Your First LLM Project

Your First LLM Project

We'll create a simple terminal interface to chat with an LLM.

To accomplish this, we need:
- API key for Gemini
- Method to call the LLM
- Data structure for chat history
- Control flow to manage the process

```
(llm) ritz@Roberts-MacBook-Air 02_first_llm_project %
```

Get Your Gemini API Key

Before building, get access to Google's Gemini 1.5 Flash and Pro.

Steps to get your API key:
- Sign into Google's AI Studio
- Agree to the legal notice
- Click **Get API Key** at the top left
- Create an API key in a new or existing project

As of now, billing is not required for a free tier API key.

Call the LLM

Once you have the API key, make your first call to Gemini.

Steps:
- Install the SDK from PyPI:
- Import the SDK and set your API key
- Select the model and call `generate_content`
- Print the results

```
pip install -U google-generativeai
```

Call the LLM

```python
import google.generativeai as genai
import os

genai.configure(api_key=os.environ["GEMINI_API_KEY"])
model = genai.GenerativeModel('gemini-1.5-flash')
response = model.generate_content("Who was the first President of Mongolia?")
print(response.text)
```

Storing the Chat History

Use a list of dictionaries to store chat history:

```python
model = genai.GenerativeModel("gemini-1.5-flash")
chat = model.start_chat(
    history=[
        {"role": "user", "parts": "Hello"},
        {"role": "model", "parts": "Great to meet you. What would you like to know?"},
    ]
)

response = chat.send_message("What is the tallest building in the world?")
print(response.text)
```

Gemini's Chat History Structure

- Each dictionary has `role` (user or model) and `parts` (content).
- The conversation is ordered from oldest to newest.
- Current user message is sent separately using `send_message`.

Gemini's chat history structure differs from OpenAI's, as the current message is not included with the previous chat messages.

Building the Control Flow

We'll create a terminal app to converse with the Gemini model, maintaining chat history.

Use a `while` loop for continuous conversation:

```
chat = model.start_chat()

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

    if user_input.lower() == "/bye":
        print("Exiting chat.")
        break

    response = chat.send_message(user_input)
    print("Gemini: " + response.text)
```

Control Chat History Manually

Manually controlling chat history can:
- Reduce the number of messages sent to the model
- Speed up replies and reduce API usage

Rebuild the chat with each new message by saving each message in a list of dictionaries.

Putting it All Together

```python
import google.generativeai as genai
import os
import sys

genai.configure(api_key=os.environ["GEMINI_API_KEY"])
model = genai.GenerativeModel('gemini-1.5-flash')
chat = model.start_chat()

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

    if user_input.lower() in ["/exit", "/quit", "/q", "/bye"]:
        print("Exiting chat.")
        break

    response = chat.send_message(user_input)
    print("Gemini: " + response.text)
```