```
!pip install -Uq transformers accelerate einops bitsandbytes chromadb langchain pymupdf pytz gradio
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.8/8.8 MB 25.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 297.4/297.4 kB 12.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.6/44.6 kB 3.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.2/102.2 MB 5.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 525.5/525.5 kB 18.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 814.5/814.5 kB 7.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.9/3.9 MB 18.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 505.5/505.5 kB 16.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 33.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 24.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 91.9/91.9 kB 5.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.8/60.8 kB 5.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.3/41.3 kB 3.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.4/5.4 MB 27.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.8/6.8 MB 41.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.1/60.1 kB 3.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.1/106.1 kB 9.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.3/67.3 kB 4.6 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 698.9/698.9 kB 44.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 49.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.6/67.6 kB 8.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 144.8/144.8 kB 18.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 83.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 278.4/278.4 kB 34.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 104.2/104.2 kB 13.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 30.8/30.8 MB 36.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.6/313.6 kB 34.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 54.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.9/129.9 kB 16.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.4/49.4 kB 5.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.9/71.9 kB 9.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 10.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 7.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.0/53.0 kB 6.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 6.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.8/50.8 kB 7.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 341.4/341.4 kB 38.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.4/3.4 MB 61.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 80.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 11.9 MB/s eta 0:00:00
  Building wheel for pypika (pyproject.toml) ... done
  Building wheel for ffmpy (setup.py) ... done
```

```python
import json
import gradio

import transformers
from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline

import torch
from torch import cuda, bfloat16
from langchain.text_splitter import RecursiveCharacterTextSplitter

import chromadb
from chromadb.utils import embedding_functions

from datetime import datetime
import json
import fitz
import pytz

import os

# if using Google Colab uncomment the following lines
# from google.colab import drive

# drive.mount("/content/drive")
# os.chdir(r"/content/drive/My Drive/AAI courses/TA Chatbot") # Change to desired dir
```

```
Mounted at /content/drive
```

Set up the database

```python
CHROMA_DATA_PATH = "chromadb_database"


def fetch_database(hf_auth=None, ef_name="sentence-transformers/all-mpnet-base-v2"):
    if hf_auth:
        ef = embedding_functions.HuggingFaceEmbeddingFunction(
            api_key=hf_auth, model_name=ef_name
        )
    else:
        ef = (
            embedding_functions.DefaultEmbeddingFunction()
        )  # use the default all-MiniLM-L6-v2

    client = chromadb.PersistentClient(path=CHROMA_DATA_PATH)
    collection = client.get_or_create_collection(
        name="course_material", embedding_function=ef
    )
    return client, collection


# with open("creds.txt", "r") as file: # load your hugging face token
#     hf_auth = json.load(file)["hf_token"]

client, collection = fetch_database()
```

```python
def load_material(collection=collection, chunk_size=250, chunk_overlap=20):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size, chunk_overlap=chunk_overlap
    )
    if collection.count():  # if the store is already built, abort
        return
    ids = 0
    data_path = "data"

    for file in os.listdir(data_path):
        docs = []
        if file.startswith(
            "module"
        ):  # excpecting data poinoning folders to have the name module#
            week_num = file[6]
        else:
            week_num = 0  # means available at all time

        try:
            curr_file = fitz.open(os.path.join(data_path, file))
            for i in range(len(curr_file)):
                if curr_file[i].get_text():
                    for doc in text_splitter.split_text(curr_file[i].get_text()):
                        docs.append(doc)
            idx = list(f"{i}" for i in range(ids, ids + len(docs)))
            ids += len(docs)
            meta_data = [{"week": f"{week_num}"}] * len(idx)

            collection.add(documents=docs, metadatas=meta_data, ids=idx)

        except Exception as e:
            print("Could not process ", file, "\nError: ", e, sep="")


load_material()
```

```python
# set the end date for each week
dates = [
    datetime.strptime("2024-04-09", "%Y-%m-%d").date(),  # beginning of week 1
    datetime.strptime("2024-04-10", "%Y-%m-%d").date(),  # end of week 1
    datetime.strptime("2024-04-11", "%Y-%m-%d").date(),  # end of week 2
    datetime.strptime("2024-04-12", "%Y-%m-%d").date(),  # end of week 3...
    datetime.strptime("2024-04-13", "%Y-%m-%d").date(),
    datetime.strptime("2024-04-14", "%Y-%m-%d").date(),
    datetime.strptime("2024-04-15", "%Y-%m-%d").date(),
]


def query_db(
    query, metadata_filters, n_results=1
):  # only fetch the most similar document
    results = collection.query(
        query_texts=query,
        n_results=n_results,
        where=metadata_filters,
    )
    return results


tz = pytz.timezone("America/Los_Angeles")


def fetch_docs(query, n_docs=3, tz=tz):
    now = datetime.now(tz).date()

    for i in range(len(dates)):
        if now <= dates[i]:
            metadata_filters = {"week": {"$in": ["0", str(i + 1)]}}

            return "\n\n".join(
                query_db(query, metadata_filters, n_docs)["documents"][0]
            )
```

Load the LLM

```
def load_model(hf_auth, model_id, temperature=0.001):
    # Determine the device (GPU if available, else CPU)
    device = f"cuda:{cuda.current_device()}" if cuda.is_available() else "cpu"

    # Configure quantization settings for loading the model with less GPU memory usage
    bnb_config = transformers.BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_use_double_quant=True,
        bnb_4bit_compute_dtype=bfloat16,
    )

    # Load the configuration for the pre-trained model
    model_config = transformers.AutoConfig.from_pretrained(model_id, token=hf_auth)

    tokenizer = AutoTokenizer.from_pretrained(model_id, token=hf_auth)

    # Load the model for causal language modeling
    model = transformers.AutoModelForCausalLM.from_pretrained(
        model_id,
        trust_remote_code=True,
        config=model_config,
        quantization_config=bnb_config,
        device_map="auto",
        token=hf_auth,
    )

    # Set the model in evaluation mode for inference
    model.eval()

    pipe = pipeline(
        task="text-generation",
        model=model,
        tokenizer=tokenizer,
        temperature=temperature,
        # top_k=40,
        eos_token_id=tokenizer.eos_token_id,
        # repetition_penalty=1.5,
        return_full_text=False,
        # do_sample=True,
    )

    # Print device information where the model is loaded
    print(f"Model loaded on {device}")

    return pipe


with open("creds.txt", "r") as file:
    hf_auth = json.load(file)["hf_token"]

pipe = load_model(hf_auth, "meta-llama/Llama-2-13b-chat-hf")
```

```
Loading checkpoint shards: 100%                                   2/2 [04:30<00:00, 121.50s/it]
Model loaded on cuda:0
```

We will use the following prompt template:

```
<s>[INST] <<SYS>>
{{ system_prompt }}
<</SYS>>

{{ user_msg_1 }} [/INST] {{ model_answer_1 }} </s><s>[INST] {{ user_msg_2 }} [/INST] {{ model_answer_2 }} </s><s>[INST] {{ user_msg_3 }} [/INST]
```

We will pass the context in the following structure:

```
Context information from multiple sources is below.
---------------------
{context_str}
---------------------
Given the information from multiple sources and not prior knowledge, answer the query.
Query: {query_str}
Answer:
```

```
SYS_PROMPT = """You are an expert teacher assistant for a course called ADS 500B. You answer questions relating to software engineering,
You are honest and helpful. You answer succinctly and professionally. You do not make up facts. Use the Context provided to answer the qu
If you're asked something you do not know the answer to, say you do not know. Do not make up facts. Be brief and to the point.\
"""

TEMPLATE = "<s>[INST] <<SYS>>\n{sys_prompt}\n<</SYS>>"

_SYS_PROMPT = TEMPLATE.format(sys_prompt=SYS_PROMPT)


def build_prompt(query_w_context: str, hist_len: int = 3):
    if not len(history):  # if first query
        return _SYS_PROMPT + f"\n\n{query_w_context} [/INST] "

    prompt = _SYS_PROMPT + "\n\n"

    for i in range(len(history)):
        if i == 0:
            prompt += f"{history[i]} [/INST] "
            continue

        if i % 2 == 0:
            prompt += f"<s>[INST]{history[i]} [/INST] "

        else:
            prompt += f"{history[i]} </s>"

    return prompt + f"<s>[INST]{query_w_context} [/INST] "


def query_model(query: str, hist_len: int = 3):
    global history

    context = fetch_docs(query=query, n_docs=1)

    query_w_context = f"""Context information from multiple sources is below.
---------------------
{context}
---------------------
Given the information from multiple sources and not prior knowledge, answer the query.
Query: {query}
Answer: """

    prompt = build_prompt(query_w_context, hist_len)
    response = pipe(prompt, max_new_tokens=1024)[0]["generated_text"]

    history.append(query)
    history.append(response)

    if len(history) > hist_len:
        history = history[
            -(hist_len * 2) :
        ]  # only keep hist_len interaction pair history.
    return response
```

```python
global history
history = []
hist_len = 2


def chatbot_interface(query, gradio_hist):
    gradio_hist = None
    result = query_model(query)
    return result


```

```python
interface.launch(share=True, debug=True)
```

```
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
Running on public URL: https://3a92f59170c09a1ddd.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces
```