# A Retrieval-Augmented Generation Approach to Conversational Analytics over Structured Data

Brian de Búrca
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00218725@myTUDublin.ie

## Introduction

Integrating Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) using a relational database can enable a chatbot to answer queries grounded in structured organisational data. This study examines three key aspects: (1) the accuracy and faithfulness of generated responses when compared with baseline results derived directly from structured data, (2) system responsiveness and usability for non-technical administrative and managerial users, and (3) the practical feasibility of adopting and maintaining such an approach within resource-constrained organisations.

## Motivation

The motivation for this research is to evaluate whether organisations with limited IT budgets and resources can successfully adopt AI through a low-cost solution. The proposed approach aims to be easy to implement, use, and maintain, without requiring significant financial or technical investment. As AI becomes an increasingly integral part of everyday business operations, this study seeks to provide smaller organisations with insight into the potential value of AI by demonstrating a low-cost, low-code solution that can be effectively managed in-house.

## Research Questions

1. Can a RAG-based chatbot accurately interpret user input and generate logically consistent responses based on the source data?

2. Does the chatbot produce consistent responses to semantically similar queries while correctly maintaining conversational context?

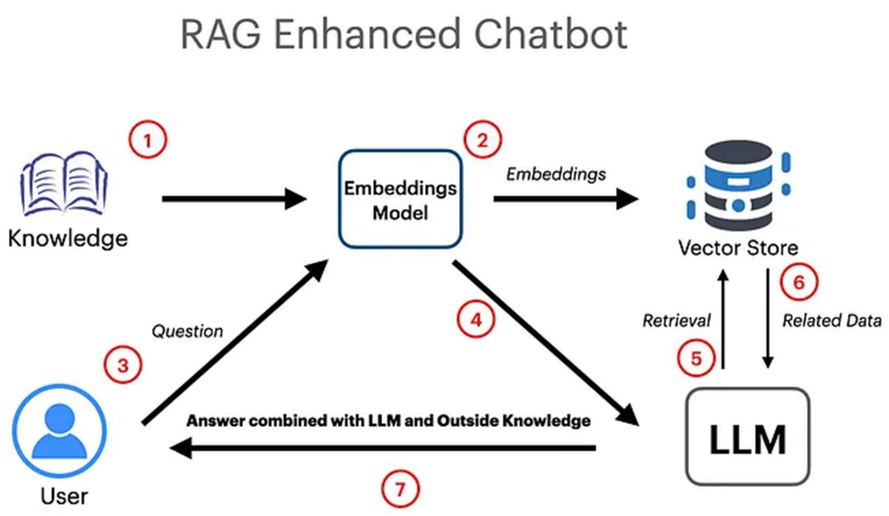
3. Can the solution be implemented and maintained at a low cost while remaining scalable?

## How a RAG Chatbot Works

The main objective of a RAG chatbot is to interpret a user's query and retrieve relevant information to generate an accurate and context-aware response.

**1. Knowledge and Embeddings**

Knowledge is extracted from a **relational database** and transformed into numerical representations using an embedding model, such as *text-embedding-3-small* from OpenAI or *NV-Embed-v1* by NVIDIA. These embeddings capture the semantic meaning of the data for efficient retrieval.




**2. Vector Store**

The generated embeddings are stored in a vector database, such as AstraDB. Each data entry is assigned a vector representation, enabling similarity-based searches to efficiently retrieve the most relevant information for a given query.

**3. Large Language Model**

The Large Language Model (LLM) combines the user's query with the relevant results retrieved from the vector database. Using this contextual information, the LLM generates a coherent and natural-language response tailored to the user's question.

## Building out the chatbot

**Step 1.** Create a vector database using `astra.datastax.com`.

**Step 2.** Populate AstraDB with data extracted from three sources representing different business contexts:

- Election data
- Student grades
- Electronic store sales (small openai embedding)
- Electronic store sales (Large openai embedding)

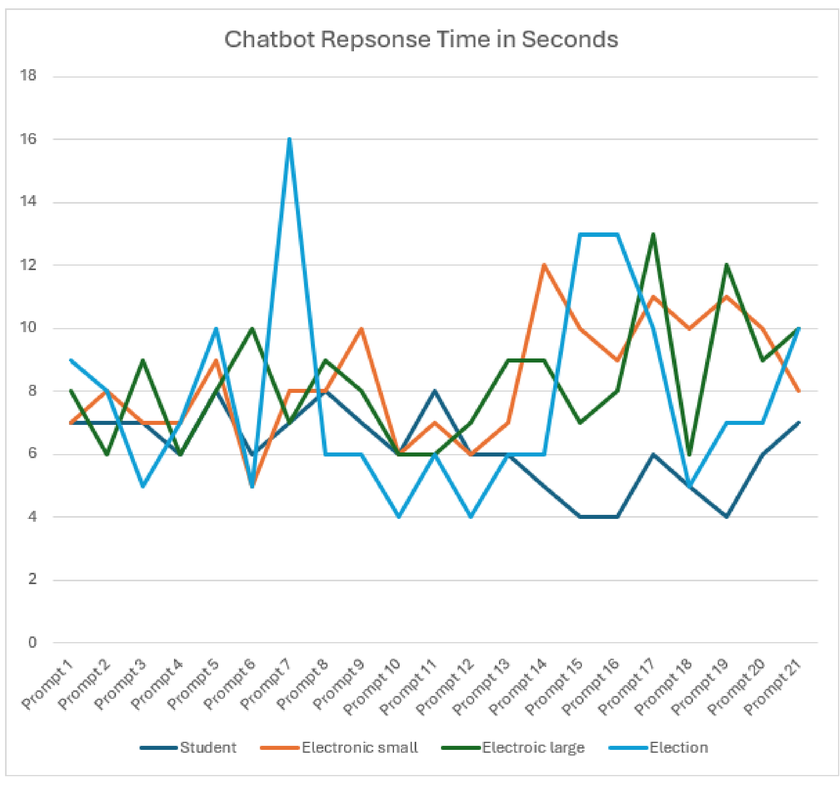
**Step 3.** Create a VM to host the Langflow application from a container.

**Step 4.** Build one flow per data source for testing.

**Step 5.** Evaluate each flow based on:

- Query response time
- Response quality compared to the original data source
- Anomalies in result sets

The image is the response times for each prompt given the different datasets




## Conclusions and Future Work

The results indicate that the chatbot can generate contextually relevant and interpretable responses, demonstrating the potential of low-code, LLM-based systems for organisational access to structured data. However, performance is less reliable for aggregation-heavy and numerically precise queries. These limitations are primarily linked to embedding-based retrieval, chunking strategies, and top-$k$ selection. For the student dataset, response accuracy improved when grade-related queries were routed to a direct database/API call rather than relying solely on retrieval from the vector store. This approach improves reliability for count-based and numeric queries while maintaining conversational quality for general queries.
Future work will explore improved embedding and chunking strategies to increase retrieval precision and response faithfulness. Further evaluation will also compare alternative low-code platforms against the Langflow implementation in terms of accuracy, usability, and cost.

## QR Code for Recording