

CREATE A CHATBOT IN PYTHON

Phase 4: Development Part 2

In this part you will continue building your project.

Continue building the chatbot by integrating it into a web app using Flask.

Dataset

Link: <https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot>

Building a chatbot and integrating it into a web app using Flask is a great way to provide interactive and personalized experiences to your users. Here's a step-by-step guide on how to do this:

Set Up Your Development Environment: Make sure you have Python installed on your system. You'll also need Flask and any necessary dependencies, such as a natural language processing library like NLTK or spaCy if your chatbot needs to process text.

Create a Flask Project: Start a new Flask project by creating a directory and setting up a basic Flask application.

Create HTML Templates: Create HTML templates for your web app. You can use these templates to display the chat interface.

Implement the Chatbot Logic: You need to implement the chatbot's logic in Python. You can use libraries like NLTK or spaCy for natural language processing, or you can use pre-built chatbot frameworks like Rasa or ChatterBot. In your Flask app, import your chatbot logic and integrate it with your route.

Add User Interface Elements: In your HTML templates, create user interface elements for input and chat display.

Test and Deploy: Test your chatbot locally to ensure it's working as expected. Once it's ready, you can deploy your Flask app to a web server.

Improve and Expand: You can continue to improve your chatbot by adding more features, training it on more data, and enhancing its natural language understanding capabilities.

Remember to secure your web app and validate user inputs, as this will be important in a production environment. Additionally, you can make your chatbot more interactive by integrating it with external services, databases, or APIs to provide more dynamic responses and functionality.

In this phase, we continue developing the chatbot

1.Alter the Application for Flask:

We created an application for flask in phase 3, now we run and debug the code and alter it if needed.

```

app.py > ...
1  from flask import Flask, request, jsonify
2  from transformers import GPT3Tokenizer, GPT2LMHeadModel
3  import openai # Make sure you install the 'openai' library and set your API key
4
5  app = Flask(__name__)
6
7  # Set your OpenAI GPT-3 API key
8  openai.api_key = ' sk-qLkr4P1YkGF5Afp4BvLMT3B1bkFJ1aLmbiFLtzdFDfeJAUev '
9
10 #import dataset
11 dataset="dialogs.txt"
12
13 # Initialize the GPT-3 model
14 tokenizer = GPT3Tokenizer.from_pretrained(dataset)
15 model = GPT2LMHeadModel.from_pretrained(dataset)
16
17 @app.route('/chat', methods=['POST'])
18 def chat():
19     user_message = request.json['user_message']
20
21     # Generate a response using GPT-3
22     input_ids = tokenizer.encode("User: " + user_message, return_tensors='pt')
23     response = model.generate(input_ids, max_length=100, num_return_sequences=1, no_repeat_ngram_size=2, top_p=0.5)
24     chat_response = tokenizer.decode(response[0], skip_special_tokens=True)
25
26     return jsonify({"response": chat_response})
27
28 if __name__ == '__main__':
29     app.run(debug=True)
30

```

2. HTML Template:

In this step, we alter the HTML template to a simple chat interface.

```

template.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Chatbot</title>
5  </head>
6  <body>
7      <h1>Chatbot</h1>
8      <div id="chat">
9          <div id="messages"></div>
10         <input id="user-input" type="text" placeholder="Ask a question...">
11         <button onclick="sendMessage()">Send</button>
12     </div>
13     <script>
14         function sendMessage() {
15             var userMessage = document.getElementById("user-input").value;
16             document.getElementById("user-input").value = "";
17             document.getElementById("messages").innerHTML += "<p>You: " + userMessage + "</p>";
18
19             // Send the user message to the server
20             fetch('/chat', {
21                 method: 'POST',
22                 headers: {
23                     'Content-Type': 'application/json'
24                 },
25                 body: JSON.stringify({ user_message: userMessage })
26             })
27                 .then(response => response.json())
28                 .then(data => {
29                     var botMessage = data.response;
30                     document.getElementById("messages").innerHTML += "<p>Chatbot: " + botMessage + "</p>";
31                 });
32
33             method: 'POST',
34             headers: {
35                 'Content-Type': 'application/json'
36             },
37             body: JSON.stringify({ user_message: userMessage })
38         })
39         .then(response => response.json())
40         .then(data => {
41             var botMessage = data.response;
42             document.getElementById("messages").innerHTML += "<p>Chatbot: " + botMessage + "</p>";
43         });
44     }
45 </script>
46 </body>
47 </html>
48

```

3.Create A CSS:

Here we create a CSS style sheet for chatbot's html interface.

Chatbot > # style.css > .card-header

```
1  body,html{
2      height: 100%;
3      margin: 0;
4      background: rgb(44, 47, 59);
5      background: -webkit-linear-gradient(to right, rgb(40, 59, 34), rgb(54, 60, 70), rgb(32, 32,
6      background: linear-gradient(to right, rgb(38, 51, 61), rgb(50, 55, 65), rgb(33, 33, 78));
7  }
8
9  .chat{
10     margin-top: auto;
11     margin-bottom: auto;
12 }
13 .card{
14     height: 500px;
15     border-radius: 15px !important;
16     background-color: rgba(0,0,0,0.4) !important;
17 }
18 .contacts_body{
19     padding: 0.75rem 0 !important;
20     overflow-y: auto;
21     white-space: nowrap;
22 }
23 .msg_card_body{
24     overflow-y: auto;
25 }
26 .card-header{
27     border-radius: 15px 15px 0 0 !important;
28     border-bottom: 0 !important;
29 }
30 .card-footer{
```

Ln 26, Col 14 (15 selected) Tab Size: 4 UTF-8

```

124 }
125 .user_info{
126     margin-top: auto;
127     margin-bottom: auto;
128     margin-left: 15px;
129 }
130 .user_info span{
131     font-size: 20px;
132     color: ■white;
133 }
134 .user_info p{
135     font-size: 10px;
136     color: ■rgba(255,255,255,0.6);
137 }
138 .video_cam{
139     margin-left: 50px;
140     margin-top: 5px;
141 }
142 .video_cam span{
143     color: ■white;
144     font-size: 20px;
145     cursor: pointer;
146     margin-right: 20px;
147 }
148 .msg_cotainer{
149     margin-top: auto;
150     margin-bottom: auto;
151     margin-left: 10px;
152     border-radius: 25px;
153     background-color: ■rgb(82, 172, 255);

```

