ภาพรวมของระบบ

ระบบนี้ทำงาน 3 ส่วนหลักๆ:

1. **การรู้จำอารมณ์จากเสียงพูดภาษาไทย** (Thai Speech Emotion Recognition)
2. **การแปลงเสียงพูดภาษาไทยเป็นข้อความ** (Thai Speech-to-Text)
3. **การสร้างภาพจากข้อความ** (Text-to-Image Generation)

ส่วนประกอบของระบบ

ระบบประกอบด้วยไฟล์หลัก 4 ไฟล์:

1. app.py - แอปพลิเคชัน Flask หลัก
2. ThaiserEmotionModel.py - โมเดลสำหรับวิเคราะห์อารมณ์จากเสียงพูดภาษาไทย
3. wav2vec2.py - โมเดลสำหรับแปลงเสียงพูดภาษาไทยเป็นข้อความ
4. imagegen.py - ฟังก์ชันสำหรับสร้างภาพจากข้อความโดยใช้ DALL-E

ขั้นตอนการทำงานแบบละเอียด

Phase 1: การตั้งค่าและเริ่มต้นแอปพลิเคชัน Flask

1. **import โมดูลที่จำเป็น:**

   ```
   from flask import Flask, render_template, request, jsonify
   import os
   from ThaiserEmotionModel import Thaiser
   from wav2vec2 import Speechtotext
   from imagegen import generate_image
   ```

2. **สร้าง Flask App**:

   ```
   app = Flask(__name__, template_folder="templates", static_folder="static")
   ```

3. **กำหนด route หลัก** สำหรับหน้าแรกของแอปพลิเคชัน:

```
@app.route("/")
def home():
  return render_template("index.html")
```

4. **รัน Flask App**:

```
python
Copy
if __name__ == "__main__":
  app.run(debug=True)
```

Phase 2: การรับและประมวลผลไฟล์เสียง

1. **กำหนด route สำหรับการอัปโหลดไฟล์เสียง:**

```
@app.route("/upload", methods=["POST"])
def upload_audio():
```

2. **ตรวจสอบไฟล์ที่อัปโหลด:**

```
if "file" not in request.files:
  return jsonify({"error": "No file provided"}), 400

audio_file = request.files["file"]
```

3. **ตรวจสอบว่ามีการส่ง transcript มาจากฝั่ง client หรือไม่:**

```
client_transcript = request.form.get("transcript", "")
```

4. **สร้างโฟลเดอร์สำหรับเก็บไฟล์ชั่วคราว:**

```
os.makedirs("uploads", exist_ok=True)
audio_path = os.path.join("uploads", audio_file.filename)
```

5. **บันทึกไฟล์เสียง:**

   file_content = audio_file.read()

   os.makedirs(os.path.dirname(audio_path), exist_ok=True)

   with open(audio_path, 'wb+') as f:

     f.write(file_content)

Phase 3: การวิเคราะห์อารมณ์จากเสียง (**ThaiserEmotionModel.py)**

1. **เรียกใช้ฟังก์ชัน Thaiser** เพื่อวิเคราะห์อารมณ์จากไฟล์เสียง:

   resultemotion = Thaiser(audio_path)

2. **การทำงานภายในฟังก์ชัน Thaiser**:
   - โหลดโมเดล wav2vec2-base-thai-ser สำหรับการรู้จำอารมณ์จากเสียงภาษาไทย

     model, feature_extractor, config = load_model_for_ser(model_name)
   - ใช้ฟังก์ชัน predict_emotion เพื่อวิเคราะห์อารมณ์

     result = predict_emotion(audio_file, model, feature_extractor, config)
   - ขั้นตอนการวิเคราะห์อารมณ์ใน predict_emotion:
     - โหลดไฟล์เสียงและแปลงเป็นความถี่การสุ่ม 16kHz

       speech_array, sr = librosa.load(audio_file_path, sr=sample_rate)
     - สกัดคุณลักษณะ (features) จากเสียง

       inputs = feature_extractor(speech_array, sampling_rate=sample_rate, return_tensors="pt", padding=True)
     - ส่งเข้าโมเดลและทำนายอารมณ์

       outputs = model(**inputs)

       logits = outputs.logits
     - แปลงผลลัพธ์เป็นคลาสอารมณ์

       predicted_class_id = torch.argmax(logits, dim=-1).item()

       predicted_emotion = emotion_labels[predicted_class_id]

- คำนวณระดับความมั่นใจ (confidence scores) สำหรับแต่ละอารมณ์

  scores = torch.nn.functional.softmax(logits, dim=-1)[0].tolist()

  confidence_scores = {emotion_labels[i]: score for i, score in enumerate(scores)}

3. **อารมณ์ที่โมเดลสามารถตรวจจับได้**:
   - "โกรธ" (anger)
   - "หงุดหงิด" (frustration)
   - "มีความสุข" (happiness)
   - "เป็นกลาง" (neutral)
   - "เศร้า" (sadness)

Phase 4: การแปลงเสียงเป็นข้อความ (**wav2vec2.py)**

1. **ตรวจสอบว่ามีข้อความจาก client หรือไม่**:

   if client_transcript:

     resultText = client_transcript

   else:

     resultText = Speechtotext(audio_path)

2. **การทำงานภายในฟังก์ชัน Speechtotext**:

   - โหลดโมเดล airesearch/wav2vec2-large-xlsr-53-th สำหรับการแปลงเสียงภาษาไทยเป็นข้อความ

     processor, model = load_model()

   - ใช้ฟังก์ชัน transcribe_audio เพื่อแปลงเสียงเป็นข้อความ

     transcription = transcribe_audio(audio_file, processor, model)

   - ขั้นตอนการแปลงเสียงเป็นข้อความใน transcribe_audio:

     - โหลดไฟล์เสียงและแปลงเป็นความถี่การสุ่ม 16kHz

       speech_array, sampling_rate = librosa.load(audio_file_path, sr=16000)

     - ประมวลผลข้อมูลเสียง

       inputs = processor(speech_array, sampling_rate=16000, return_tensors="pt", padding=True)

     - ส่งเข้าโมเดลและทำนายตัวอักษร

       logits = model(inputs.input_values).logits

       predicted_ids = torch.argmax(logits, dim=-1)

     - แปลง ID ที่ทำนายได้เป็นข้อความ

       transcription = processor.batch_decode(predicted_ids)

Phase 5: การสร้างภาพจากข้อความ (**imagegen.py)**

1. **เรียกใช้ฟังก์ชัน generate_image** เพื่อสร้างภาพจากข้อความ:

   resultimgUrl = generate_image(resultText)

2. **การทำงานภายในฟังก์ชัน generate_image**:
   - ใช้ OpenAI API กับโมเดล DALL-E 3

     response = client.images.generate(
       model="dall-e-3",
       prompt=prompt,
       size="1024x1024",
       quality="standard",
       n=1,
)
   - ส่งคืน URL ของภาพที่สร้างขึ้น

     return response.data[0].url

Phase 6: การส่งผลลัพธ์กลับไปยัง **Client**

1. **รวบรวมข้อมูลทั้งหมด** เพื่อส่งกลับไปยัง client:

```
return jsonify({
  "message": "Emotion recognized successfully",
  "probabilities": {
    "anger": resultemotion['confidence_scores'][0],
    "frustration": resultemotion['confidence_scores'][1],
    "happiness": resultemotion['confidence_scores'][2],
    "neutral": resultemotion['confidence_scores'][3],
    "sadness": resultemotion['confidence_scores'][4],
  },
  "transcript": resultText,
  "image": resultimgUrl
})
```

2. **การจัดการกับข้อผิดพลาด:**

```
except Exception as e:
  try:
    if os.path.exists(audio_path):
      os.remove(audio_path)
  except:
    pass

  return jsonify({
    "error": f"Error processing audio: {str(e)}"
  }), 500
```

สรุปกระบวนการทำงานทั้งหมด

1. **Client ส่งไฟล์เสียง** ไปยังเซิร์ฟเวอร์ผ่าน HTTP POST request
2. **เซิร์ฟเวอร์บันทึกไฟล์เสียงชั่วคราว** ใน directory "uploads"
3. **วิเคราะห์อารมณ์จากเสียง** โดยใช้โมเดล wav2vec2-base-thai-ser
4. **แปลงเสียงเป็นข้อความ** โดยใช้โมเดล airesearch/wav2vec2-large-xlsr-53-th หรือใช้ข้อความที่ส่งมาจาก client
5. **สร้างภาพจากข้อความ** โดยใช้ DALL-E 3
6. **ส่งผลลัพธ์ทั้งหมดกลับไปยัง client** ในรูปแบบ JSON
7. **ลบไฟล์เสียงชั่วคราว** หลังจากประมวลผลเสร็จสิ้น