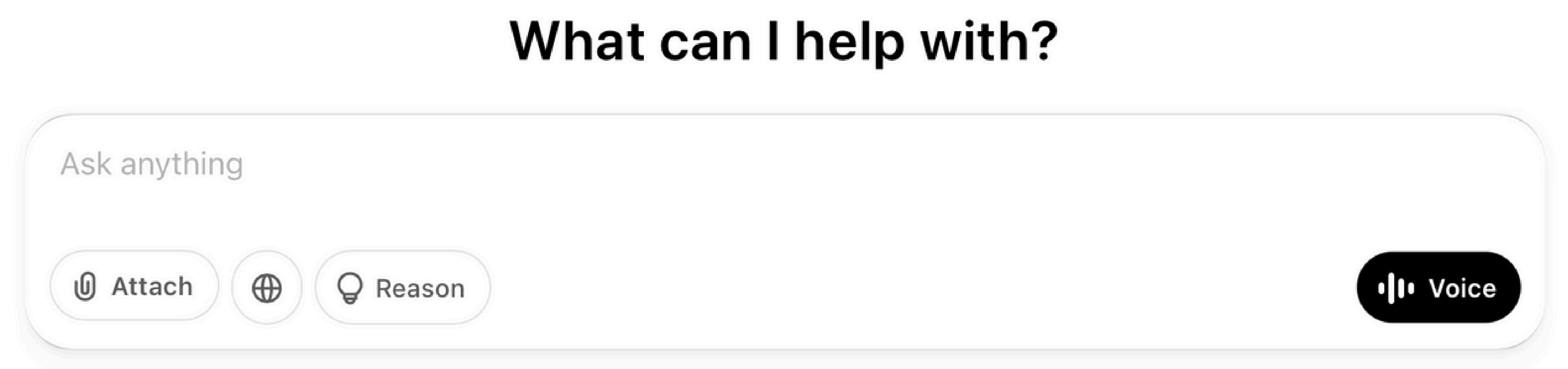
March 2025

Vector DBs and LLMS



What can I help with?

Practical 2

Grace Kelner, Milena Perez-Gerus, and Claire Mahon

Team Members

Grace Kelner

Milena Perez-Gerus

Claire Mahon

Overview

In this project, we built a local Retrieval-Augmented Generation (RAG) system to query and generate responses from a collection of course notes using various technologies, including embedding models, vector databases, and local LLMs. We experimented with different configurations of text preprocessing, chunking strategies, embedding models, and vector databases to evaluate and optimize the system's performance, ultimately recommending the best pipeline based on our findings.

Vector Databases

Chroma DB

Dedicated open-source vector DB built for AI applications. Embedded metadata support and python integration. Great for academic uses.

Weviate DB

Schema based and API driven dedicated vector DB with hybrid queries, built in modules, and rich queries to support contextual search.

Redis DB

Open-source, in-memory data structure store that supports vector search that is renowned for its speed and minimal latency. Designed to retrieve vector embeddings of complex and high-dimensional data.

Embedding Models

mpnet-basev2

Uses a novel pre-training method, named masked and permuted language modeling, to inherit the advantages of masked language modeling and permuted language modeling for natural language understanding.

Nomic

Open-source, long-context (8,192 tokens) text embedding model that surpasses OpenAI ada-002 performance on short and long context tasks.

all-MiniLM-L6-v2

Lightweight and computationally efficient model trained to generate high-quality sentence embeddings. Takes a sentence or short paragraph and maps it into a 384-dimensional vector.

Data Sources

Class Notes

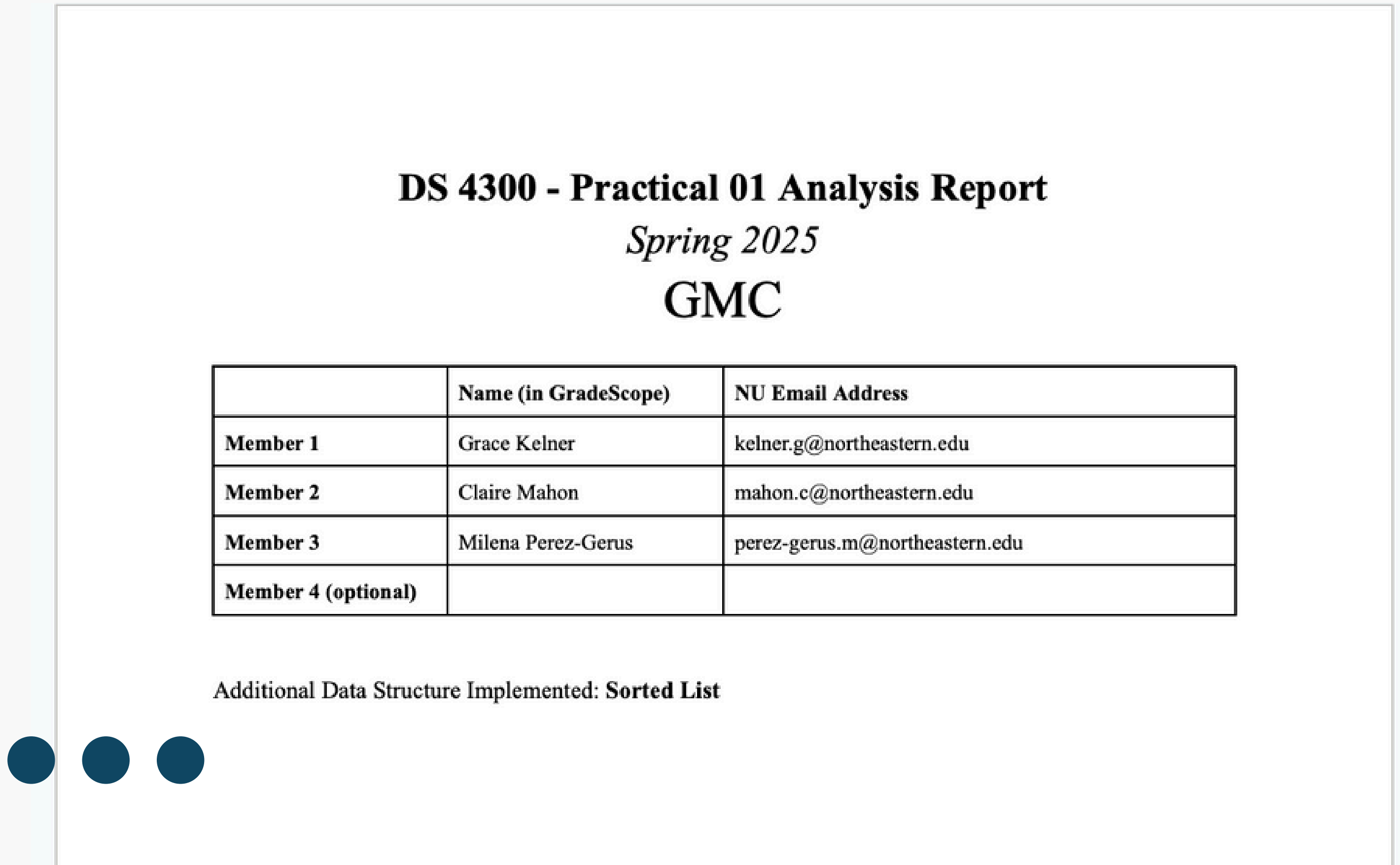
Notes taken in class, on docs or by hand and typed to docs

Homework and Project Submissions

Files from previous homeworks and Practical 1 Project submission

Course Material

PDFs and Code Examples from Dr Fontenot



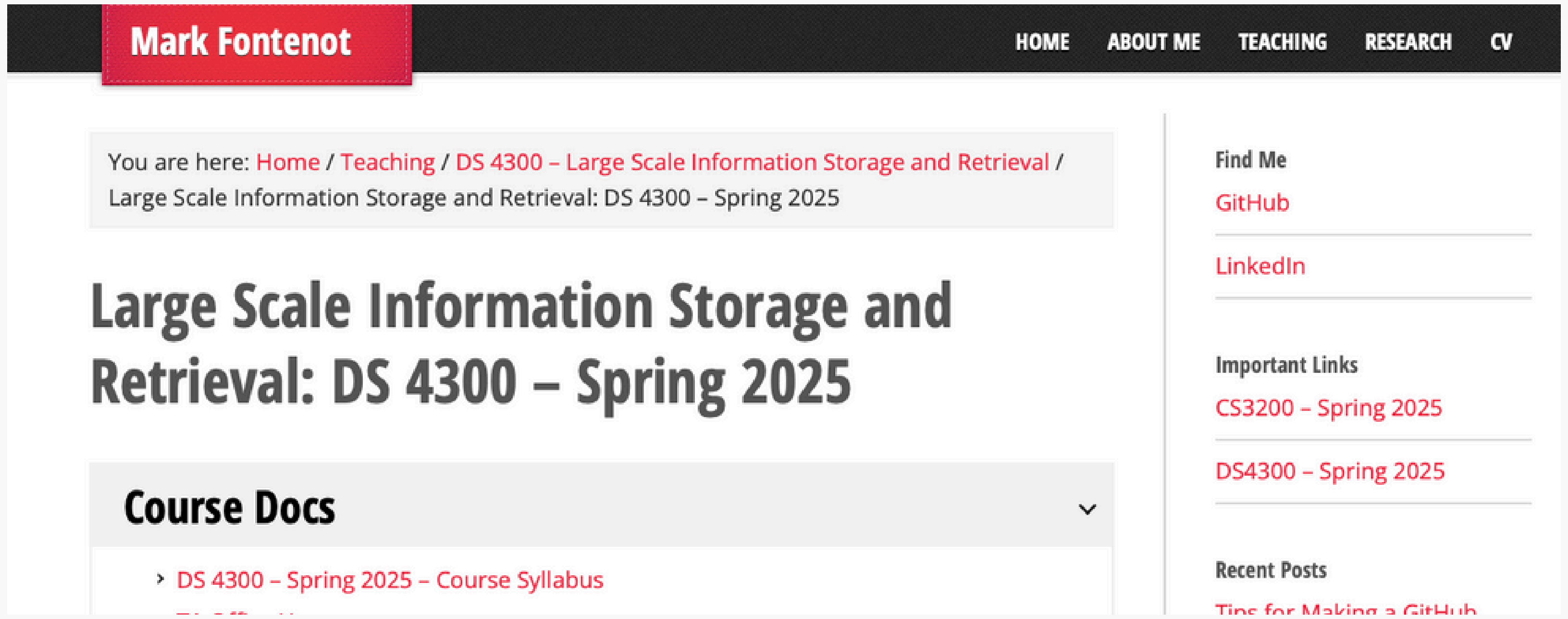
DS 4300 - Practical 01 Analysis Report

Spring 2025

GMC

	Name (in GradeScope)	**NU Email Address**
Member 1	Grace Kelner	kelner.g@northeastern.edu
Member 2	Claire Mahon	mahon.c@northeastern.edu
Member 3	Milena Perez-Gerus	perez-gerus.m@northeastern.edu
Member 4 (optional)		

Additional Data Structure Implemented: **Sorted List**



You are here: Home / Teaching / DS 4300 – Large Scale Information Storage and Retrieval / Large Scale Information Storage and Retrieval: DS 4300 – Spring 2025

Large Scale Information Storage and Retrieval: DS 4300 – Spring 2025

Course Docs

- DS 4300 – Spring 2025 – Course Syllabus

Find Me

GitHub

LinkedIn

Important Links

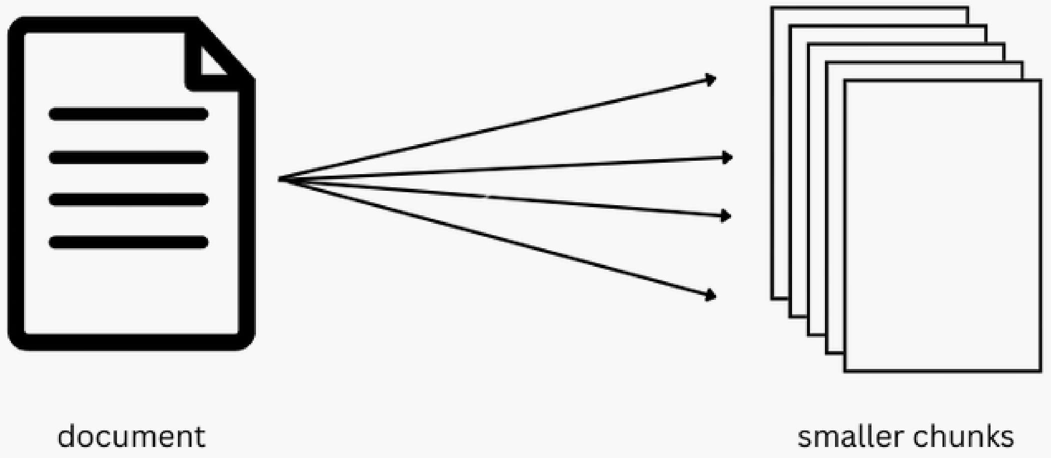
CS3200 – Spring 2025

DS4300 – Spring 2025

Recent Posts

Tips for Making a GitHub

Data Processing



Processing the articles:

- Text Extraction - used PyMuPDF (Fitz) to extract text from PDFs, processing each page separately.
- Chunking Strategy - Split text into smaller chunks to improve embedding efficiency and search accuracy.
- Balanced memory usage and processing speed while ensuring meaningful context in each chunk.

Chunk testing:

- Size testing - Started with 500 word chunks, tried 200 (smaller) and settled on 300 as the best tradeoff between context retention and embedding quality.
- Overlap testing - Experimented with overlaps ranging from 20 to 200. 50 was the best in order to preserve sentence contuinuity between chunks.

LLMs for Queries

Llama3.2

A collection of powerful language models developed by Meta. Designed to improve efficiency and performance of large-scale language models, with language understanding and image reasoning capabilities.

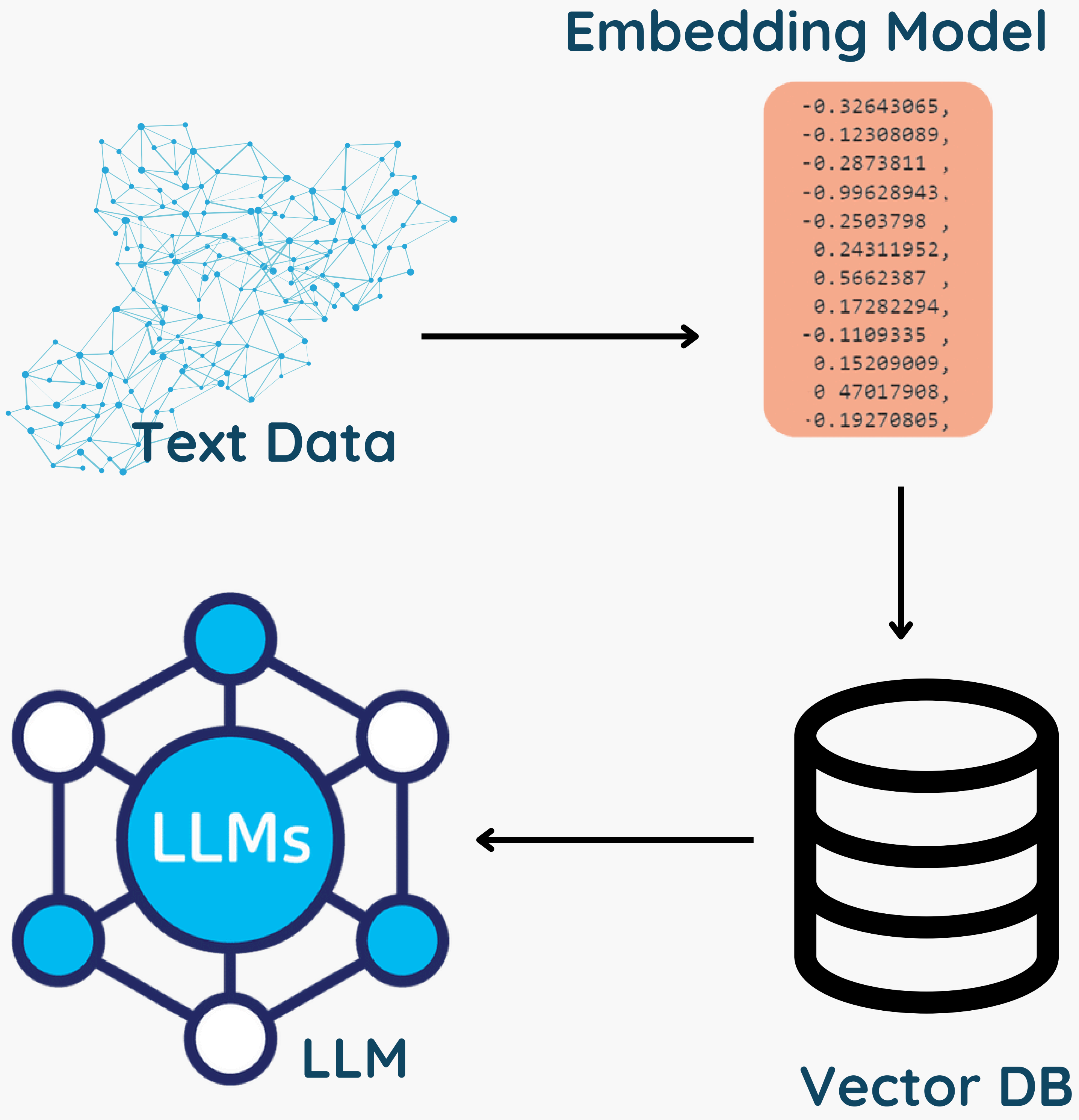
Mistral

Open source LLM models with a focus on efficiency and performance, offering multimodal capabilities, long context windows, and strong reasoning, knowledge, and coding capabilities.

Experimental Design

We utilized the following pipeline designs...

Pipeline
Weviate Nomic Mistral
Weviate Nomic llama3.2
Chroma mpnet-base-v2 Mistral
Chroma mpnet-base-v2 llama3.2
Redis MiniLM Mistral
Redis MiniLM llama3.2

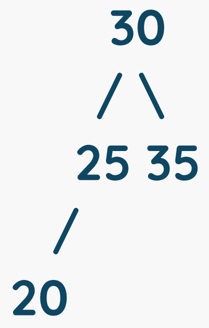


```
-0.32643065,
-0.12308089,
-0.2873811 ,
-0.99628943,
-0.2503798 ,
 0.24311952,
 0.5662387 ,
 0.17282294,
-0.1109335 ,
 0.15209009,
 0 47017908,
·0.19270805,
```

User Queries

We used the following queries relevant to both the class, and our own notes

1. Who is the professor for DS 4300?
2. What is linear search?
3. Add 23 to the AVL Tree below. What imbalance case is created with inserting 23?
4. Succinctly describe the four components of ACID compliant transactions.
5. Write a Mongo query based on the movies data set that returns the titles of all movies released between 2010 and 2015 from the suspense genre.



Data Collected

Some examples of queries our model returned ...

Write a Mongo query based on the movies data set that returns the titles of all movies released between 2010 and 2015 from the suspense genre?

Redis Minilm Mistral: “db.movies.find({ "year": { $gte: 2010, $lte: 2015 }, "genres": "Suspense" }, { "_id": 0, "title": 1 })”

Retrieved in 14.97 seconds

Who is the professor for DS4300?

Weviate Nomic llama3.2: “The professor for DS4300 appears to be Mark Fontenot, who is also a PhD at Northeastern University.”

Retrieved in 5.21 seconds

What is a linear search?

Chroma Mpnet Mistral: “ A linear search is the most common operation performed by a database system, where you start at the beginning of a list and proceed element by element until either you find what you're looking for or you get to the last element without finding it. In the worst case, the time complexity of a linear search is O(n).”

Retrieved in 12.05 seconds

Results - Pipeline Testing Summary

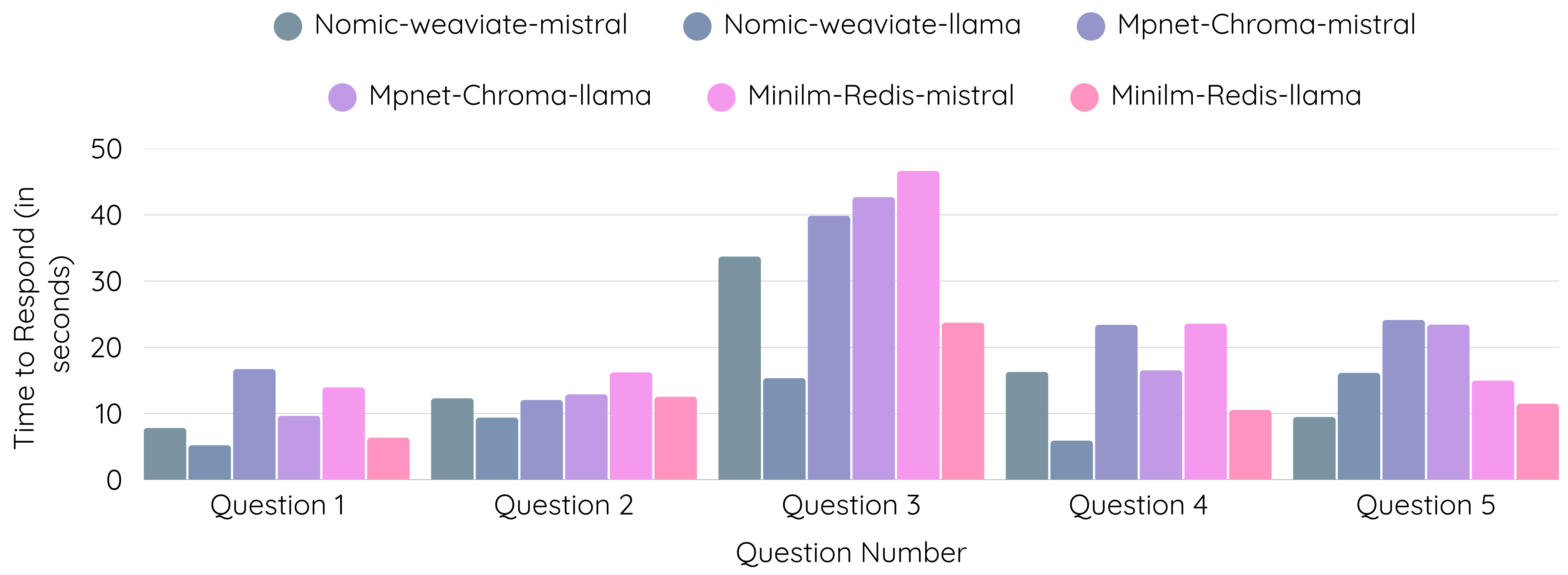
Performance from the following combinations of embedding models, dbs, and LLMs

Pipeline	Average Timing	Answer Quality
Weviate Nomic Mistral	15.92 s	Concise and accurate
Weviate Nomic llama3.2	10.40 s	Overly wordy
Chroma mpnet-base-v2 Mistral	23.23 s	Fairly accurate
Chroma mpnet-base-v2 llama3.2	21.03 s	Overly wordy
Redis MiniLM Mistral	23.07 s	Concise and accurate
Redis MiniLM llama3.2	12.93 s	Overly wordy

Results - Timing Performance

The best LLM timing performance was ...

Nomic-Weaviate-llama3.2



Results - Answer Quality

The best LLM answer quality was ...

Nomic-Weaviate-Mistral

Succinctly describe the four components of ACID compliant transactions.

The four components of ACID (Atomicity, Consistency, Isolation, Durability) compliant transactions are as follows:
1. Atomicity - A transaction is treated as an atomic unit; it is either fully executed or not at all.
2. Consistency - A transaction takes the database from one consistent state to another consistent state, ensuring that all data meets integrity constraints.
3. Isolation - Two transactions cannot affect each other when they are being executed simultaneously. This prevents problems like Dirty Reads, Non-repeatable Reads, and Phantom Reads.
4. Durability - Once a transaction is committed successfully, its changes become permanent and remain preserved, even in the event of a system failure.

or

MiniLM-Redis-Mistral

Succinctly describe the four components of ACID compliant transactions.

The four components of ACID (Atomicity, Consistency, Isolation, Durability) compliant transactions are as follows:
* Atomicity - a transaction is treated as an atomic unit, meaning that it is either fully executed or no parts of it are executed.
* Consistency - a transaction takes the database from one consistent state to another consistent state; all data must meet integrity constraints within this state.
* Isolation - each transaction executes independently of other transactions, ensuring that concurrent operations do not interfere with one another.
* Durability - once a transaction is completed and committed successfully, its changes are permanent, even in the event of a system failure; committed transactions will be preserved.

Final Recommendation

For a team of similar size (2-4) wanting to create a RAG-based system for use on an exam...

Our Recommendation

Vector DB:

Weaviate

Embedding model:

NOMIC

LLM:

This architecture returned the best combination of answer quality, succinctness, and time performance.

Thank you