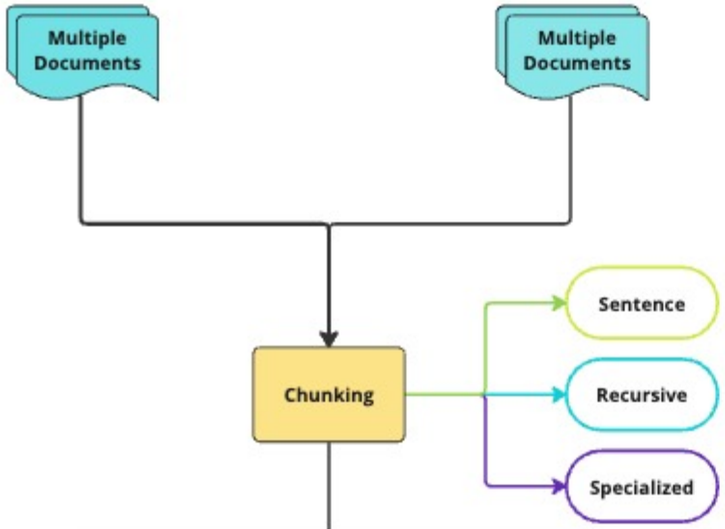
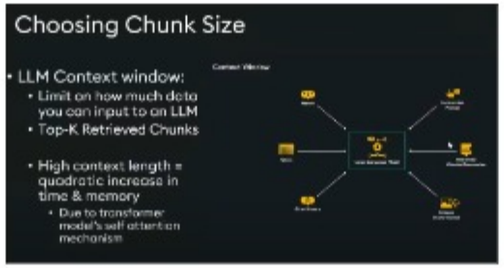
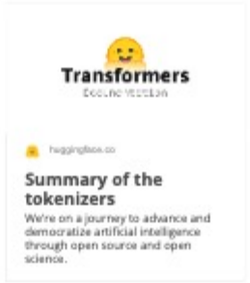
Stage 1 : Data Preparation





Chunking Best Practices for RAG Applications

LangChain Data Loaders, Tokenizers, Chunking, and...



Stage 2 : Embedding

Sentence Transformers

Embedding-1

Sentence Transformers

Embedding-n

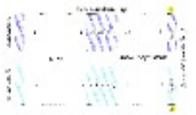
Architecture

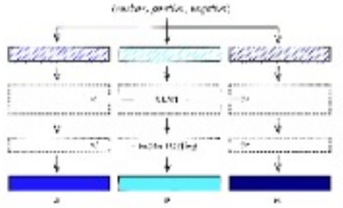
Training

Inference

Evaluation

Deploy

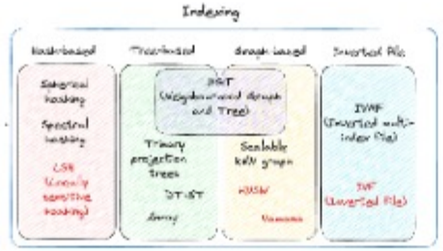




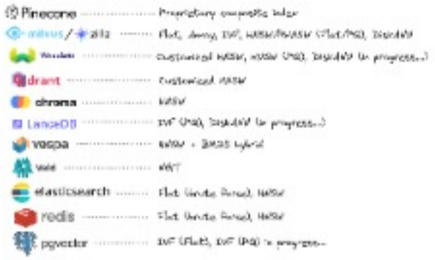
Stage 3 : Add / Update Record in Vector DB

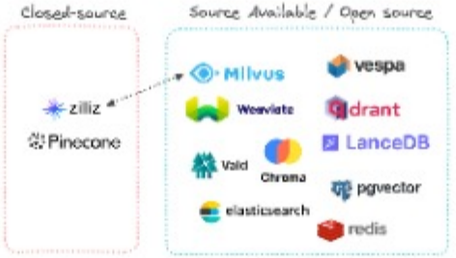
Similarity

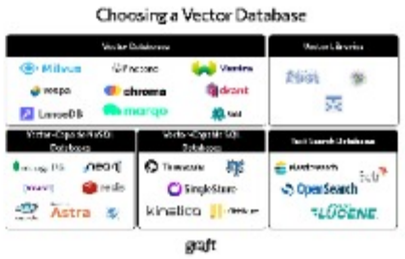
Indexing



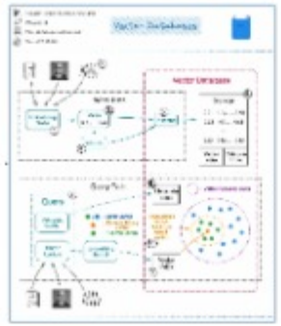
Vector DB



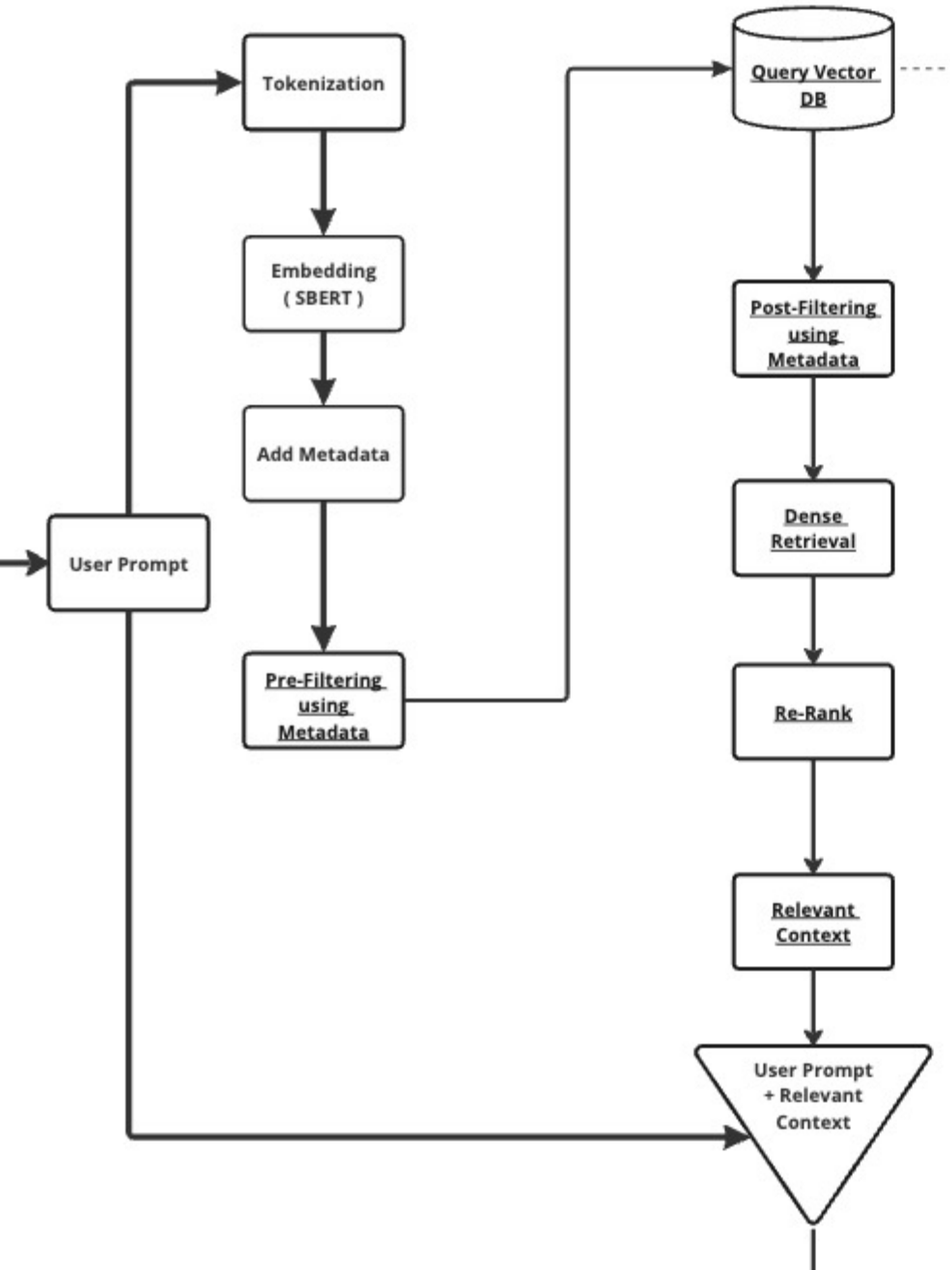


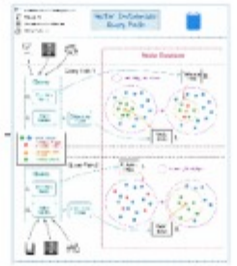


Write to Vector DB



Stage 4 : Query Processing + Semantic Search

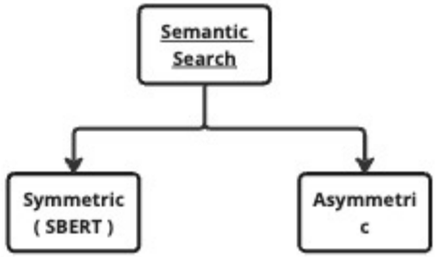




www.sbert.net

Semantic Search - Sentence-Transformers documentation

Semantic search seeks to improve search accuracy by understanding the content of the search query. In contrast to traditional search engines which only find documents based on lexical matches, semantic search can also find synonyms. The idea behind sema...



util.semantic_search

Instead of implementing semantic search by yourself, you can use the util.semantic_search function.

The function accepts the following parameters:

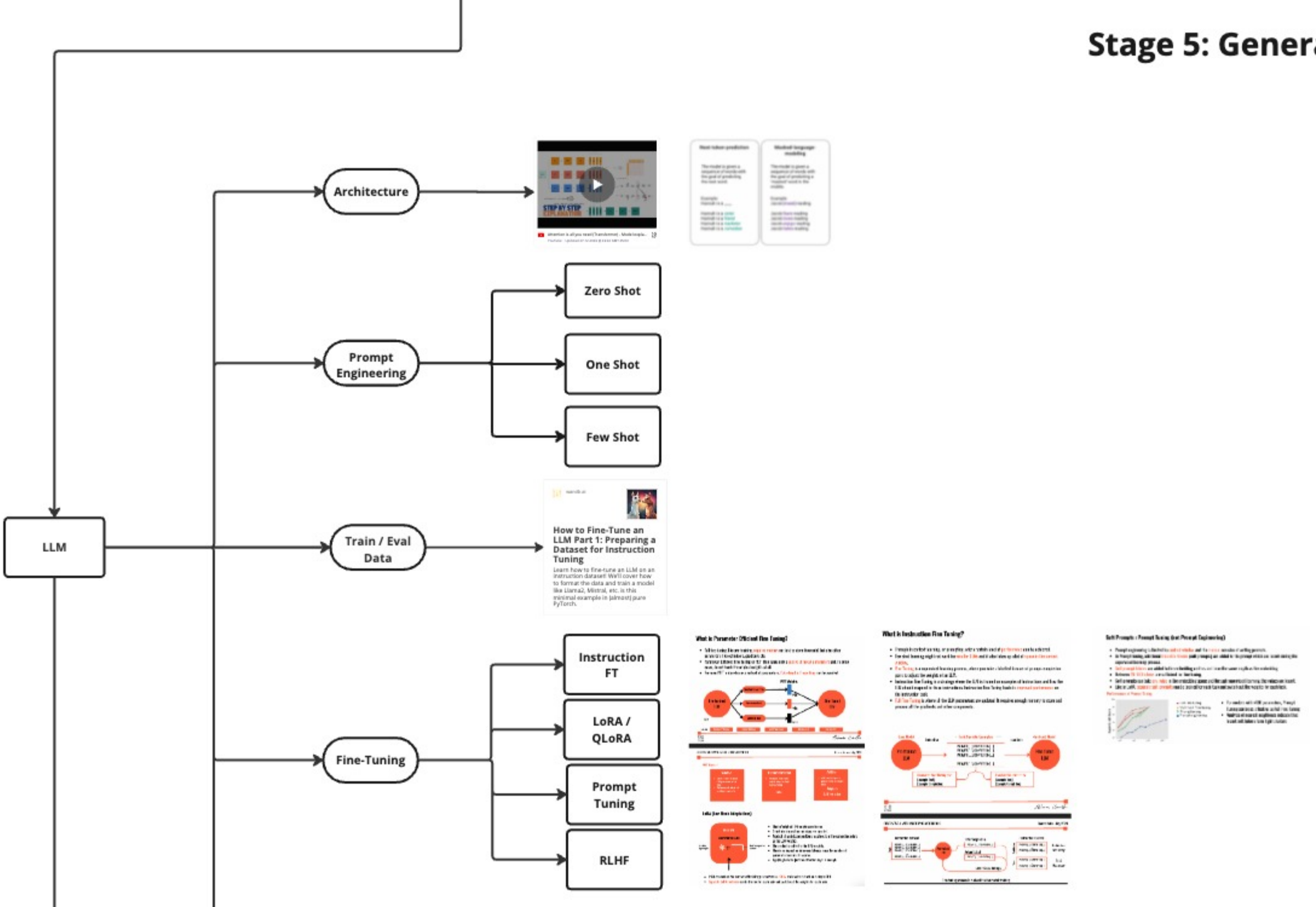
```
sentence_transformers.util.semantic_search(query_embeddings: torch.Tensor, corpus_embeddings: torch.Tensor, query_chunk_size: int = 100, corpus_chunk_size: int = 500000, top_k: int = 10, score_function: typing.Callable[[torch.Tensor, torch.Tensor], torch.Tensor] = <function cos_sim>)
```

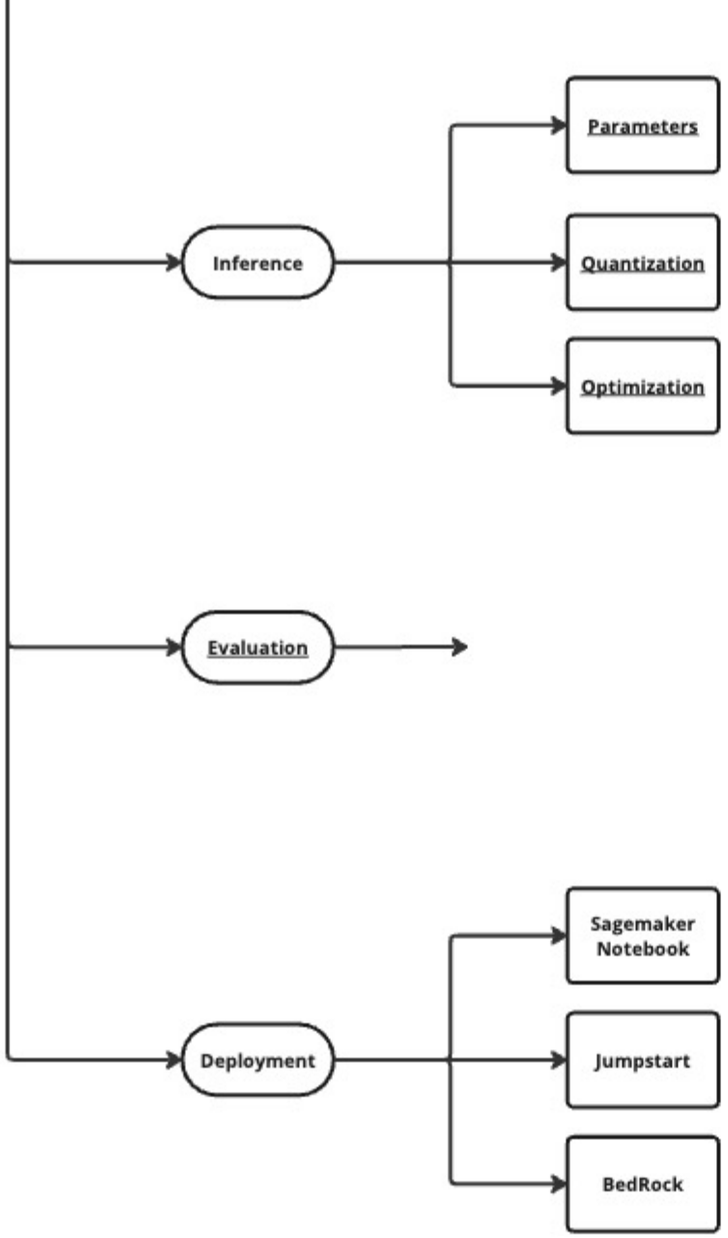
This function performs a cosine similarity search between a list of query embeddings and a list of corpus embeddings. It can be used for Information Retrieval / Semantic Search for corpora up to about 1 Million entries.

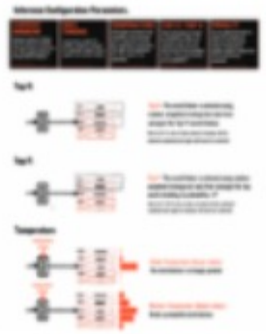
Parameters:

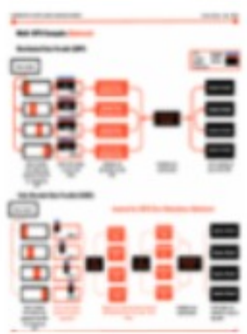
- **query_embeddings** – A 2 dimensional tensor with the query embeddings.
- **corpus_embeddings** – A 2 dimensional tensor with the corpus embeddings.
- **query_chunk_size** – Process 100 queries simultaneously. Increasing that value increases the speed, but requires more memory.
- **corpus_chunk_size** – Scans the corpus 100k entries at a time. Increasing that value increases the speed, but requires more memory.
- **top_k** – Retrieve top k matching entries.
- **score_function** – Function for computing scores. By default, cosine similarity.

Stage 5: Generation LLM



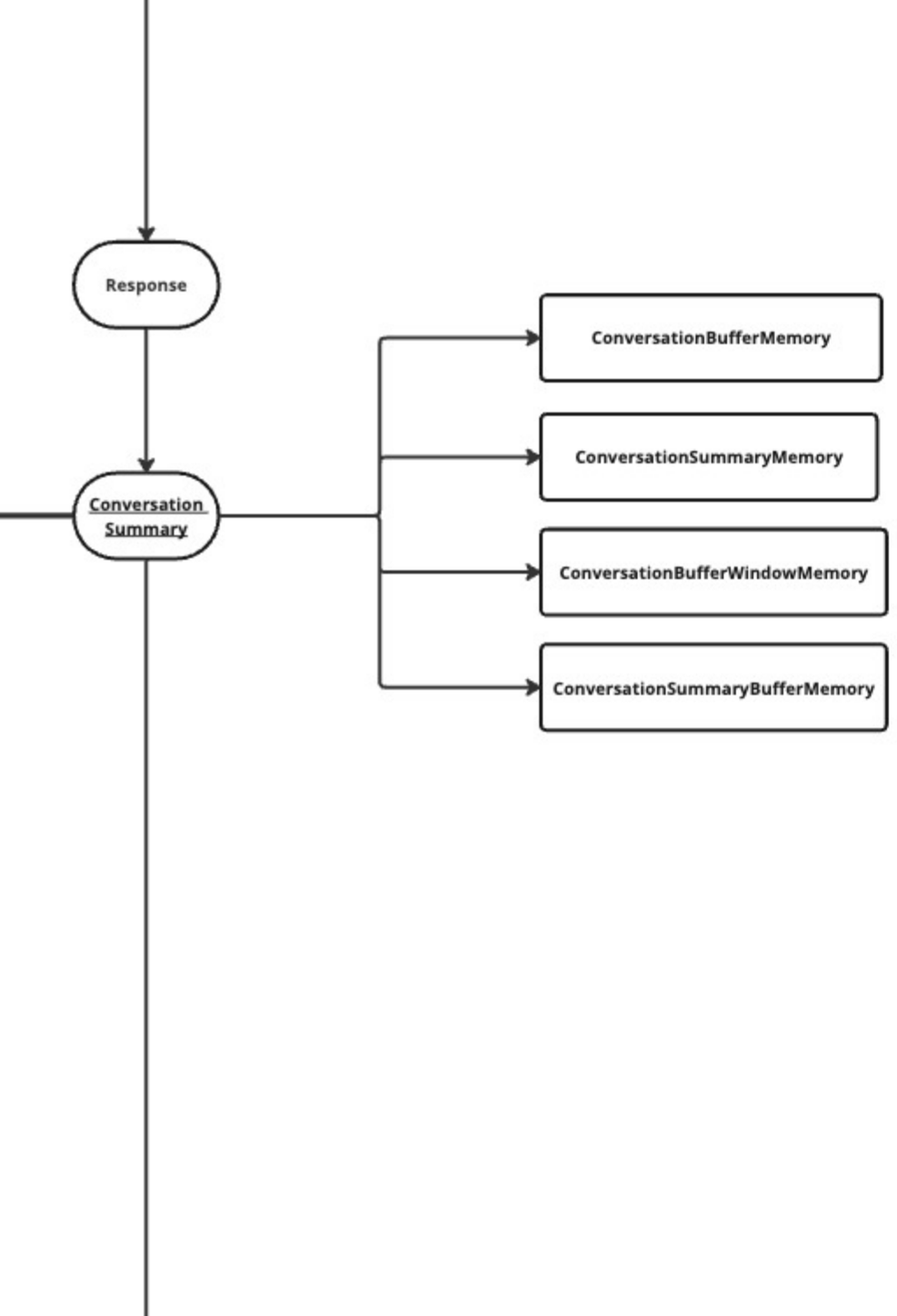






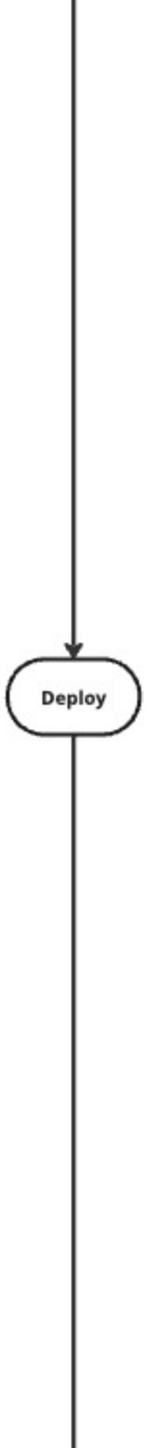
Stage 5: Generation LLM

Stage 6: Conversation Summary

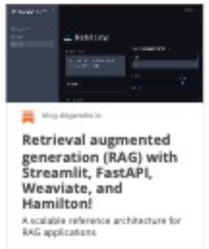


Stage 7: Evaluation

Stage 8 : Deployment



Stage 10: Front-End



Retrieval augmented generation (RAG) with Streamlit, FastAPI, Weaviate, and Hamilton!

A scalable reference architecture for RAG applications

UI