**CHAPTER 7:**

**ingest.py**

*import os*

*import click*

*from typing import List*

*from langchain.document\_loaders import PyPDFLoader*

*from langchain.vectorstores import FAISS*

*from langchain.text\_splitter import RecursiveCharacterTextSplitter*

*from langchain.docstore.document import Document*

*from langchain.embeddings import HuggingFaceInstructEmbeddings*

*#function to load pdf*

*def load\_documents():*

*# To Load the PDF document from source documents directory*

*loader = PyPDFLoader('SOURCE\_DOCUMENTS/Attention.pdf')*

*docs = loader.load()*

*return docs*

*@click.command()*

*@click.option('--device\_type', default='cuda', help='select gpu or cpu for execution')*

*def main(device\_type, ):*

*if device\_type in ['cpu', 'CPU']:*

*device='cpu'*

*else:*

*device=’cuda’*

*#  To load the documents and split it into chunks*

*print(f"Loading documents from Source Directory”)*

*documents = load\_documents()*

*textsplitter = RecursiveCharacterTextSplitter (chunk\_size=800, chunk\_overlap=120)*

*texts = textsplitter.split\_documents(documents)*

*print(f" loaded {len(documents)} documents from Source Directory")*

*print(f" split into {len(texts)} text chunks")*

*# Create embeddings*

*embeddings = HuggingFaceInstructEmbeddings(model\_name="hkunlp/instructor-base",*

*model\_kwargs={"device": device})*

*db = FAISS.from\_documents(texts,embeddings)*

*db.save\_local('faiss\_index')*

*if \_\_name\_\_ == "\_\_main\_\_":*

*main()*

**run\_llm.py**

*import torch, click*

*from langchain.vectorstores import FAISS*

*from langchain.embeddings import HuggingFaceInstructEmbeddings*

*from langchain.llms import HuggingFacePipeline*

*from langchain.chains import RetrievalQA*

*from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline*

*def load\_model(device):*

*""" Model can be selected from huggingface. It will download the model for first execution.*

*It will use the model from the disk for next iteration of runs*

*"""*

*model = ‘tiiuae/falcon-7b-instruct’*

*if device == "cuda":*

*tokenizer = AutoTokenizer.from\_pretrained(model)*

*else: # cpu will be used*

*tokenizer=AutoTokenizer.from\_pretrained(model)*

*model=AutoModelForCausalLM.from\_pretrained(model, trust\_remote\_code=True)*

*Pipe = pipeline(‘text-generation’, tokenizer=tokenizer, model=model, torch\_dtype=torch.float32 if device =="cpu" else torch.bfloat16, device\_map=device if device =="cpu" else "auto", max\_length=2048, temperature=0, top\_p=0.90, top\_k=10, repetition\_penalty=1.15,num\_return\_sequences=1, pad\_token\_id= tokenizer.eos\_token\_id )*

*local\_llm = HuggingFacePipeline(pipeline=Pipe)*

*return local\_llm*

*@click.command()*

*@click.option('--device\_type', default='cuda', help='select gpu or cpu for execution')*

*def main(device\_type, ):*

*# load the instructorEmbeddings*

*if device\_type in ['cpu', 'CPU']: device='cpu'*

*else: device='cuda'*

*print(f"Running on: {device}")*

*embeddings = HuggingFaceInstructEmbeddings(model\_name="hkunlp/instructor-base", model\_kwargs={"device": device})*

*# load the vectorstore from disk which was saved earlier*

*database = FAISS.load\_local('faiss\_index',embeddings)*

*retriever = database.as\_retriever()*

*# load the LLM for returning the responses to the questions asked*

*llm = load\_model(device)*

*query = RetrievalQA.from\_chain\_type(llm=llm, retriever=retriever, chain\_type="stuff", return\_source\_documents=True)*

*while True:*

*query = input("\nEnter the query: ")*

*if query == "exit":*

*break*

*# Get the answer from the question & answer chain*

*result = query(query)*

*answer, docs = result['result'], result['source\_documents']*

*# Print the result*

*print("\n\n> Question:")*

*print(query)*

*print("\n> Answer:")*

*print(answer)*

*# Print the relevant sources which was used for answering*

*print("----------------------------------Source---------------------------")*

*for document in docs:*

*print("\n> " + document.metadata["source"] + ":")*

*print(document.page\_content)*

*print("----------------------------------Source---------------------------")*

*if \_\_name\_\_ == "\_\_main\_\_":*

*main()*

**CHAPTER 8: NO CODES**

**CHAPTER 9: NO CODES**

**CHAPTER 10: NO CODES**