LLM Query Generation:** The prompt is processed by the LLaMA model as shown in Appendix B, which returns a syntactically valid SQL query aligned with the user's intent as shown in Appendix A.
- **Execution and Delivery:** The query is executed on the local database, and results are returned as HTML-formatted tables.

3.5 Inference Runtime and Hardware Integration

All inference tasks semantic generation and SQL translation are executed locally on a GPU-enabled virtual machine cluster. This setup provides low latency responses while preserving data privacy through on-premise computation (Gupta et al., 2009).

3.6 Error Handling and Monitoring

A dedicated *Process Explorer* continuously monitors system operations. It captures runtime errors, such as invalid SQL or embedding failures, and triggers fallback responses via the Flask interface to maintain conversation continuity (Relan, 2019).

3.7 Session Termination and Re-initialization

Session lifecycle is managed through a termination component (Hunt et al., 2003) that gracefully resets application state upon user request. This ensures memory cleanup and readiness for subsequent interactions.

3.8 Interconnected Workflow

All modules communicate through a unified message-passing and orchestration layer. The Process Manager directs transitions between RAG and SQL modules (Bartczak, 2024), while the Interconnect layer manages asynchronous task execution. This tightly coupled architecture enables real-time multimodal retrieval with extensibility to additional data domains.

4 Case Study

This section evaluates the InfoTech Assistant in a real-world infrastructure context. The system, powered by Meta LLaMA 3.1 8B and hosted on a GPU-enabled VM, was tested using FHWA InfoBridge content (Federal Highway Administration, 2024b) and a structured SQLite database built from Fairfax County bridge datasets (Federal Highway Administration, 2024a). Evaluation spans two modes: RAG for unstructured semantic queries and schema-guided Text-to-SQL for structured analytics. This dual setup supports multimodal, knowledge grounded QA over technical infrastructure data.

4.1 Experimental Setup

The InfoTech Assistant is deployed as a full-stack application consisting of a Flask-based backend (Haeruddin et al., 2025) and an interactive HTML/JavaScript frontend. The system is hosted on a GPU-enabled Virtual Machine (VM) cluster at George Mason University's ORC cluster (Office of Research Computing, 2025), equipped with CUDA 12.4 and the `llama-cpp-python` library for accelerated inference, as detailed in The model was hosted and executed on a high-performance GPU system and key runtime configuration details are listed in Table 1.

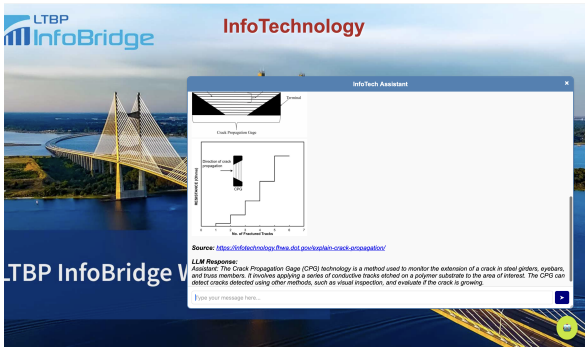
All necessary Python dependencies, including NLP and LLM toolkits such as spaCy, Sentence-Transformers, and Transformers, are managed via a `requirements.txt` configuration (Srinivasa-Desikan, 2018). The Meta LLaMA 3.1 8B model in GGUF format is automatically retrieved from Hugging Face using the `huggingface_hub` API (Meta AI, 2024) and loaded for local inference. All backend modules including RAG, Text-to-SQL, and semantic search are integrated within a single runtime environment to support low-latency, context-aware interactions. Evaluation results are summarized in Table 2 and Table 3.

4.2 Interaction Modes and System Responses

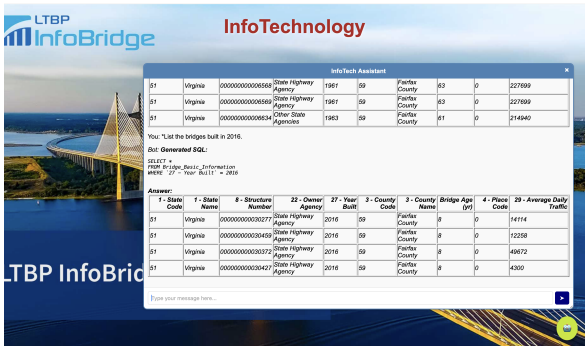
The InfoTech Assistant supports two core interaction modes: RAG and Text-to-SQL (Dou et al., 2023). These modes support both explanatory and analytical information needs, catering to a broad range of users, including domain experts, researchers, and non-technical personnel.

Figure 2 illustrates responses from each mode. In RAG mode, natural language queries are se-

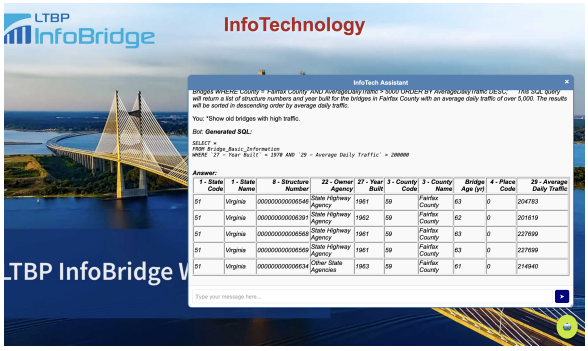
(a) User Query with Textual Response



(b) LLM-Summarized Response with Reference Link



(c) SQL Table Response for User Query



(d) Text-to-SQL Generated Tabular Output

Figure 2: Multimodal responses from the InfoTech Assistant, demonstrating RAG-based and Text-to-SQL responses with visual, textual, and tabular outputs.

mantically matched to a document corpus using sentence embeddings, and the top-ranked chunks are summarized via a local LLM. For example, Figure 2a shows the assistant explaining Crack Propagation Gages (Liaw et al., 1983), while Figure 2b demonstrates retrieval of an associated image and source link.

For structured data access, users prepend queries with an asterisk (*), activating the Text-to-SQL module. The assistant then generates and executes SQL over a structured SQLite database. As shown in Figure 2c, the assistant returns tabular results for a traffic query, and Figure 2d highlights filtered results based on bridge age and traffic conditions. This dual-mode architecture offers seamless transitions between semantic explanation and structured data querying. [1]

4.3 Evaluation Metrics

To evaluate system performance, distinct metrics were used for RAG and Text-to-SQL modules.

[1]Banner source: https://infobridge.fhwa.dot.gov/Content/images/LTBP%20InfoBridge%20Social%20Inspection%20Banner.jpg

Semantic Retrieval Evaluation: The semantic retrieval component is evaluated using two metrics: Cosine Similarity between Sentence Transformer embeddings of the query and the retrieved chunk (Gadiraju et al., 2024), and manual accuracy ratings. Table 2 reports results for representative domain-specific queries. The average similarity score is 0.95, and the mean accuracy is 95%, reflecting strong alignment between user intent and the retrieved content.

Evaluation Protocol for Text-to-SQL: The effectiveness of SQL generation is assessed using a structured SQLite database (Dar and Iqra, 2016) derived from the Fairfax County bridge dataset (Federal Highway Administration, 2024a). A three-tiered difficulty taxonomy is adopted to reflect varying query complexities, as outlined in Equation 1:

- **Level 1:** Simple column retrievals without conditions or aggregation.
- **Level 2:** Queries involving logical filters, numeric thresholds, or aggregate functions.
- **Level 3:** Complex queries requiring multi-table joins and interpretation of implicit semantics (e.g., “old bridges”).

Table 1: System Configuration Summary

Component	Details
LLM	Meta LLaMA 3.1-8B-Instruct
Inference Engine	`llama-cpp-python`
Execution Platform	George Mason University's ORC GPU cluster
Configuration Parameters	`n_ctx=4096` (context window size), `n_gpu_layers=-1` (all layers offloaded to GPU), `batch_size=1024` (tokens processed in parallel), `flash_attn=True` (enables FlashAttention for faster computation)
Hardware Specifications	NVIDIA RTX 4090 GPU with 24 GB GDDR6X memory, 16,384 CUDA cores, 384-bit memory interface, 1,008 GB/s memory bandwidth

Table 2: Sample Questions: Similarity and Accuracy Results for RAG-Based Semantic QA

Question	Similarity Score	Accuracy
What is Electrical Resistivity?	0.94	95%
What are the benefits of Hammer Sounding?	0.92	94%
Can you explain the Crack Propagation Gage (CPG)?	0.97	99%
How to do Screw Withdrawal Testing?	0.98	98%
Explain Transverse Vibration of Structural Systems.	0.96	96%
Why is coring considered best for visual inspection?	0.94	92%
Can you explain Stress Wave Timing?	0.94	96%

A total of 30 natural language queries (10 per level) are executed five times each to account for variability in LLM-generated outputs. Each result is manually validated for both syntactic correctness and semantic fidelity. Errors are categorized as either (i) non-executable SQL statements or (ii) logically incorrect results that misinterpret the user's intent.

Table 3 summarizes the performance across all levels. The assistant achieves an overall accuracy of 88%, with highest precision on Level 1 queries (92%), followed by Level 2 (88%) and Level 3 (86%). Accuracy is computed using the formula:

$$\text{Accuracy} = \frac{\text{Number of correct SQL queries}}{\text{Total number of queries executed}} \times 100\% \quad (1)$$

5 Conclusions

This paper presents *InfoTech Assistant*, a unified multimodal conversational framework designed to improve access to both structured and unstructured data through natural language interaction. Unlike general-purpose dialog systems, the framework integrates RAG with a schema-aware Text-to-SQL module (Dou et al., 2023), enabling efficient retrieval from technical documents, relational databases, and visual data sources.

Developed using Python and deployed locally with GPU acceleration, the system supports low-latency, privacy-preserving inference and provides an accessible web-based interface. A case study on infrastructure datasets from the FHWA demonstrates the framework's effectiveness in domain-specific knowledge retrieval. The system is intended as a reference architecture for real-world knowledge management tasks in public-sector and industrial settings. Its multimodal capabilities support both technical and non-technical user needs, offering adaptability across a broad range of applications. Ongoing community feedback and open-ended extensibility are encouraged to guide future enhancements.

Limitations

The current implementation embeds the full database schema into the prompt, which may limit scalability for larger or more complex databases due to the context window constraints of the language model. Additionally, the system does not support interactive learning or dynamic adaptation based on user feedback, limiting its capacity for autonomous improvement. Full schema context can also result in increased response latency under high-load scenarios.

Moreover, the current evaluation is limited to a single domain-specific dataset and does not include baseline or ablation comparisons. While the

Table 3: Sample Questions: Accuracy Results for Text-to-SQL Module

Sample Question	Difficulty Level	Accuracy
Show the year built for all bridges.	Level 1	92%
Retrieve the average daily traffic for all bridges.	Level 1	92%
Show bridges older than 80 years.	Level 2	88%
Retrieve all columns for bridges built in 2016.	Level 2	88%
List bridges with average daily traffic over 100,000 and built after 2000.	Level 2	88%
List congested bridges in Fairfax.	Level 3	86%
Show old bridges with high traffic.	Level 3	86%
Retrieve structurally deficient bridges built before 1970.	Level 3	86%

results demonstrate consistent performance across varying query difficulty levels, broader validation through comparative studies and statistical analyses remains an area for future work. Future enhancements may also include schema summarization, adaptive query routing, and optimization strategies to address these challenges.

Ethical Considerations

The InfoTech Assistant operates entirely on a secure, local GPU cluster at George Mason University, ensuring no user data is transmitted externally. It uses publicly available datasets (e.g., FHWA InfoBridge (Federal Highway Administration, 2024a)) and maintains only short-lived session data for contextual interactions.

All responses are grounded through document retrieval or structured database queries to minimize hallucinations. While efforts are made to ensure factual accuracy, occasional errors may occur, especially for ambiguous queries. Human oversight is recommended for safety-critical use cases.

Acknowledgments

The authors thank the Federal Highway Administration (FHWA) (Federal Highway Administration, 2024b) for providing public datasets that were instrumental in testing and evaluation. This research utilized the Hopper GPU cluster (Office of Research Computing, 2025) and resources of the Department of Information Sciences and Technology provided by the George Mason University Online Research Computing Environment.

References

Meizan Arthur Alfianto, Yudi Priyadi, and Kusuma Ayu Laksitowening. 2023. Semantic textual similarity in requirement specification and use case description based on sentence transformer model. In *2023 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 220–226. IEEE.

Zuzanna Bartczak. 2024. From rag to riches: Evaluating the benefits of retrieval-augmented generation in sql database querying.

Diego Cabezas, Ricardo Fonseca-Delgado, Irma Reyes-Chacón, Paola Vizcaino-Imacana, and Milton Morocho-Cayamcela. 2024. Integrating a llama-based chatbot with augmented retrieval generation as a complementary educational tool for high school and college students. In *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*, pages 8–10, Dijon, France.

Songlin Chen, Weicheng Wang, Xiaoliang Chen, Peng Lu, Zaiyan Yang, and Yajun Du. 2024. Llama-lora neural prompt engineering: A deep tuning framework for automatically generating chinese text logical reasoning thinking chains. *Data Intelligence*, 6(2):375–408.

Showkat Ahmad Dar and Jan Iqra. 2016. Synchronization of data between sqlite (local database) and sql server (remote database). *IUP Journal of Computer Sciences*, 10(4).

Longxu Dou, Yan Gao, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Jian-Guang Lou, and Dechen Zhan. 2023. Unisar: a unified structure-aware autoregressive language model for text-to-sql semantic parsing. *International Journal of Machine Learning and Cybernetics*, 14(12):4361–4376.

Federal Highway Administration. 2024a. Fhwa infobridge: National bridge inventory and inspection data. `https://infobridge.fhwa.dot.gov/Data`. Accessed: March 28, 2025.

Federal Highway Administration. 2024b. Fhwa infotechnology portal. Accessed: Nov. 15, 2024.

S. S. Gadiraju, D. Liao, A. Kudupudi, S. Kasula, and C. Chalasani. 2024. Infotech assistant: A multimodal conversational agent for infotechnology web portal

queries. In *Proceedings of the 2024 IEEE International Conference on Big Data (BigData)*, pages 3264–3272, Washington, DC, USA. IEEE.

Vishakha Gupta, Ada Gavrilovska, Karsten Schwan, Harshvardhan Kharche, Niraj Tolia, Vanish Talwar, and Parthasarathy Ranganathan. 2009. Gvim: Gpu-accelerated virtual machines. In *Proceedings of the 3rd ACM Workshop on System-level virtualization for High Performance Computing*, pages 17–24.

Haeruddin Haeruddin, Sabariman Sabariman, and Vincent Su. 2025. Designing a chatbot application using the flask framework and rule-based algorithm. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, 7(1):133–142.

John Hunt, Chris Loftus, John Hunt, and Chris Loftus. 2003. Session management and life cycle monitoring. *Guide to J2EE: Enterprise Java*, pages 341–364.

Cheonsu Jeong. 2023. Generative ai service implementation using llm application architecture: Based on rag model and langchain framework. *Journal of Intelligence and Information Systems*, 29(4):129–164.

Arnav Jha, Naman Anand, and H. Karthikeyan. 2025. Conversion of natural language text to sql queries using generative ai. In *Hybrid and Advanced Technologies*, pages 25–32. CRC Press.

PK Liaw, HR Hartmann, and EJ Helm. 1983. Corrosion fatigue crack propagation testing with the krakgage® in salt water. *Engineering Fracture Mechanics*, 18(1):121–131.

Chuangtao Ma, Sriom Chakrabarti, Arijit Khan, and Bálint Molnár. 2025. Knowledge graph-based retrieval-augmented generation for schema matching. *arXiv preprint arXiv:2501.08686*.

Meta AI. 2024. Llama 3.1 8b instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct. Accessed: 2025-03-26.

Ali Mohammadjafari, Anthony S. Maida, and Raju Gottumukkala. 2024. From natural language to sql: Review of llm-based text-to-sql systems. *arXiv preprint arXiv:2410.01066*.

George Mason University Office of Research Computing. 2025. About hopper. Accessed: March 29, 2025.

Dijana Oreški and Dino Vlahek. 2024. Retrieval augmented generation in large language models: Development of ai chatbot for student support. *arXiv preprint arXiv:2403.12345*.

Piyush Pandey, Dhavalkumar Patel, Shreekant Mandvikar, and Naresh Kota. Ensuring data accuracy in text-to-sql systems: A comprehensive validation framework.

Kunal Relan. 2019. Deploying flask applications. In *Building REST APIs with Flask: Create Python Web Services with MySQL*, pages 159–182. Springer.

Marco Siino. 2024. All-mpnet at semeval-2024 task 1: Application of mpnet for evaluating semantic textual relatedness. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 379–384.

Bhargav Srinivasa-Desikan. 2018. *Natural Language Processing and Computational Linguistics: A Practical Guide to Text Analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd.

Yuli Vasiliev. 2020. *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press.

Raja Vavekanand and Kira Sam. 2024. Llama 3.1: An in-depth analysis of the next-generation large language model. Accessed: March 2025.

Haiyuan Wang, Deli Zhang, Jianmin Li, Zelong Feng, and Feng Zhang. 2025. Entropy-optimized dynamic text segmentation and rag-enhanced llms for construction engineering knowledge base. *Applied Sciences*, 15(6):3134.

Yiyun Zhao, Prateek Singh, Hanoz Bhathena, Bernardo Ramos, Aviral Joshi, Swaroop Gadiyaram, and Saket Sharma. 2024. Optimizing llm based retrieval augmented generation pipelines in the financial domain. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 279–294.

Appendix A: Backend Response Log

The following is a backend-generated response to a real user-issued Text-to-SQL query.

User Query: `*List bridges with average daily traffic over 220000.`

Generated SQL:

```
SELECT * FROM ‘Bridge_Basic_Information‘
WHERE ‘29 - Average Daily Traffic‘ > 220000
```

HTML Answer Snippet:

```
<html>
<b>Generated SQL:</b>
<pre>SELECT * FROM ‘Bridge_Basic_Information‘
WHERE ‘29 - Average Daily Traffic‘ > 220000</pre>
<b>Answer:</b>
<table border="1" class="dataframe">
...
</table>
</html>
```

Appendix B: Prompt Engineering Template

The following structured prompt is issued to guide the LLaMA model in generating syntactically correct and executable SQL queries based on user input and schema context. This prompt is automatically constructed at runtime.

Prompt Template:

```
You are a domain-aware SQL generation assistant.
Your goal is to generate exactly one valid SQLite SQL query
based on the provided database schema and user request.

========================
Database Schema:
{schema_json}
========================

Instructions:
1. Return only a single SQL query.
2. Do not include explanations, comments, or markdown formatting.
3. Use table and column names exactly as shown in the schema.
4. Wrap column names with spaces, dashes, or special characters in backticks.
5. Use JOINs where appropriate based on column relationships.
6. Ensure compatibility with SQLite syntax.
7. Avoid ambiguity or unsupported features.

Now generate a query for the following user request:
========================
User Request:
{user_input}
========================
```