第9章　程式

9-1 元件清單及其規格描述

▼表9-1-2 元件清單及其規格描述表(後端)

|  |  |  |
| --- | --- | --- |
| Python檔案(後端)－ChatGPT API | | |
| 編號 | 檔案名稱 | 功能 |
| 2-5-1 | house\_describe\_with\_neo4j.py  house\_describe\_with\_neo4j.ipynb | 連接neo4j給chatgpt當作回答的依據。 |
| 2-5-2 | chatgpt\_question.py  chatgpt\_question. ipynb | 連接chatgpt，且提供neo4j的房屋資料當作回答的參考依據。 |
| 2-5-3 | Line\_all\_function.py | Line聊天機器人連接到chatgpt做智慧問答。 |
| 2-5-4 |  |  |
| 2-5-5 |  |  |

9-2 其他附屬之各種元件

▼表9-2-1 部分程式碼－house\_describe\_with\_neo4j.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-5-1 | 程式名稱 | house\_describe\_with\_neo4j.py |
| 目的 | 連接neo4j給chatgpt當作回答的依據。 | | |
| 部分程式碼 | | | |
| 1.連接到 Neo4j 資料庫   * 使用 py2neo 模組連接到 Neo4j 資料庫，指定連接的 URI 和授權憑據。   2.定義函數 generate\_description   * 輸入參數：hid（房屋 ID） * 建立查詢語句：   + 匹配房屋節點 h，該節點的 hid 等於輸入參數 hid。   + 可選匹配房屋節點 h 與店家節點 s 之間的 NEAR\_STORE 關係。   + 返回房屋節點 h 和所有匹配的店家節點 s（以列表形式返回）。 * 執行查詢，將結果存儲到 result。   3.檢查查詢結果   * 如果 result 為空：   + 返回 "未找到與該HID相關的房屋資料。" * 否則：   + 獲取房屋資訊 house\_info 和店家資訊 stores。   4.初始化描述列表   * 創建一個空的描述列表 descriptions。   5.生成房屋屬性描述   * 如果 house\_info 包含 address：   + 添加描述 "房屋地址位於 {address}。" * 如果 house\_info 包含 type：   + 添加描述 "該房屋類型為 {type}。" * 如果 house\_info 包含 pattern：   + 添加描述 "房屋格局為 {pattern}。" * 如果 house\_info 包含 size：   + 添加描述 "房屋大小為 {size} 坪。" * 如果 house\_info 包含 layer：   + 添加描述 "樓層為 {layer}。" * 如果 house\_info 包含 price：   + 添加描述 "房屋租金為 {price} 元。" * 如果 house\_info 包含 subway：   + 添加描述 "距離最近的地鐵站為 {subway}。" * 如果 house\_info 包含 bus：   + 添加描述 "附近的公車站包括 {bus}。"   6.生成店家描述   * 如果 stores 不為空：   + 從 stores 中提取每個店家的 name。   + 如果 store\_names 不為空：     - 添加描述 "附近有以下店家：{店家清單}。"   7.返回完整描述   * 將所有描述合併成一個字符串並返回。   8.主程式   * 獲取用戶輸入的 hid。 * 調用 generate\_description 函數，將 hid 傳入。 * 打印生成的描述。 | | | |

▼表9-2-2 部分程式碼－xxx.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-5-2 | 程式名稱 | chatgpt\_question.py |
| 目的 | 連接chatgpt，且提供neo4j的房屋資料當作回答的參考依據。 | | |
| 部分程式碼 | | | |
| 1.**初始化與設置**   * 設定 OpenAI 的 API 密鑰。 * 使用 py2neo 模組連接到 Neo4j 資料庫，指定連接的 URI 和授權憑據。   2.**定義函數 generate\_description**   * **輸入參數**：hid（房屋 ID） * **建立查詢語句**：   + 匹配房屋節點 h，該節點的 hid 等於輸入參數 hid。   + 可選匹配房屋節點 h 與店家節點 s 之間的 NEAR\_STORE 關係。   + 返回房屋節點 h 和所有匹配的店家節點 s（以列表形式返回）。 * **執行查詢**，將結果存儲到 result。   3.**檢查查詢結果**   * 如果 result 為空：   + 返回 "未找到與該HID相關的房屋資料。" * 否則：   + 獲取房屋資訊 house\_info 和店家資訊 stores。   4.**初始化描述列表**   * 創建一個空的描述列表 descriptions。   5.**生成房屋屬性描述**   * 如果 house\_info 包含 address：   + 添加描述 "房屋地址位於 {address}。" * 如果 house\_info 包含 type：   + 添加描述 "該房屋類型為 {type}。" * 如果 house\_info 包含 pattern：   + 添加描述 "房屋格局為 {pattern}。" * 如果 house\_info 包含 size：   + 添加描述 "房屋大小為 {size} 坪。" * 如果 house\_info 包含 layer：   + 添加描述 "樓層為 {layer}。" * 如果 house\_info 包含 price：   + 添加描述 "房屋租金為 {price} 元。" * 如果 house\_info 包含 subway：   + 添加描述 "距離最近的捷運站為 {subway}。" * 如果 house\_info 包含 bus：   + 添加描述 "附近的公車站包括 {bus}。"   6.**生成店家描述**   * 如果 stores 不為空：   + 從 stores 中提取每個店家的 name。   + 如果 store\_names 不為空：     - 添加描述 "附近有以下店家：{店家清單}。"   7.**返回完整描述**   * 將所有描述合併成一個字符串並返回。   8.**定義函數 gpt\_analyze\_input**   * **輸入參數**：hid（房屋 ID），message（使用者提問） * 調用 generate\_description 函數以生成房屋描述。 * 如果描述中包含 "未找到與該HID相關的房屋資料"：   + 返回該訊息。 * 否則，構建 OpenAI GPT 的 prompt，包含房屋描述和使用者提問。 * 使用 OpenAI GPT 生成回答：   + 發送 prompt 給 OpenAI GPT-3.5-turbo-16k 模型。   + 使用系統訊息設定模型角色為專業房屋中介助手，並指定回答語言為繁體中文。 * 返回模型生成的回覆。   9.**例外處理**   * 如果調用 OpenAI API 或其他操作發生異常，捕捉異常並打印錯誤訊息。 * 返回 "在處理您的請求時發生錯誤。"   10.**主程式**   * 獲取使用者輸入的 hid 和 question。 * 打印輸入訊息。 * 調用 gpt\_analyze\_input 函數，傳入 hid 和 question。 * 打印生成的回答。 * 如果未收到回答，則打印錯誤訊息。 | | | |

▼表9-2-3 部分程式碼－xxx.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-5-3 | 程式名稱 | Line\_all\_function.py |
| 目的 | Line聊天機器人連接到chatgpt做智慧問答。 | | |
| 部分程式碼 | | | |
| 1.安裝所需的 Python 套件   * 使用 pip 安裝以下套件：   + openai==0.28：用於調用 OpenAI API。   + line-bot-sdk==1.20.0：用於與 LINE Bot API 互動。   + pyngrok：用於建立本地伺服器的 ngrok 隧道。   + Flask-Session：用於 Flask 應用程序中的會話管理。   2.引入所需模組   * 引入 ngrok 用於管理隧道連接。 * 引入 openai 用於調用 OpenAI 的 GPT 模型。 * 引入 Flask 和相關模組用於建立網頁應用和處理 HTTP 請求。 * 引入 linebot 和相關模組用於處理 LINE Bot 的消息和回調。 * 引入 py2neo 用於與 Neo4j 資料庫進行交互。 * 引入 secrets 用於生成隨機的密鑰。 * 引入 re 用於正則表達式操作。   3.設定 OpenAI API 金鑰   * 設定 openai.api\_key 為您的 OpenAI API 金鑰，用於認證和授權。   4.設定 Neo4j 連接   * 使用 py2neo 的 Graph 類連接到 Neo4j 資料庫。 * 指定資料庫的 URI 為 bolt://localhost:7687 和授權憑據（用戶名為 neo4j，密碼為 12345678）。   5.設定 LINE Bot API   * 設定 LineBotApi 和 WebhookHandler，分別用於處理 LINE Bot 的 API 請求和 Webhook 回調。 * 設定 LINE Bot 的 Channel Access Token 和 Channel Secret。   6.設定 ngrok   * 使用 ngrok.set\_auth\_token 設定 ngrok 的認證 token。 * 斷開任何現有的 ngrok 隧道連接。 * 創建一個新的 ngrok 隧道，指向本地的 5000 埠，並打印出公開 URL。   7.初始化 Flask 應用   * 創建一個 Flask 應用實例。 * 使用 secrets.token\_hex(16) 設定 Flask 應用的隨機秘密金鑰，用於會話管理。   8.全域變數   * 建立一個 user\_states 字典，用於儲存使用者狀態。   9.生成房屋描述的函數 generate\_description   * 輸入參數：hid（房屋 ID） * 建立查詢語句：   + 匹配房屋節點 h，該節點的 hid 等於輸入參數 hid。   + 可選匹配房屋節點 h 與店家節點 s 之間的 NEAR\_STORE 關係。   + 返回房屋節點 h 和所有匹配的店家節點 s（以列表形式返回）。 * 執行查詢：將結果存儲到 result。 * 處理查詢結果：   + 如果沒有找到匹配的結果，返回 "未找到與該HID相關的房屋資料"。   + 如果找到匹配的房屋資料，構建描述字符串，包含房屋的地址、類型、格局、大小、樓層、租金、最近的捷運站、公車站以及附近的店家。 | | | |