CODE & COFFEE
Boston

CIC INNOVATION CAMPUS
Location Sponsor

Build a RAG-based chatbot

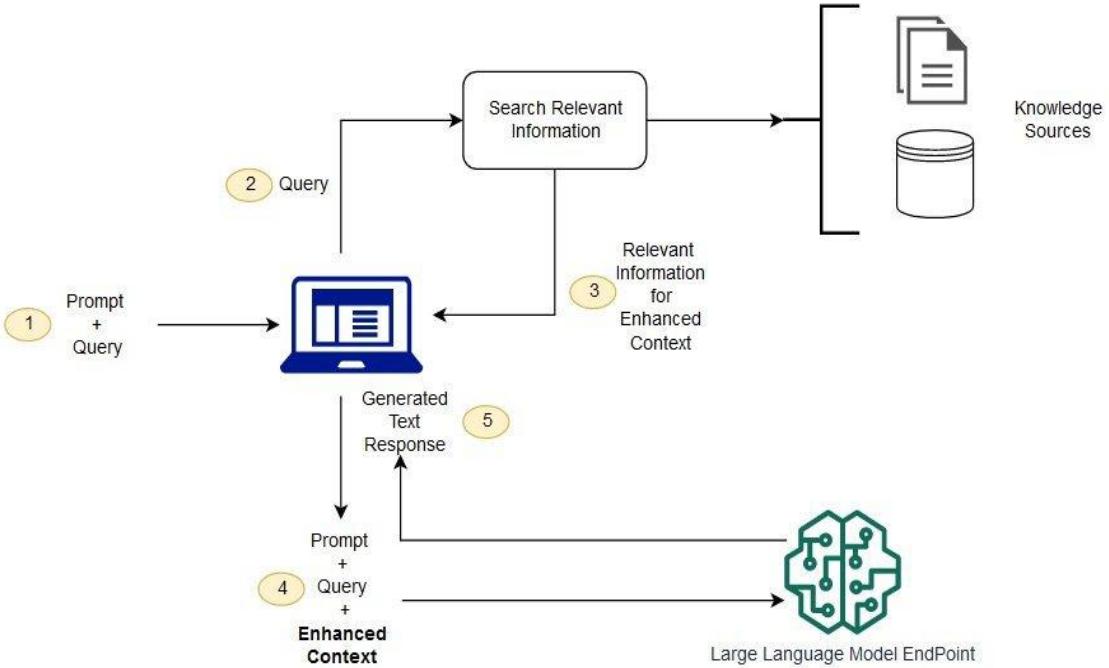
hosted by
Nazar Mammedov

What is Retrieval-Augmented Generation (RAG)?

- Retrieval-Augmented Generation (RAG) is a powerful technique that combines information retrieval with language model generation helpful for answering factual or domain-specific questions.

- Alternatives to RAG
  - Fine-tuning the model - retraining the model with your own data
  - Tool-Calling / Function Calling - let the model find additional information by calling external tools/API
  - etc.

What is Retrieval-Augmented Generation (RAG)?

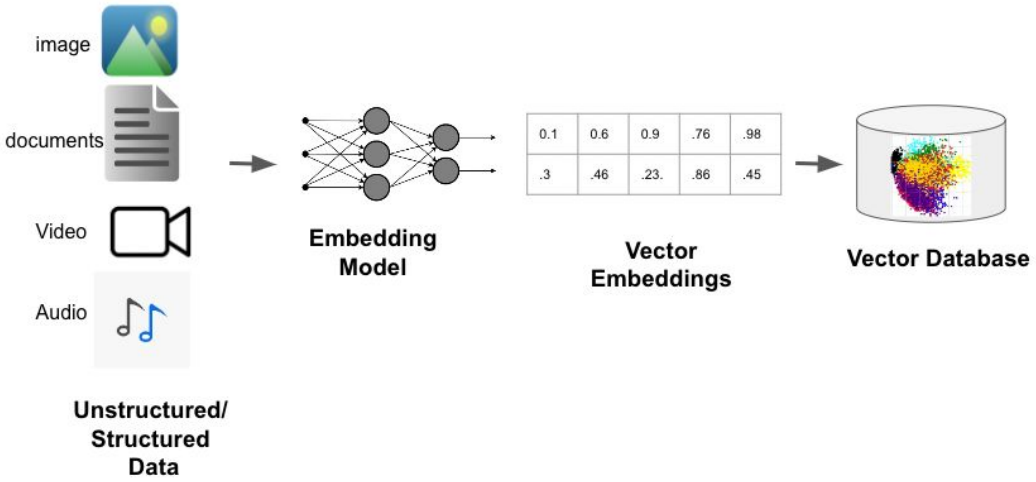


Source: https://aws.amazon.com/what-is/retrieval-augmented-generation/

What do we need to build a RAG chatbot?

- Large language model (LLM) to understand and construct proper sentences
  - Server running the LLM
- Embedding model to convert human-text data to numeric (vector) data
  - Server running the embedding model
- Vector database storage to store company or domain specific information
  - Server running the database

What is embedding?



Source: https://zilliz.com/learn/what-are-binary-vector-embedding

Online Tools for visualization

Tokenizer:
https://huggingface.co/spaces/Xenova/the-tokenizer-playground

Embeddings:
https://projector.tensorflow.org/

What will we use?

- IDE
  - VS Code
- Backend
  - Flask to create the web application
  - Chroma as vector database
  - Accessing remotely Google Gemini API
  - Running local server with LLM and embedding model in LM Studio (or Ollama)
- Frontend
  - Simple HTML page with JQuery and TailwindCSS
- Simple REST API testing
  - Bruno API client: download at https://www.usebruno.com/

Backend steps

1. Install Python
2. Download git repo: *git clone* https://github.com/berkesas/rag-chatbot/
3. Create virtual environment: *python -m venv venv*
4. Install Python packages: *pip install -r requirements.txt*
5. Create own API_KEY on Google (if needed)
6. Modify source files with API_KEY and custom parameters
7. Run the app: *python app.py*
8. Test with Bruno API client or directly with the frontend app

Frontend steps

- Download git repo (if needed): *git clone* *https://github.com/berkesas/rag-chatbot/*
- Install Live Server extension for VS Code (or an alternative)
- Run the frontend with “Go live”

Beyond basics

- Creating stateful chat sessions (remembering previous answers in the chat)
- Running vector database as a server
- Optimizing chunk sizes for your custom needs
- Optimizing prompts for your custom needs
- Streaming responses from the server