RAG 語音檢索問答系統

一、系統架構

本系統為一個語音和文字作為資料來源的問答輔助平台，核心採用 Retrieval-Augmented Generation 架構，搭配多種工具模組實現完整流程，包括語音轉文字、文本修復、向量資料庫建構，以及自然語言查詢與回應。

整體流程如下：

1. **語音處理階段**：
    - 使用 OpenAI Whisper 模型進行語音辨識，將音訊檔轉為逐字稿。
    - 初步的逐字稿會送入 Gemini 進行語句修正，包括錯字、標點符號與段落格式重整。

2. **知識庫建構階段**：
    - 修復後的文字會加入 FAISS 向量資料庫，使用 Langchain 提供的介面與 Google Generative AI Embedding 模型建立語意向量。
    - 使用者可上傳任意資料進行建庫，支援txt、pdf格式。

3. **摘要生成階段**：
    - 使用者可選擇任一文字檔，透過自訂的 prompt 將內容輸入 Gemini 模型，自動生成條列式摘要。
    - 摘要內容以 Markdown 格式顯示，幫助快速掌握文件重點。

4. **查詢與回應階段**：
    - 使用者可透過 Gradio 互動介面以自然語言輸入問題。
    - 系統會先將問題轉為向量，再於資料庫中檢索最相關的片段，最後由 Gemini 回應並生成自然語言回答。
    - 此過程結合了語意檢索與生成式 AI，有效降低 hallucination 問題。

5. **學習歷程分析階段**：

- 系統會記錄所有提問與回答內容，並提供「查詢歷史紀錄匯出」功能，協助使用者回顧過往的提問與系統回應。
- 此外，透過斷詞與詞性標註，從回應中擷取高頻名詞，並生成「關鍵詞詞雲（Word Cloud)」，使用者可藉此觀察自己重複查詢的主題，掌握哪些概念尚未熟悉，是一種個人學習進度的視覺化輔助工具。

二、實現方法

- **語音處理**：
  採用 `whisper` 套件將 MP3 音檔轉成文字。使用者上傳音訊後，系統判斷是否已轉檔過，若尚未處理則進行語音辨識，並將結果保存為文字檔。

- **文本修復**：
  利用 Gemini 模型進行文本修復，透過自定義 prompt 引導模型替換錯字、加上標點符號與段落格式，進一步提升語意表達的可讀性與準確度。

- **摘要產生**：
  - 系統設計了 `summarize_prompt` ，將使用者上傳或轉錄後的文本傳入 Gemini 模型。
  - 回傳結果為條列式 Markdown 摘要，並自動儲存於 `summary_folder` 中，供使用者快速掌握文件重點。
  - 此功能亦可應用於大量影片逐字稿的整理與教學筆記生成。

- **資料庫建構**：
  - 初始版本使用 `llama-index` 搭配 Gemini Embedding 建構向量庫，但因檢索結果容易失焦，最終改為 LangChain + FAISS 的實作架構。
  - 文本將自動分段、嵌入向量，儲存至 FAISS，以支援語意檢索。

- **查詢介面**：
  - 採用 Gradio 建立互動式前端，提供語音上傳、摘要預覽與文字查詢功能。
  - 使用者輸入問題後，系統呼叫 LangChain 的 Retrieval QA Chain，從向量庫中檢索片段，交由 Gemini LLM 回應，生成具根據性的自然語言答案。

- **學習歷程分析**：

- 系統內建查詢紀錄功能，會將每一次的問題與回答儲存在 `history_log` 中，可匯出為文字檔回顧使用。
- 另提供詞雲生成功能，利用 `jieba.posseg` 萃取所有回答中的名詞，產出關鍵詞詞雲 (Word Cloud)，幫助使用者視覺化掌握自己常問主題、知識盲點與學習趨勢。

三、實際查詢案例

案例 1：

- **查詢時間**：2025-03-27 17:43:30
- **提問**：RNN的資料量太大會發生什麼問題
- **回應**：

> RNN的資料量太大，無止境地記憶權重，會出現兩種狀況：梯度消失 (vanishing gradient) 或梯度爆炸 (exploding gradient)。
>
> - **梯度消失 (vanishing gradient)**：權重在經過一次次的計算之後，越來越小，最後趨近於 0。
> - **梯度爆炸 (exploding gradient)**：權重越來越大，最後數值過大而爆掉。

案例 2：

- **查詢時間**：2025-03-27 17:43:40
- **提問**：那該怎麼解決
- **回應**：

> 文本中提到，RNN資料量太大導致梯度消失或梯度爆炸的問題，後來有人推出了RNN的變形，其中最有名的是 LSTM 和 GRU。 但具體如何解決，將會在下一支影片中介紹，因此我不知道。

案例 3：

- **查詢時間**：2025-03-27 17:45:46
- **提問**：請向我介紹資料預處理
- **回應**：

根據提供的文本，資料預處理包含以下步驟：

1. **檢查空值 (NA)**： 尋找各個欄位裡面是否有 NA 值。
2. **NA 補值**： 對於缺失值進行填補，可以使用以下方法：
    - 以 0 補值 (`fillna(0)`)
    - 用前一筆資料補值 (`fillna(method='ffill')`)
    - 用後一筆資料補值 (`fillna(method='bfill')`)
    - 用眾數 (Mode) 補值
    - 用平均值 (Mean) 補值 (例如：針對花瓣長度欄位，用該欄位的平均值補值)
3. **資料轉換**： 常用的資料轉換是標準化，可以去除各欄位的單位。
4. **One-Hot Encoding**： 將類別型資料轉換為數值型資料。
5. **隨機抽樣**： 將資料集分割成訓練集和測試集 (例如：70% 作為訓練集，30% 作為測試集)。可以使用 `random_state` 來確保每次抽樣結果一致，概念類似於 `set.seed()` 。

此外，文本中還提到兩個重要的資料預處理步驟，但未在此次處理中展示，將在之後的影片中介紹：

- **離群值的找尋**
- **特徵工程 (自變量的篩選)**： 可以移除或合併特徵。

四、遇到的挑戰與解決方案

1. **語音辨識品質低落**：
    - Whisper 雖功能強大，但在中英夾雜與專有名詞上辨識率偏低，常出現明顯錯誤（如將 PyTorch 辨識為「拍拓」）。
    - **解決方案**：將初步文字輸入 Gemini，請其根據上下文修正錯誤，準確率從原本五成提升至九成以上，成效顯著。

2. **語意檢索失焦問題**：
    - 初始使用 GeminiEmbedding 建立向量資料庫，但在處理長篇或多個不同主題的文本時，容易找不到答案。
    - **解決方案**：改用 LangChain 框架與 FAISS，失焦問題大幅改善，搜尋結果更穩定精準。

五、學習心得與反思

優勢

這套系統最大優勢在於「自然語言查詢 + 精準回應」，讓使用者可以像真人對話一樣提出模糊問題，也能獲得依據文本的具體答案。這相較於傳統搜尋引擎需要準確關鍵字查詢的方式，更貼近人類思維。

此外，透過結合 Gemini 模型的語意理解能力與資料庫的實際文本依據，可大幅降低 LLM 的幻覺機率，讓回答更可信。

不足與改進方向

Whisper 對中英夾雜與專業術語仍不夠精準，考慮串接更強的語音辨識模型 API，或由使用者直接上傳文本。

若知識庫中文件主題過多，仍可能影響查詢準確性，未來可嘗試加入主題分類機制，提升檢索相關性。

發想與延伸應用

這次作業給我帶來的很大的啟發，我計畫將這套系統改造為「遊戲劇情問答助手」。我所遊玩的一款遊戲，每 42 天更新一次劇情，但每次更新時，常常忘了之前的細節，透過這套系統輸入過往劇情的文本，即可隨時提問、回顧角色背景與事件關聯，非常實用。

六、成果展示

功能示範影片 (https://www.youtube.com/watch?v=YX8PeTnjxFg)