**Q&A Chatbot**

**Chunking technique used = Semantic Chunking used in this Q&A Chatbot**

Semantic chunking is the process of breaking down large text documents into smaller, meaningful parts (chunks) based on their content rather than just word or character limits. Instead of cutting text midway, it ensures each chunk contains a complete and relevant idea. This helps in maintaining context and improving retrieval Retrieval-Augmented Generation (RAG).

Here in semantic chunking we will split text to series of sentences. Then for each consecutive sentences like sen 1, sen 2 sen3 … we will check cosine similarity to next sentence, ex sen 1 with sen 2 , sen 2 with sen 3 and so on. Whenever cosine similarity score less than expected threshold will merge previous all sentence as one chunk and and next chunk will be start from next sentences.

**Why is Semantic Chunking Helpful?**

1. **Better Search Results** – It improves information retrieval by ensuring relevant sections are retrieved rather than incomplete fragments.
2. **Improved AI Understanding** – AI models can better process and generate responses when they receive complete, coherent chunks of text.
3. **Efficient Storage & Processing** – By storing text in structured chunks, retrieval and indexing become more efficient.
4. **Enhanced Context Retention** – Preserving the logical flow of information prevents disjointed or misleading AI responses.
5. **More Accurate Responses** – It helps reduce irrelevant or incorrect answers by focusing retrieval on meaningful text units.

**Further improvement on retrieval**

* Implementing reranker to re rank documents based on how relevant document to user query
* Implementing agentic retrieval to help in query rewriting, query transformation to get better retrieved documents from vectors tore

**Best and fast ways to parse the documents**

* Selecting best parser for document to parse the document, ex PyMuPDF for PDFs, python-docx for DOCX, pandas for Excel, and BeautifulSoup for HTML. For large documents, chunk text using semantic methods (e.g., LangChain's RecursiveCharacterTextSplitter) or fixed-size chunks.
* Leverage **multiprocessing** or **asyncio** for faster parsing