**Summary of work to implement RAG based LLM**

* **Scraping**
  + **Approach**

Wrote a script to crawl the Nvidia SDK and Toolkits till a depth of 4 levels. I had to curate a list of URLs to do this since I had multiple failed attempts at scraping the whole Nvidia docs site. I tried using libraries like SiteMap crawler too but it was just too much to crawl from my machine. Once I curated a list for the sdk and tools I did the same exercise for blogs and forums. Tried different depths too but settled on 4 as comprehensive enough but not too heavy to scrape

* + **Result**

I have a decent dataset of documentation, blogs and forums for training but it can be made more comprehensive.

* **Model selection**
  + **Approach**

I tried 2 LLM’s models, one was an Alpaca model ('chavinlo/alpaca-native') and the other was a Llama model ('meta-llama/Llama-2-7b-chat-hf'). The alpaca model was more consistent in generating responses that were useful and had good summarization with minimal hallucinations but it had a limit of 2046 tokens as an input to the model and since our token chunk size for appropriate context was found to be 2000 this would fail for some queries.

* + **Result**

Finally used the llama model (4096 token limit ) that hallucinated a lot initially but with proper prompting instructions and a more refined retrieval process to provide it with the right context to generate responses, it works decently well.

* **Vector Store**
  + **Approach**

Tried to use FAISS and ChromaDb to store the vectorized document chunks. Found FAISS to be better at search and retrieval. Initially I was working with cosine similarity for retrieval. I tried both with and without a threshold. The semantic search was good but there were a lot of duplicate chunks that were being retrieved and that would make the LLm responses inconsistent based on which chunk it used in a context. Moved to MMR ( Maximal Marginal Relevance ) and found better results since the documents were semantically similar but diverse

* + **Result**

Using MMR with FAISS retrieval and a restriction of just 3 context responses.

* **Prompt**
  + **Approach**

Tried different ways to prompt the LLM to return accurate results and not hallucinate if it didn’t know the answer or no context was given to it. There was a lot of trial and error to get this right

* + **Result**

Finally found a prompt that works well to instruct the LLM to give the right kind of responses.

* **Retrieval**
  + **Approach**

Tried a few retrieval techniques to improve the retrieved context results from the vectorstore.

Ensemble Retriever(Hybrid search) with FAISS and BM25 with different weights. BM25 being a keyword search algorithm and FAISS for semantic search. The combination worked decently well but I did get a lot of mixed results in responses for queries from the LLM

Self Querying Retriever worked great but for some reason the metadata search was not comprehensive to the kind of documents we had and would miss the relevant documents. Better structuring of the metadata would help it but the size of our input documents didn’t make it a feasible option to tinkler with.

MultiQuerying Retriever worked the best and would have been my goto solution but required more GPU RAM and was very slow in response time because of the fact that it creates prompts of its own to get more relevant information and this builds up in time to get context from FAISS or the ensemble.

* + **Result**

Didn’t end up using any retriever besides the FAISS with MMR search

* **Post-Retrieval**
  + **Approach**

The results returned from the FAISS retrieval were good but the LLm responses were unpredictable and sometimes would just go off. I went with an approach of doing some post retrieval processing by using a reranker. The goal of the reranker was to use a light weight LLM to score the context that has to most relevance to the query and then return a subset of what its got to the LLM to generate a response.

* + **Result**

This was by far the most consistent in generating similar responses on multiple tries and the results were accurate too