CONTEXT HANDLING FOR FOLLOW - UPS

INTRODUCTION:

Handling the context for follow-up conversations allows the chatbot to handle multi-turn conversations. The chatbot is able to remember the important information from the previous messages and use it in the next message. The system holds minimal context information to make it simple and give accurate responses.

OBJECTIVE:

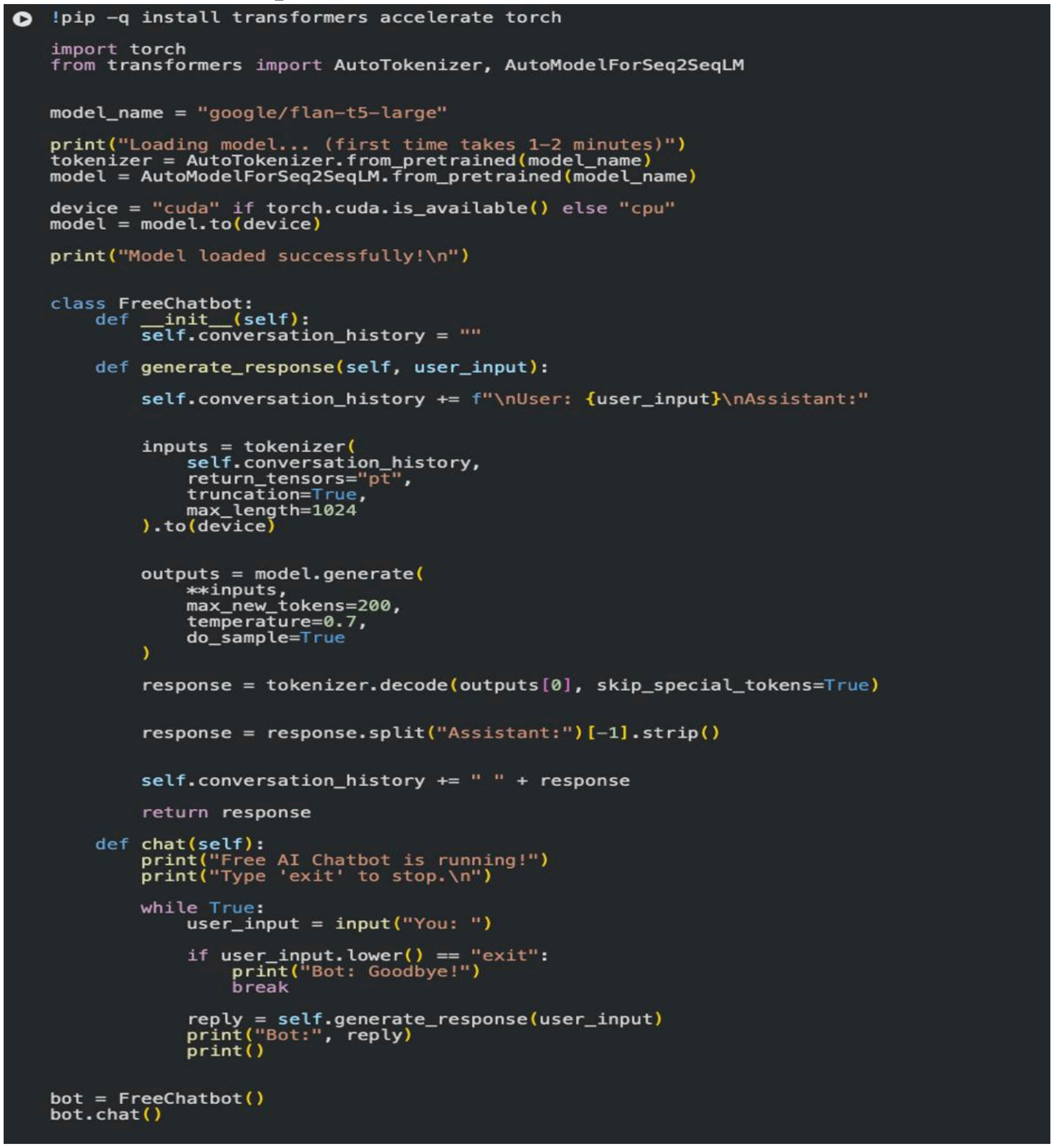
- Enable chatbot to support short multi-turn conversations.
- Maintain minimal conversation state.
- Store previously extracted entities.
- Merge previous and current entities.
- Improve natural interaction experience.
- Keep implementation simple and efficient.

TECHNOLOGY USED:

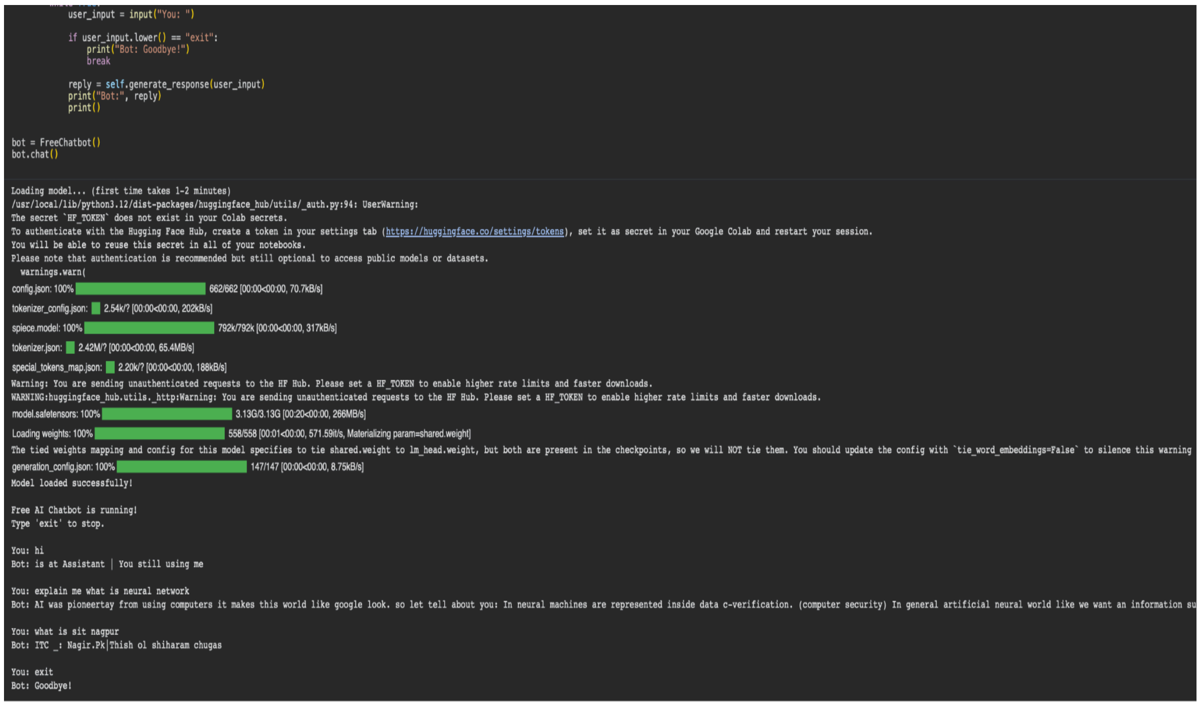
- Python – Used for implementing entity extraction and context handling logic.
- Natural Language Processing (NLP) – Used to process and understand user queries.
- Regular Expressions (Regex) – Used to extract semester numbers, course codes, dates, and exam-related keywords.
- Dictionary-Based Context Memory – Used to store previous entities and support short multi-turn conversations.

IMPLEMENTATION:

- Extract entities from query
- Update conversation state
- Fill missing entities using previous state
- Generate final response



```
!pip -q install transformers accelerate torch

import torch
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

model_name = "google/flan-t5-large"

print("Loading model... (first time takes 1-2 minutes)")
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

device = "cuda" if torch.cuda.is_available() else "cpu"
model = model.to(device)

print("Model loaded successfully!\n")


class FreeChatbot:
    def __init__(self):
        self.conversation_history = ""

    def generate_response(self, user_input):

        self.conversation_history += f"\nUser: {user_input}\nAssistant:"


        inputs = tokenizer(
            self.conversation_history,
            return_tensors="pt",
            truncation=True,
            max_length=1024
        ).to(device)


        outputs = model.generate(
            **inputs,
            max_new_tokens=200,
            temperature=0.7,
            do_sample=True
        )

        response = tokenizer.decode(outputs[0], skip_special_tokens=True)


        response = response.split("Assistant:")[-1].strip()


        self.conversation_history += " " + response

        return response

    def chat(self):
        print("Free AI Chatbot is running!")
        print("Type 'exit' to stop.\n")

        while True:
            user_input = input("You: ")

            if user_input.lower() == "exit":
                print("Bot: Goodbye!")
                break

            reply = self.generate_response(user_input)
            print("Bot:", reply)
            print()


bot = FreeChatbot()
bot.chat()
```



```
        user_input = input("You: ")

        if user_input.lower() == "exit":
            print("Bot: Goodbye!")
            break

        reply = self.generate_response(user_input)
        print("Bot:", reply)
        print()


bot = FreeChatbot()
bot.chat()
```

```
Loading model... (first time takes 1-2 minutes)
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100% 662/662 [00:00<00:00, 70.7kB/s]
tokenizer_config.json: 2.54k/? [00:00<00:00, 202kB/s]
spiece.model: 100% 792k/792k [00:00<00:00, 317kB/s]
tokenizer.json: 2.42M/? [00:00<00:00, 65.4MB/s]
special_tokens_map.json: 2.20k/? [00:00<00:00, 188kB/s]
Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to enable higher rate limits and faster downloads.
WARNING:huggingface_hub.utils._http:Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to enable higher rate limits and faster downloads.
model.safetensors: 100% 3.13G/3.13G [00:20<00:00, 266MB/s]
Loading weights: 100% 558/558 [00:01<00:00, 571.59it/s, Materializing param=shared.weight]
The tied weights mapping and config for this model specifies to tie shared.weight to lm_head.weight, but both are present in the checkpoints, so we will NOT tie them. You should update the config with `tie_word_embeddings=False` to silence this warning
generation_config.json: 100% 147/147 [00:00<00:00, 8.75kB/s]
Model loaded successfully!

Free AI Chatbot is running!
Type 'exit' to stop.

You: hi
Bot: is at Assistant | You still using me

You: explain me what is neural network
Bot: AI was pioneertay from using computers it makes this world like google look. so let tell about you: In neural machines are represented inside data c-verification. (computer security) In general artificial neural world like we want an information su

You: what is sit nagpur
Bot: ITC _: Nagir.Pk|Thish ol shiharam chugas

You: exit
Bot: Goodbye!
```

FLOWCHART:

- Start
  ↓
- User Input
  ↓
- Text Preprocessing
  ↓
- Entity Extraction
  ↓
- Context State Check
  ↓
- Update Memory & Generate Response
  ↓
- Display Output / End

OUTCOMES:

- Accurate Entity Extraction – The chatbot can identify semester, course, course code, exam type, and dates from user queries.
- Context-Aware Responses – Supports short multi-turn conversations, understanding follow-up questions.
- Improved User Interaction – Provides more natural and intelligent responses to academic queries.
- Structured Data Generation – Converts unstructured student queries into organized, actionable information.
- Scalable Architecture – Can be enhanced with NLP libraries, databases, or web deployment in the future.