By Ethan Wang

Goal

- Build a Customer Service chat bot that
- ☐ use our company's data
- ☐ run locally

Obervations

- huggingface indeed have free model but
  - Quality of response is not good enough. (cn-tw)
  - Only support text generation not QA over docs.
- Fine-tuning the original pre-trained LLM is not the solution.

Challenges and Problems to solve

- ☐ Want an LLM run locally so we don't have to send our data to OpenAI or any other external companies' LLM.
- GPT4All model runs slowly locally.
- Again, quality of response is not good enough
- Evaluation?

Solutions

QA over documents

https://github.com/yousenwang/information-retrieval/blob/main/YT_HF_Instructor_Embeddings_Chroma_DB_Multi_Doc_Retriever_LangChain_Part2.ipynb

QA over Documents Steps:

1. Load our knowledge.
    1. PyPDFLoader for .pdf
    2. TextLoader for .txt
2. Split doc into text chunks.

```
from lang.text_splitter import RecursiveCharacterTextSplitter
```

3. Download Embeddings.
    1. HF Instructor Embeddings (e.g. hkunlp/instructor-xl)
    2. LlamaCppEmbedding (e.g. ggml-model-q4_0.bin)

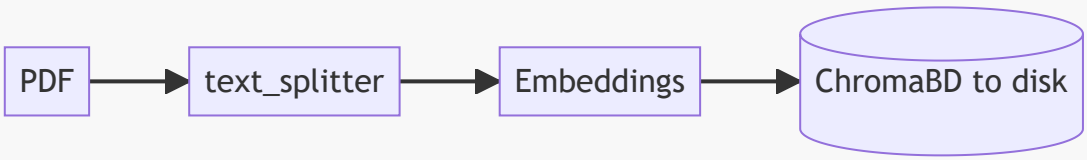
4. Save ChromaBD to disk.
5. Load Retriever 檢索器 from disk.
6. Create an LLM.
    1. OpenAI
    2. GPT4All (e.g. gpt4all-converted.bin)

```
from langchain.llms import GPT4All
```

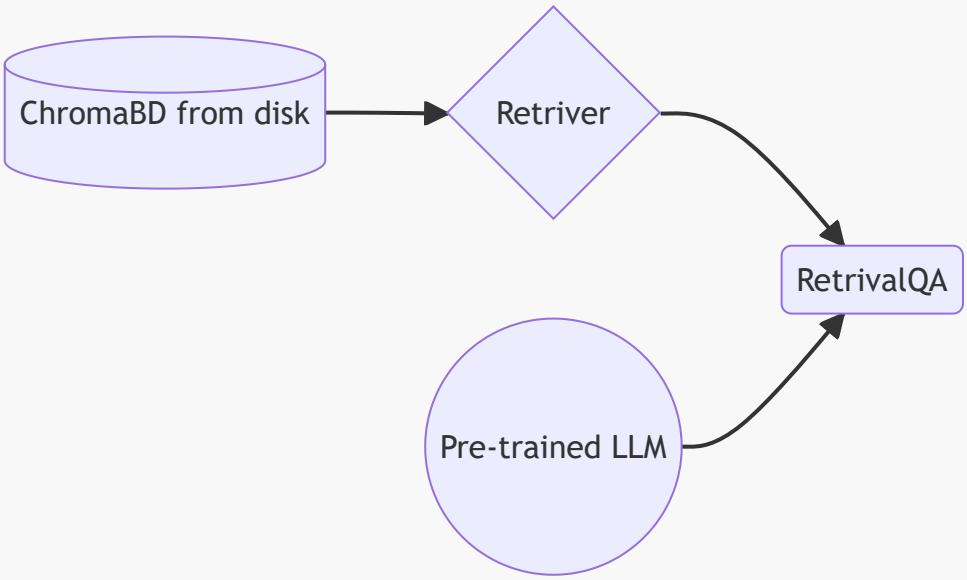
7. Use Chain (RetrievalQA) to feed Question and Retriever's answers to LLM.

Langchain Flowchart

Step 1. Establish our own `speicalized knowledge database` locally.



Step 2. Put 檢索器 and Large Lang Model (LLM) into the Retrival QA object.



Step 3. Based on `the input question`, the retriver finds the top `K` most relevant `text chuncks' embeddings` from `ChromaBD` saved in local disk.

Chart A

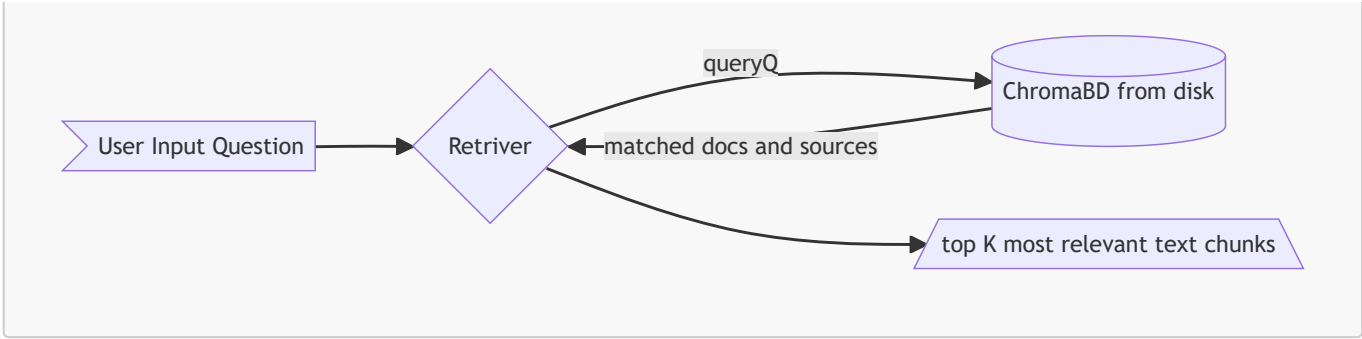
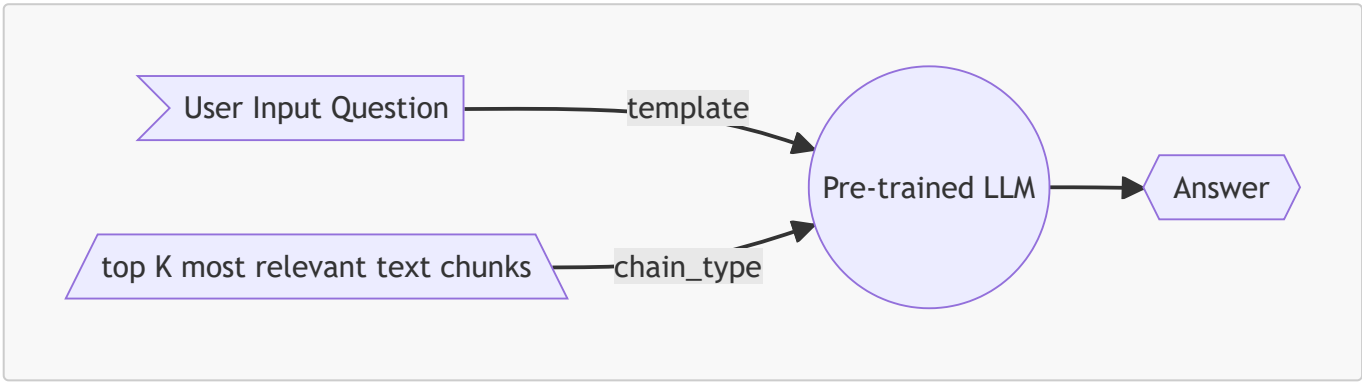


Chart B



Embeddings

```
from langchain.embeddings import HuggingFaceInstructEmbeddings

instructor_embeddings =
HuggingFaceInstructEmbeddings(model_name="hkunlp/instructor-xl",
                                                    model_kwargs={"device":
"cuda"})
```

```
Downloading (…)7f436/.gitattributes: 100%
1.48k/1.48k [00:00<00:00, 91.2kB/s]
Downloading (…)_Pooling/config.json: 100%
270/270 [00:00<00:00, 20.1kB/s]
Downloading (…)/2_Dense/config.json: 100%
116/116 [00:00<00:00, 5.68kB/s]
Downloading pytorch_model.bin: 100%
3.15M/3.15M [00:00<00:00, 43.4MB/s]
Downloading (…)0daf57f436/README.md: 100%
66.3k/66.3k [00:00<00:00, 2.85MB/s]
Downloading (…)af57f436/config.json: 100%
1.52k/1.52k [00:00<00:00, 71.6kB/s]
Downloading (…)ce_transformers.json: 100%
122/122 [00:00<00:00, 2.03kB/s]
Downloading pytorch_model.bin: 100%
4.96G/4.96G [00:23<00:00, 277MB/s]
Downloading (…)nce_bert_config.json: 100%
53.0/53.0 [00:00<00:00, 3.89kB/s]
```

```
Downloading (…)cial_tokens_map.json: 100%
2.20k/2.20k [00:00<00:00, 136kB/s]
Downloading spiece.model: 100%
792k/792k [00:00<00:00, 50.9MB/s]
Downloading (…)7f436/tokenizer.json: 100%
2.42M/2.42M [00:00<00:00, 9.84MB/s]
Downloading (…)okenizer_config.json: 100%
2.40k/2.40k [00:00<00:00, 162kB/s]
Downloading (…)f57f436/modules.json: 100%
461/461 [00:00<00:00, 26.1kB/s]
load INSTRUCTOR_Transformer
max_seq_length  512
```

LLM

```
# Retrieve model
gptj = gpt4all.GPT4All("ggml-gpt4all-j-v1.3-groovy.bin")
```

```
100%|██████████| 3.79G/3.79G [01:25<00:00, 44.3MiB/s]
Model downloaded at:  /root/.cache/gpt4all/ggml-gpt4all-j-v1.3-groovy.bin
```

Reference

https://huggingface.co/ckip-joint/bloom-1b1-zh

https://oalieno.tw/2023/03/02/bloom-zh/