Gemini Q&A Application: An Introduction to Generative AI

yash.s.halwai

Introduction

In this article, we'll walk through the creation of a simple yet powerful web application using Streamlit, Google Generative AI, and Python. This project will guide beginners into the field of generative AI, providing hands-on experience in building an interactive Q&A application. The application leverages Google's Gemini model to generate responses based on user input.

Project Title

Gemini Q&A Application: A Beginner's Guide to Generative AI

Prerequisites

Before diving into the code, ensure you have the following prerequisites installed:
- Python 3.7 or higher
- Streamlit
- google-generativeai library
- dotenv library

Additionally, you need a Google API key to access the Generative AI service.

Project Structure

1. Environment Configuration: Using `dotenv` to manage environment variables.

2. Streamlit Setup: Creating a web interface for user interaction.

3. Generative AI Integration: Configuring and utilizing Google's Generative AI model to generate responses.

4. User Interaction: Capturing user input and displaying the AI-generated response.

Code Explanation

Let's break down the code into sections and explain each part in detail.

1. Environment Configuration

```
from dotenv import load_dotenv

load_dotenv()   Load environment variables from .env.
```

Here, we use the `dotenv` library to load environment variables from a `.env` file. This is a secure way to manage sensitive information such as API keys.

2. Importing Required Libraries

```
import streamlit as st
import os
import pathlib
import textwrap

import google.generativeai as genai

from IPython.display import display
from IPython.display import Markdown
```

We import the necessary libraries:
- `streamlit` for building web applications.
- `os` for accessing environment variables.
- `pathlib` and `textwrap` for handling file paths and text formatting.
- `google.generativeai` for integrating Google's Generative AI.
- `IPython.display` for displaying markdown content.

3. Markdown Formatting Function

```
def to_markdown(text):
   text = text.replace('•', '  *')
   return Markdown(textwrap.indent(text, '> ', predicate=lambda _: True))
```

This function converts plain text to markdown format, indenting each line for better readability.

4. Configuring the API Key

```
os.getenv("GOOGLE_API_KEY")
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
```

We retrieve the Google API key from the environment variables and configure the Generative AI library with it.

5. Function to Get AI Response

```python
def get_gemini_response(question):
   model = genai.GenerativeModel('gemini-pro')
   response = model.generate_content(question)
   return response.text
```

This function interacts with the Gemini model. It takes a user question as input, generates a response using the model, and returns the generated text.

6. Streamlit Application Setup

```python
st.set_page_config(page_title="Q&A Demo")
st.header("Gemini Application")

input = st.text_input("Input: ", key="input")
```

```
submit = st.button("Ask the question")
```

We configure the Streamlit application with a title and header. Then, we create an input field for the user to enter their question and a button to submit it.

7. Handling User Interaction

```
if submit:
    response = get_gemini_response(input)
    st.subheader("The Response is")
    st.write(response)
```

When the user clicks the submit button, the application calls the `get_gemini_response` function with the user's input, retrieves the AI-generated response, and displays it on the web page.

Conclusion

By following this guide, you can create a simple yet powerful Q&A application using generative AI. This project serves as an excellent starting point for beginners to understand the basics of integrating AI models into web applications.

Summary of Steps

1. Set up environment variables: Securely manage sensitive information.
2. Import necessary libraries: Streamlit for the web interface, Google's generative AI for the model, and utility libraries for text formatting and display.
3. Configure the AI model: Use the API key to access Google's Gemini model.
4. Create the Streamlit interface: Set up user input fields and buttons.
5. Generate and display responses: Handle user interactions, generate AI responses, and display them on the web interface.

Additional Resources

For further learning, consider exploring the following:
- Streamlit Documentation(https://docs.streamlit.io/)
- Google Generative AI Documentation(https://cloud.google.com/generative-ai)

- Python Official Documentation(https://docs.python.org/3/)
- YouTube Video(https://youtu.be/CC6qMpqgUMU?si=mQJ4ChWXEJpuImoz)

By building this application, you take your first step into the exciting world of generative AI, opening up numerous possibilities for future projects and innovations.

Connect With Me:

- **LinkedIn**
- **GitHub**
- **YouTube**