**Coreshield - Chatbot**

**Model Details**

We have agreed on Mistral 7B for now. I have found 2 ways to use it:

1. Using Ollama - This is easy to use locally. Uses 4 -bit quantization so is also running good on my laptop. We can set the temperature and other parameters using a ‘Makefile’ . It applies 4 Bit quantization, so the whole model is only 4.3GB. However, it does not Supports fine-tuning via PEFT (LoRA, QLoRA). So, for RAG we must implement a separate langchain pipeline ( I have implemented a sample code below).

Sample Modelfile :

A black screen with white text

Description automatically generated

1. Using HuggingFace : Tried many times, not enough RAM in my laptop. Inference is paid, so we must download with transformers. Also, downloading provides more hugging face features upstream for fine tuning. I have 8GB RAM, it will run in 16 or 12GB I guess.

**RAG**

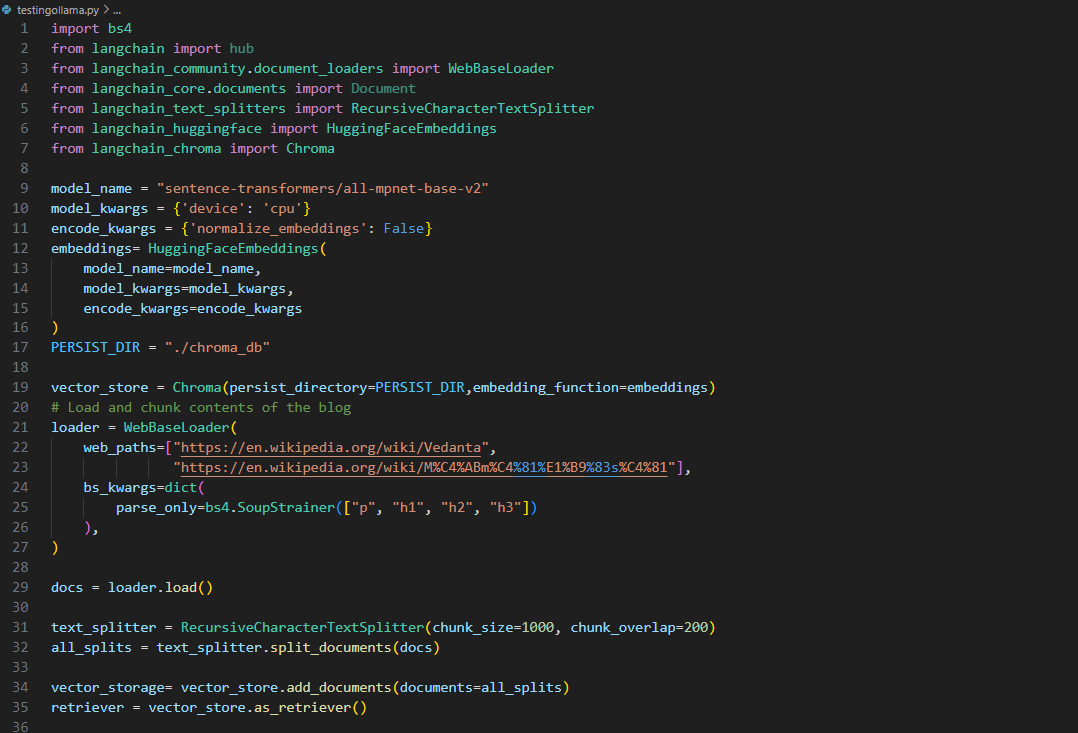
Firstly, we need to store the data, like relevant Docs in a vector database like chromaDB. We can do that using langchain.

Example :

For RAG implementation, there are two ways :

1. Use Langchain vector storage. I implemented it using two sample wiki pages (related to my coursework). The results were fine.

I used ‘ollama’ library in python to infer from mistral downloaded in my laptop.

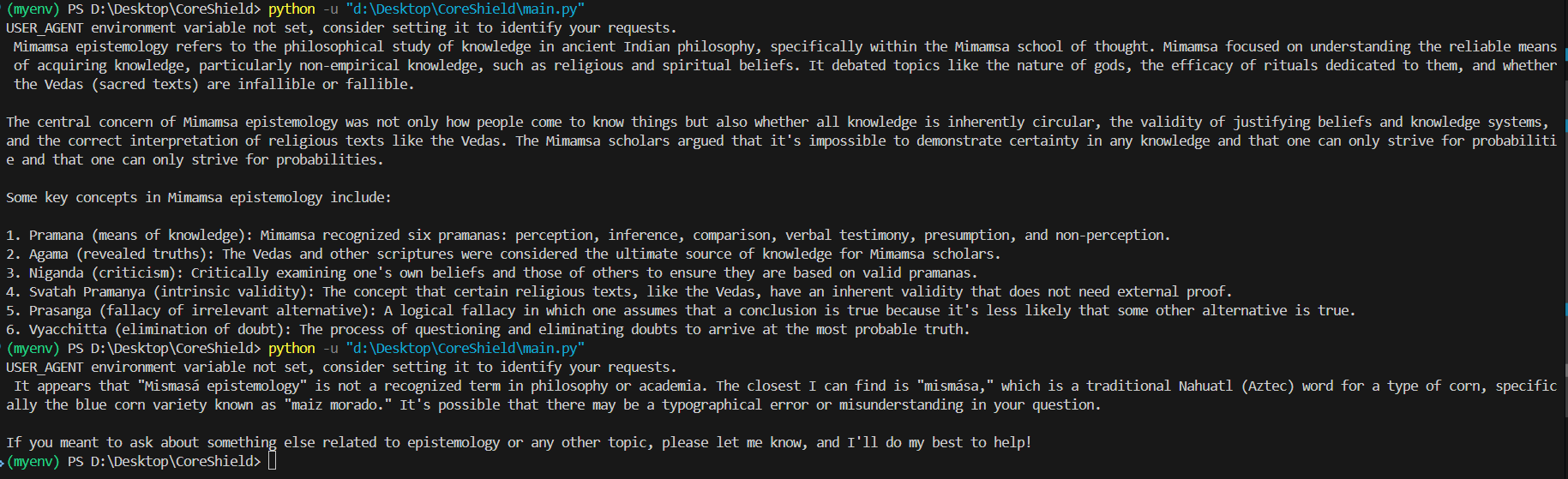


main.py

A screen shot of a computer program

Description automatically generated

The following screenshot shows output with and without RAG:

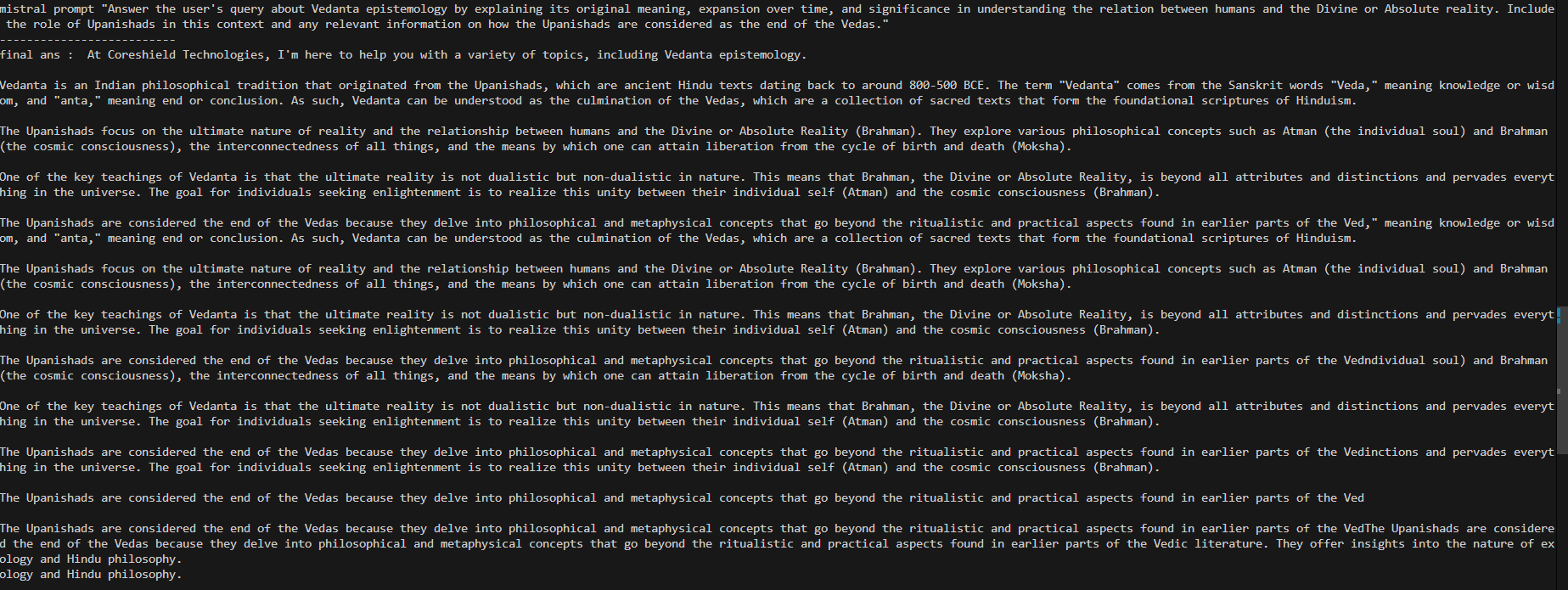


Agentic AI

I tried to use another LLM in between to refine the input to the final LLM. I used the prompt\_template method of Lang chain for this purpose.

The result was fine. We can refine the template and generate better results.





For better RAG implementation, we can use Tavily API key, it uses an AI agent to do real-time web search based on the user query. It can provide context as well as the source links. (I used it in one hackathon). We can then provide the fetched links to our mistral model. But it is free only till 1000 API calls per month.

<https://tavily.com/#features>

**Reranking Algorithms**

Suppose our RAG gives us 100 documents from the database as relevant to the prompt. We can add a layer of reranking algorithms, which ranks the docs based on some scoring function. We can then filter out the most relevant docs and pass it to the LLM as context. I will research about such algorithms. Code implementation needed.

**Knowledge Graphs**

Stores data in the form of entity relation pair, in the form of a directed graph. This can be implemented improve the RAG further. Neo4j is the most popular graph database that I found on the web. Will try to implement its code this week.

**Pipeline :**

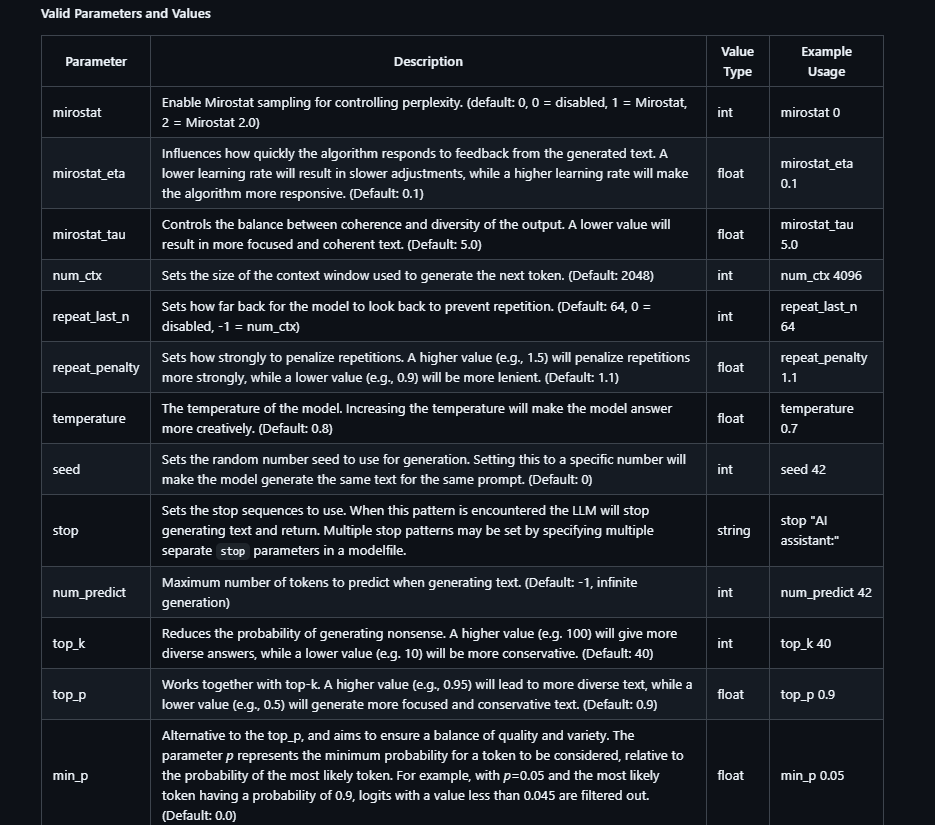
**Prompt 🡪 Knowledge Graph 🡪 LLM agent 🡪 Vector databases (Agent choose the best one) 🡪 Reranking system 🡪 LLM ( with prompt and context ) 🡪 Response.**

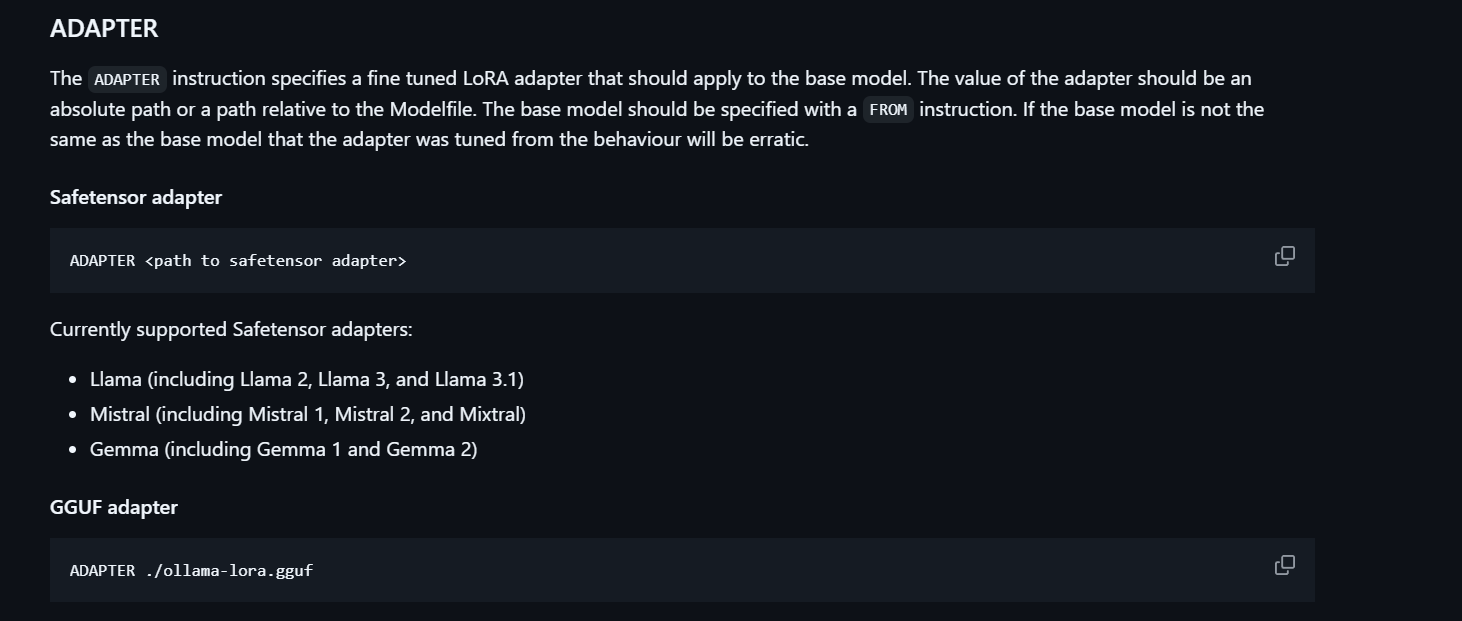
Crawl4AI 🡪 web Crawler and Scrapper, especially for LLM. (Open Source), faster than langchain’s WebBaseLoader, due to parallel processing.

Tasks :

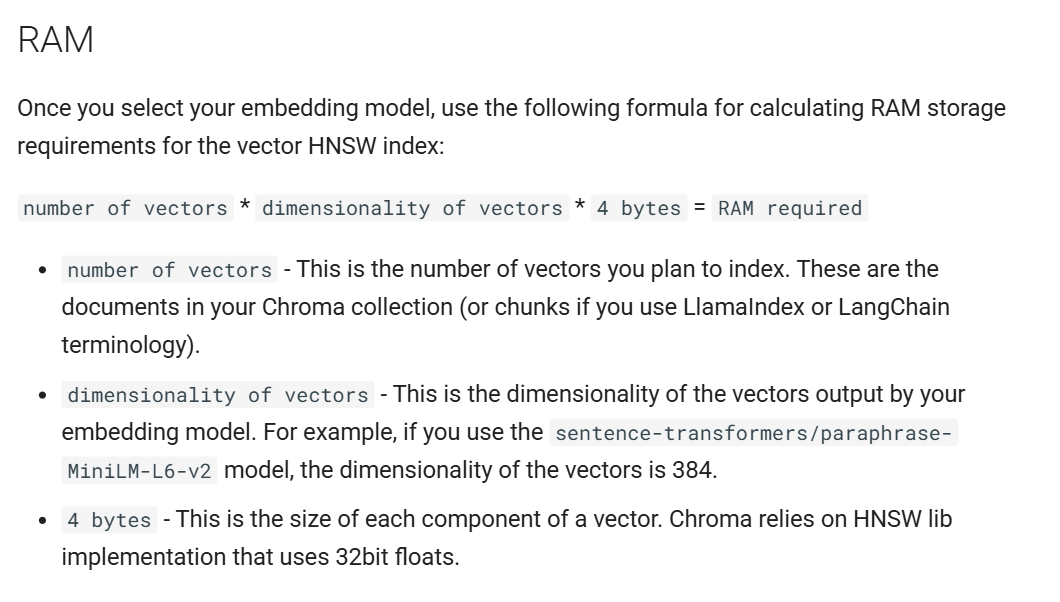
1. Upload the pdf, context that pdf only.
2. What is vector DB limit ?
3. What is the upper limit of the documents that can be uploaded? Bench marking
4. Supervised learning for better results

<https://github.com/ollama/ollama/blob/main/docs/modelfile.md#valid-parameters-and-values>





For chromaDB



For document only context :

We can do a cosine similarity search for the query and the chunks in the document. Now, set a threshold for the value obtained, say 0.8. If threshold is not met, se a hard RAG for the bot to answer as “Answer not available in the given document” or whatever.

Agentic RAG for doc only context.

Can use supabase, it is a mixed database type, can store vector embeddings as well as normal texts and urls, and can run semantic search on both. Good for defining ‘tools’ for agentic RAG implementation, also good for document only context. Open-source Code for the implementation :

<https://github.com/coleam00/ottomator-agents/tree/main/crawl4AI-agent>

A screenshot of a black and green website

Description automatically generated