Practical 2 Report

HashMap<Jonathan, Shishir>

Team:
Shishir Pokhrel
Jonathan Ding

GitHub Link: https://github.com/pokhrel-sh/RAG

Overview

Retrieval-Augmented Generation (RAG) system where the LLM are trained based off of the course data and materials (DS 4300 - Large Scale Information Retrieval)

How did we achieve this?

- Gather the data (such as class notes) in a pdf format. Extract the text and create chunks
- Store the vector embeddings (bunch of numbers, done by paraphrase, nomic, minilm) in databases such as Redis, Chroma or Qdrant and retrieve relevant chunks based on query embedding
- Used LLMs such as Deepseek, mistral, llama3.2 to contextualize the retrieved chunks.
- Used over 30 different combinations to find the most accurate combination (based off our intuition)

Vector Databases

Redis

- In memory database, very fast lookup.
- Easy to set up (it was done for us) and very easy to use.

Chroma

- Made for embedding storage and retrieval and RAG.
- Library was convenient, but little trouble getting it to work. A little slow (took 5 hours to complete all combinations (9)).

Qdrant

- Little difficult to set up and use it.
- Needed a library, and docker image to use it, but was super quick in terms of getting the data.

Embedding Models

Paraphrase-multilingual

- Trained on multiple languages, works well for cross-lingual retrieval tasks
- We did not use other languages and this felt the same as the other embedding models

Nomic-embed-text

- Very good embeddings as most of the information was accurate while using nomic
- A bit too slow to embed all of the stuff. Took me about 5 - 15 min to embed 55mb worth of data

All-minilm

- EXTREMELY FAST and SUPER LIGHT WEIGHT
- Less accuracy compared to others (but who cares, its FAST)

Data Processing

Document preprocessing and chunk sizes/overlap

- Stripped the PDF into text, cleaned the texts
- Created chunks of texts (100, 250, 500)
- Created the overlap of the chunks (50, 125, 250)
- 50% overlap gave the best result on average as there was more context from each side of the isle.

LLMs for Queries

Deepseek

- Provided detailed answers with reasoning.
- Reasoning took forever, would not use it if I need information fast.

Ollama3.2

- Provided accurate and relevant information quickly
- Did not go into depth leaving out some important information

Mistral

- Worked the fastest, convenient for quick responses
- Said “Dr. Fontenot is a professional chef in New Orleans, Louisiana”, so tends to hallucinate.

Experimental Design

We wrote three scripts (one for each database). The script will store the text in given chunks, embedding and answer using the LLM.

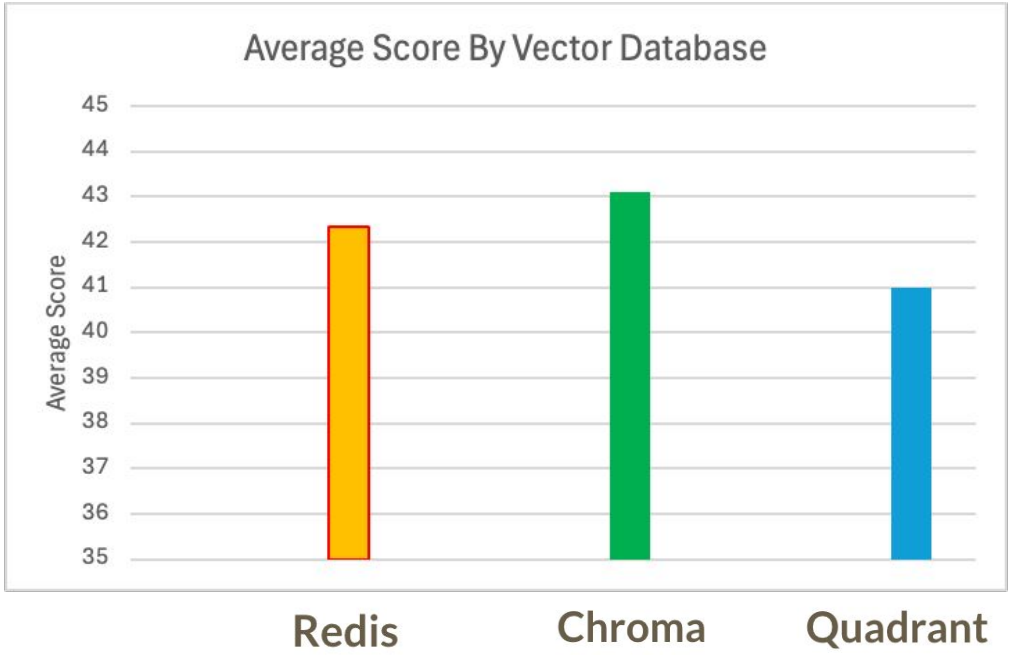
There were 27 combination with just 1 chunk size.

We then tried 3 chunk sizes for the best-performing pipeline.

TLDR: Brute forced every possible combination

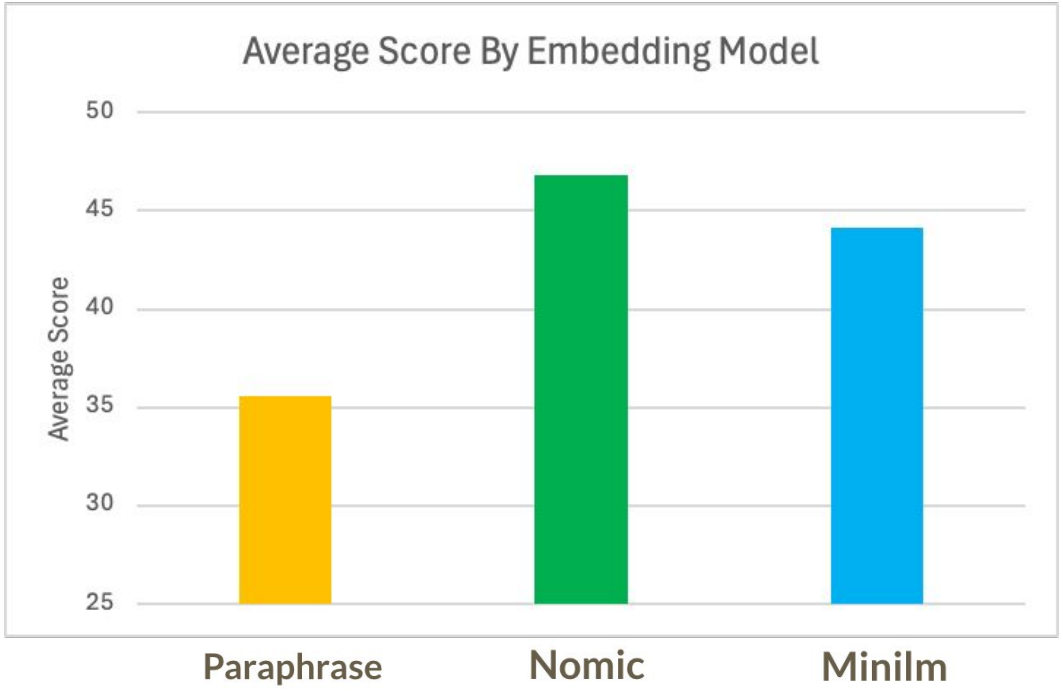
Vector Database Data

Average score for Chroma was higher than QDrant. Redis wasn't as far back.



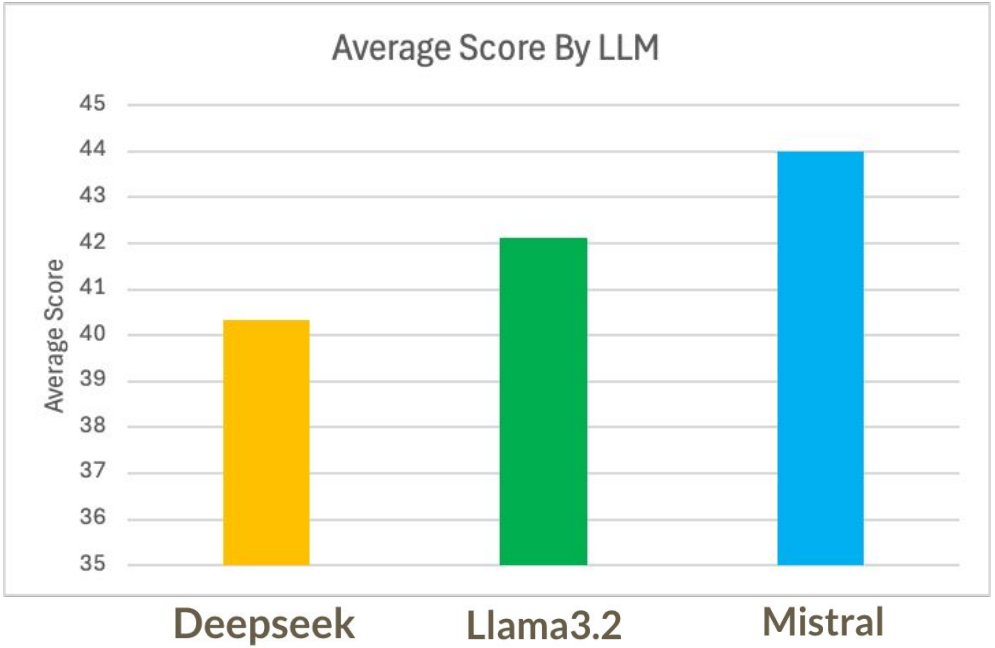
Embedding Model Data

Average score for Nomic far outpaced Paraphrase, but minilm was right behind!



LLM Data

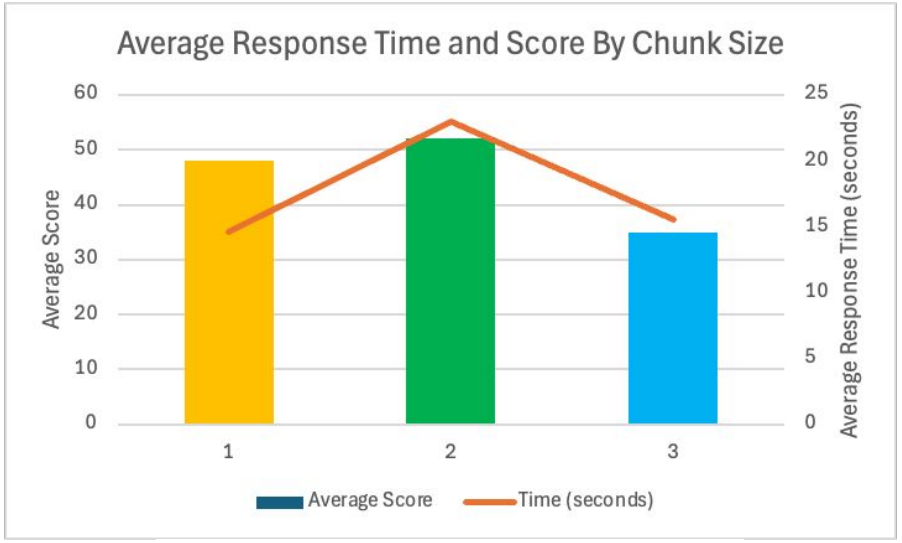
Mistral performed best, with Llama3.2 taking second place and Deepseek at the end.



Chunk Size Data

The 100 and 500 token chunk size queries response times (orange line) were similar, while the 250 token response time was the highest.

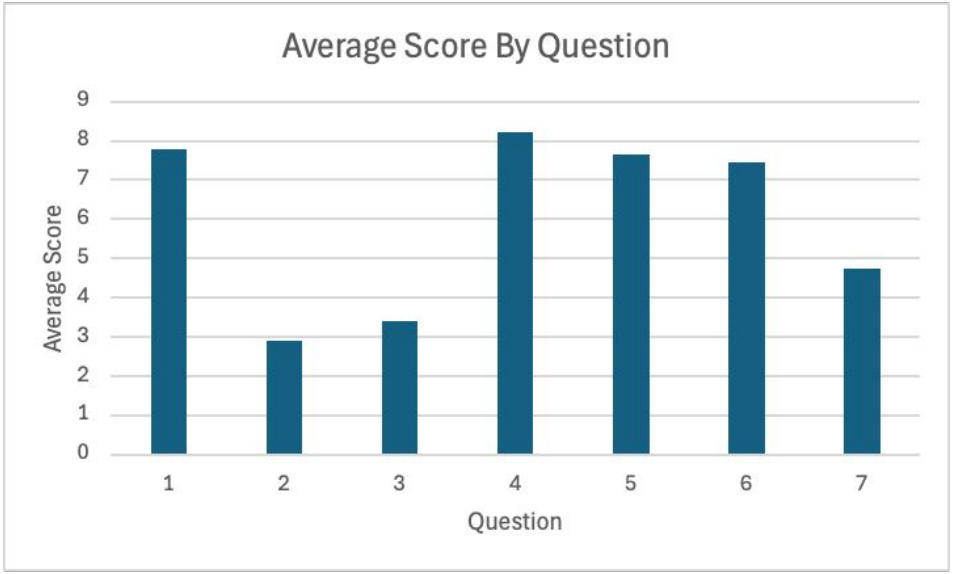
100 and 250 chunk size had similar average scores, while 500 had a lower average score.



Question Data

Questions 2 and 3 were the toughest, while most pipelines did well on questions 1, 4, 5, and 6.

Question 7 had the most variance, being mostly all-or-nothing.



Link to questions

Data Analysis

We created a set of questions related to the training material that we used for each pipeline. We then assigned a score to each response based on relevancy, accuracy, and usefulness.

- *We wrote three scripts. One for each databases. The script will store the text in given chunks, embedding and answer using the LLM.*
- *There were 27 combinations with just 1 chunk.*
- *We then rated each answer from 0-10. Totaled the score for all 7 questions. Found the highest score.*
- *The combination with the highest score is our best pipeline.*
- *We then used the same pipeline to answer the questions in 2 different chunks, and recalibrated the decision within the pipeline*

Recommendation

HashMap<Jonathan, Shishir>'s recommendation is using the following pipeline

- Embedding: Nomic
- Vector Database: Redis
- LLM: Mistral

since it had the highest score.

For chunk size, there were tradeoffs between response time and response quality.

A chunk size of 100 tokens showed the best response time without sacrificing much response quality.