1. **Ways to improve the accuracy and reliability of LLMs**
2. **Finetuning**

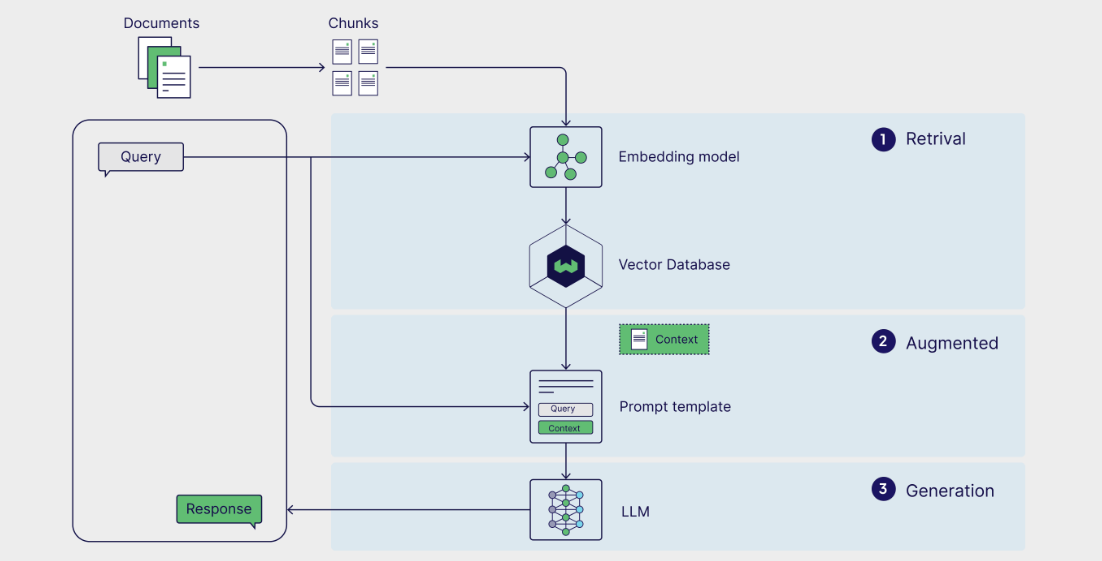
Cons:

Requires high computation Power

<https://www.youtube.com/watch?v=T2Fp67y6d24>

<https://www.youtube.com/watch?v=4MTtfTZnH5Y&t=355s>

1. **In context learning (Adding Examples)**
2. **Retrieval Augmented Generation (RAG)**

****

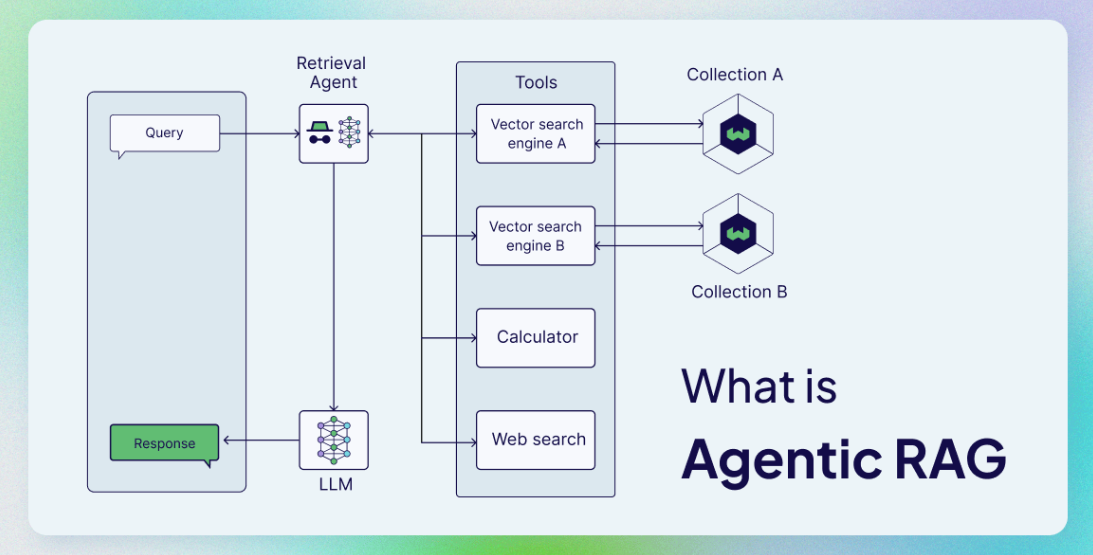
1. **Agentic RAG**

<https://www.youtube.com/watch?v=7GhWXODugWM&t=35s>

<https://www.youtube.com/watch?v=tioFxI_WTkk>

<https://www.youtube.com/watch?v=kkF9gsBF7gg&t=156s>

<https://weaviate.io/blog/what-is-agentic-rag>



Pros:

**Best LLM from Huggingface for RAG** - <https://www.perplexity.ai/search/define-output-schema-for-the-l-3P2gZ94ZT.SYdyveKmvNSQ#7>

**RAG**

I reviewed some research papers and came up with a technique that we can use for our chatbot which is RAG  
Retrieval augmented generation is a pipeline structure that enhances responses from a large language model. It does this by incorporating relevant data retrieved from a vector database, adding it as context to the prompt, and sending it to the LLM for generation.

**Retrieval-Augmented Generation (RAG)** is a technique that combines two AI capabilities:

1. **Retrieval**: Finding and pulling in relevant information from a large collection of data (like searching a database).
2. **Generation**: Creating new content based on the retrieved information.

**Pipeline**

Document loaders 🡪 Text splitting (chunking) 🡪 embedding conversion 🡪 vector storage 🡪 Output Generation using LLMs

**Technologies that we can use for this task are:**

* **Pinecone, chromadb:** for vector storage
* **HuggingFace embedding:** Word embedding
* **Not decided any kind of llm as of now:** LLM for output generation
* **Langchain:** for document loader and LLM Integration
* **Langsmith:** For monitoring chatbot responses and LLM evaluation