Интеграция модели в Flask приложение

В этом документе описан процесс создания Flask приложения, которое интегрирует мультимодальную модель.  
Также добавлены страницы для регистрации пользователей, владельцев и пользователей.

# 1. Структура проекта

Структура проекта состоит из следующих компонентов:  
  
- Flask-приложение (`app.py`)  
- HTML-шаблоны для отображения страниц:  
 - `index.html` — главная страница  
 - `owner.html` — страница для владельцев  
 - `registration.html` — страница регистрации  
 - `user.html` — страница для пользователя, отображающая информацию и предсказания  
- Статические файлы (например, JavaScript для клиентской логики)  
  
Проект использует Flask для обслуживания HTML-шаблонов и статических файлов, а также для взаимодействия с моделью через API.

# 2. Код Flask приложения

Вот полный код Flask-приложения, которое обслуживает страницы для получения предсказаний от мультимодальной модели:

import os  
import torch  
from flask import Flask, render\_template, request, jsonify  
from transformers import T5Tokenizer, T5ForConditionalGeneration  
from PIL import Image  
import clip  
  
app = Flask(\_\_name\_\_)  
  
# Загрузка модели и токенизатора  
device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
model = torch.load('multimodal\_model.pth', map\_location=device)  
model.eval()  
  
# Инициализация T5 и CLIP  
t5\_tokenizer = T5Tokenizer.from\_pretrained("t5-small")  
clip\_model, clip\_preprocess = clip.load("ViT-B/32", device=device)  
  
# Функция для предсказания  
def predict(text, image\_path):  
 # Обработка текста  
 inputs = t5\_tokenizer(text, return\_tensors="pt", padding=True, truncation=True).input\_ids.to(device)  
 # Обработка изображения  
 image = Image.open(image\_path)  
 image = clip\_preprocess(image).unsqueeze(0).to(device)  
  
 with torch.no\_grad():  
 # Получение предсказания  
 text\_output = model(inputs)  
 image\_features = clip\_model.encode\_image(image)  
   
 # Вычисление комбинированного вывода  
 combined\_output = (text\_output + image\_features) / 2 # Простое комбинирование для примера  
 return combined\_output.cpu().numpy()  
  
# Главная страница  
@app.route('/')  
def index():  
 return render\_template('index.html')  
  
# Страница владельца  
@app.route('/owner')  
def owner():  
 return render\_template('owner.html')  
  
# Страница регистрации  
@app.route('/registration')  
def registration():  
 return render\_template('registration.html')  
  
# Страница пользователя  
@app.route('/user')  
def user():  
 # Пример данных пользователя (это можно заменить на реальные данные)  
 user\_data = {  
 'name': 'Kolya Ivanov',  
 'predictions': [0.85, 0.90, 0.88]  
 }  
 return render\_template('user.html', user\_data=user\_data)  
  
# API для предсказания  
@app.route('/predict', methods=['POST'])  
def api\_predict():  
 data = request.json  
 text = data.get('text')  
 image\_file = request.files.get('image')  
  
 if not text or not image\_file:  
 return jsonify({"error": "Missing text or image"}), 400  
  
 image\_path = "temp\_image.jpg"  
 image\_file.save(image\_path)  
  
 # Получаем предсказания  
 prediction = predict(text, image\_path)  
  
 # Удаляем временный файл  
 os.remove(image\_path)  
  
 return jsonify({"prediction": prediction.tolist()})  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

# 3. Cтраницы для пользователя

Пример страницы для пользователя, которая отображает информацию и результаты предсказаний:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>User Page</title>  
</head>  
<body>  
 <h1>User Information</h1>  
  
 <div>  
 <h2>Welcome, <span id="userName">User</span></h2>  
 <p>Here are your prediction results:</p>  
 <pre id="userPredictionResults"></pre>  
 </div>  
  
 <h2>Make a Prediction</h2>  
 <form id="predictionForm">  
 <label for="textInput">Enter text:</label>  
 <textarea id="textInput" rows="4" cols="50"></textarea><br><br>  
  
 <label for="imageInput">Upload an image:</label>  
 <input type="file" id="imageInput" accept="image/\*"><br><br>  
  
 <button type="submit">Submit</button>  
 </form>  
  
 <h2>Prediction Result</h2>  
 <pre id="predictionResult"></pre>  
  
 <script>  
 document.getElementById('predictionForm').addEventListener('submit', async function(event) {  
 event.preventDefault();  
  
 const textInput = document.getElementById('textInput').value;  
 const imageInput = document.getElementById('imageInput').files[0];  
  
 const formData = new FormData();  
 formData.append('text', textInput);  
 formData.append('image', imageInput);  
  
 try {  
 const response = await fetch('/predict', {  
 method: 'POST',  
 body: formData,  
 });  
  
 if (response.ok) {  
 const data = await response.json();  
 document.getElementById('predictionResult').innerText = JSON.stringify(data.prediction, null, 2);  
 } else {  
 const error = await response.json();  
 document.getElementById('predictionResult').innerText = error.error;  
 }  
 } catch (error) {  
 document.getElementById('predictionResult').innerText = 'Error occurred: ' + error.message;  
 }  
 });  
 </script>  
</body>  
</html>