RAG 檢索增強生成應用

使用 LangChain 和 Ollama 實現的簡單 RAG (Retrieval Augmented Generation) 系統，可讀取文字檔案並回答相關問題

功能特點

- ✓ 讀取文件夾中的所有文字檔案作為知識庫
- ✓ 使用 Ollama 的 nomic-embed-text 模型進行文本嵌入
- ✓ 使用 Ollama 的 llama3-taide-lx-8b-chat-alpha 模型進行文本生成
- ✓ 提供命令行介面、API 介面和網頁前端三種使用方式
- ✓ 支援流式輸出，即時顯示生成結果
- ✓ 顯示匹配的文檔上下文，提高透明度
- ✓ 支援多種文檔編碼（UTF-8、Big5）

前置需求

在開始使用前，請確保您已準備以下環境：

- Python 3.8 或更高版本
- Ollama 已安裝並運行
- 確保 Ollama 中已安裝以下模型：
  - `nomic-embed-text:latest` - 用於文本嵌入
  - `cwchang/llama3-taide-lx-8b-chat-alpha1:latest` - 用於文本生成

安裝步驟

1. 安裝依賴套件

```
pip install -r requirements.txt
```

使用方法

1. 準備數據

將您的文本文件（.txt 格式）放在 `data` 目錄下。如果該目錄為空，系統會自動建立一個簡單的示例文件。

2. 命令行介面

運行以下命令啟動命令行介面：

```
python main.py
```

然後按照提示輸入問題，系統會先顯示匹配的上下文，然後流式輸出答案。

3. 網頁介面和 API

運行以下命令啟動 Web 服務器：

```
python api.py
```

> #### 訪問方式
>
> **網頁前端：** 通過瀏覽器訪問 `http://localhost:8000`

> **API 端點：** API 服務在 `http://localhost:8000/query` 和 `http://localhost:8000/stream_query`

API 端點詳情

- `POST /query`：發送問題並獲取回答（非流式）

  ```
  { "question": "您的問題" }
  ```

- `POST /stream_query`：流式輸出回答，先返回上下文，再流式返回答案
- `GET /health`：檢查 API 服務器狀態

專案結構

```
rag-example/
├── rag.py          # RAG 核心功能
├── main.py         # 命令行介面
├── api.py          # API 介面和網頁服務
├── templates/      # HTML 模板
│   └── index.html     # 前端頁面
├── static/         # 靜態文件目錄
│   ├── css/        # CSS 樣式
│   └── js/         # JavaScript 文件
├── data/           # 數據文件目錄
├── chroma_db/      # 向量數據庫存儲目錄（運行後自動創建）
└── requirements.txt
```

注意事項

- 確保 Ollama 服務在使用應用前已啟動

- 文本檔案應使用 UTF-8 或 Big5 編碼
- 如果沒有足夠相關的資訊，系統會回應無法找到相關資訊
- 首次運行時，系統會建立向量資料庫，可能需要較長時間

進階使用

您可以通過修改 `rag.py` 文件中的設置來自定義系統行為：

- 調整 `chunk_size` 和 `chunk_overlap` 以改變文檔分割方式
- 修改搜索參數 `search_kwargs={"k": 3}` 以改變檢索文檔數量
- 調整語言模型的 `temperature` 參數以控制生成的隨機性
- 修改提示模板以改變系統的回應風格
