```
!pip install spacy
!python -m spacy download en_core_web_sm
!pip install dateparser
!pip install parsedatetime
!pip install chromadb
!pip install sentence-transformers
!pip install transformers
!pip install langchain
!pip install -U langchain-community
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/py
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/python3.
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/pyth
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/pyth
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/l
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/pyth
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.1
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pytho
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=4.6.1 in /usr/local/lib/pyt
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.1
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/pyt
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.1
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/p
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/pyt
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: marisa-trie>=0.7.7 in /usr/local/lib/python3.1
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/pyth
Requirement already satisfied: wrapt in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-p
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en
```

```
                        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 27.4 MB/s eta 0:00
    Requirement already satisfied: spacy<3.8.0,>=3.7.2 in /usr/local/lib/python3.
    Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/
    Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/
    Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/py
    Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.
    Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python
    Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/python3.
    Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3
    Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.
    Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/pyth
    Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3
    Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.
    Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.
    Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/pyth
```

```python
import spacy
import dateparser
import parsedatetime
from datetime import datetime
from langchain.document_loaders import TextLoader
from sentence_transformers import SentenceTransformer, util
from transformers import AutoModelForCausalLM, AutoTokenizer
import chromadb
import re

# Load SpaCy model
nlp = spacy.load("en_core_web_sm")

# Function to extract dates using spaCy
def extract_dates(paragraph):
    doc = nlp(paragraph)
    return [ent.text for ent in doc.ents if ent.label_ == "DATE"]

# Function to convert date strings into exact date objects
def convert_dates(extracted_dates):
    cal = parsedatetime.Calendar()
    converted_dates = []
    for date_str in extracted_dates:
        parsed_date = dateparser.parse(date_str)
        if parsed_date is None:
            time_struct, parse_status = cal.parse(date_str)
            if parse_status == 1:
                parsed_date = datetime(*time_struct[:6])
            else:
                parsed_date = None
        converted_dates.append((date_str, parsed_date))
    return converted_dates

# Function to validate email and phone number
def validate_user_info(name, phone, email):
    phone_valid = re.fullmatch(r"\+?\d{10,15}", phone)
    email_valid = re.fullmatch(r"[^@]+@[^@]+\.[^@]+", email)
    return phone_valid is not None, email_valid is not None

# Function to collect user info conversationally
def collect_user_info():
    name = input("Please provide your name: ")
    phone = input("Please provide your phone number: ")
    email = input("Please provide your email: ")

    phone_valid, email_valid = validate_user_info(name, phone, email)
    if not phone_valid:
        return "Invalid phone number format. Please try again."
    if not email_valid:
        return "Invalid email format. Please try again."

    return f"Thank you, {name}. We will call you at {phone} or email you at {email}

# Function to load documents and create ChromaDB collection
def setup_document_collection(filepath):
    client = chromadb.Client()
    now = datetime.now()
    timestamp = now.strftime("%Y%m%d_%H%M%S")  # Format: YYYYMMDD_HHMMSS
    collection_name = f"document_embeddings_{timestamp}"
    collection = client.create_collection(collection_name)
    loader = TextLoader(filepath)
    documents = loader.load()

    embedding_model = SentenceTransformer('all-MiniLM-L6-v2')
    texts = [doc.page_content for doc in documents]
    embeddings = embedding_model.encode(texts)

    for i, (embedding, text) in enumerate(zip(embeddings, texts)):
        collection.add(
            ids=[f"doc_{i}"],
            documents=[text],
            embeddings=[embedding.tolist()],
            metadatas=[{"text": text}]
        )
    return collection

# Function to setup the TinyLLaMA model and tokenizer
def setup_tiny_llama():
    model = AutoModelForCausalLM.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.
    tokenizer = AutoTokenizer.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.0")
    return model, tokenizer

# Function to generate answers using the TinyLLaMA model
def generate_answer(prompt, model, tokenizer):
    inputs = tokenizer(prompt, return_tensors="pt")
    # outputs = model.generate(inputs.input_ids, max_new_tokens=150, do_sample=True
        # Generate response with a limit on max tokens and prevent excessive lengtl
    outputs = model.generate(
        inputs.input_ids,
        max_new_tokens=150,       # Control the length of the output
        do_sample=True,           # Sampling for variability
        num_return_sequences=1,   # Single output response
        temperature=0.7,           # Adjust temperature for more focused responses
        repetition_penalty=1.2   # Penalize repetitive responses
    )
    return tokenizer.decode(outputs[0], skip_special_tokens=True)

# Function to answer a query by fetching relevant document context and generating a
def answer_query(query, collection, model, tokenizer, embedding_model, history):
    # Update history with the new query
    history.append((query, ""))
    if len(history) > 5:
        history.pop(0)  # Keep only the latest 5 entries

    # Generate the query embedding
    query_embedding = embedding_model.encode([query])[0]
    results = collection.query(
        query_embeddings=[query_embedding.tolist()],
        n_results=1
    )

    if results['metadatas']:
        # Extract the relevant context text
        context_text = results['metadatas'][0][0]['text']

        # Generate the answer without including the prompt in the output
        generated_answer = generate_answer(context_text, model, tokenizer)

        # Calculate the similarity score
        generated_embedding = embedding_model.encode([generated_answer])[0]
        similarity_score = util.cos_sim(query_embedding, generated_embedding).item(
        print(f"Similarity Score: {similarity_score}")

        # Update the history with the generated answer
        history[-1] = (query, generated_answer)  # Update the latest history entry

        # Return answer if it meets the similarity threshold
        if similarity_score > 40:
            return generated_answer.strip()
        else:
            return "Content not found."
    else:
        return "Content not found."

# Function to book an appointment based on user input
def book_appointment(user_input):
    extracted_dates = extract_dates(user_input)
    converted_dates = convert_dates(extracted_dates)

    for original, converted in converted_dates:
        if converted:
            return f"Appointment successfully booked for {converted.strftime('%Y-%m

    return "Could not book the appointment. Please provide a valid date."

def chatbot_function(filepath):
    # Setting up document collection
    collection = setup_document_collection(filepath)
    tiny_llama_model, tiny_llama_tokenizer = setup_tiny_llama()
    embedding_model = SentenceTransformer('all-MiniLM-L6-v2')

    history = []  # Initialize the history for queries and responses

    # Interaction loop
    while True:
        query = input("You: ")
        if "call me" in query.lower():
            print(collect_user_info())
        elif "book appointment" in query.lower():
            appointment_message = book_appointment(query)
            print(appointment_message)
        else:
            answer = answer_query(query, collection, tiny_llama_model, tiny_llama_t
            print(f"Chatbot: {answer}")

# Example usage
chatbot_function('/content/Agriculture(1).txt')
```

```
/usr/local/lib/python3.10/dist-packages/sentence_transformers/cross_encoder/C
  from tqdm.autonotebook import tqdm, trange
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.
  warnings.warn(
You: call me
Please provide your name: testing
Please provide your phone number: 98
Please provide your email: testing@gmail.com
Invalid phone number format. Please try again.
You: call me
Please provide your name: testing
Please provide your phone number: 9878898878
Please provide your email: te
Invalid email format. Please try again.
You: call me
Please provide your name: testing
Please provide your phone number: 9878899667
Please provide your email: testing@gmail.com
Thank you, testing. We will call you at 9878899667 or email you at testing@gm
You: book appointment for Next Friday
```

```
You: book appointment for Next Friday
Appointment successfully booked for 2024-10-18.
You: book appointment for Sunday
Appointment successfully booked for 2024-10-13.
You: Explain about Agriculture.
Similarity Score: 61.16095781326294
Chatbot: Agriculture, one of the oldest human activities, is the foundation o
Historical Background
Agriculture dates back to the Neolithic Revolution, around 10,000 BC, when hu
The Green Revolution
The 20th century saw a dramatic transformation in agriculture with the advent
Modern Agricultural Practices
Modern agriculture has continued to evolve, embracing advanced technology, da
Crop Diversification and Sustainable Farming
Sustainability has become a central theme in agriculture as the world grapple
Challenges Facing Modern Agriculture
While modern agriculture has made remarkable progress in terms of food securi
Soil degradation - One of the most pressing problems affecting modern agricul
You: Explain about computer.
Similarity Score: 9.184428304433823
Chatbot: Content not found.
You:
```

Start coding or generate with AI.