【Day 10】小結與延伸方向

今天是第一階段「RAG 個人助手」的最後一篇。在過去的九天裡，我們從零開始，搭建了一個功能完整的 RAG 個人助理。今天，讓我們一起回顧這段旅程，並展望未來可以探索的延伸方向。

1. 小結

結論：回顧

- 學會了 LangChain 的基本操作，知道如何搭建一個可以對話的系統。
- 使用過 FAISS 建立並讀取資料庫。
- 實作完整 RAG 流程、利用 Streamlit 建構簡單的介面並部署上線。

心得：寫文章比想像中的累

從規劃進度、寫 code、讀文檔到寫文章每個環節都花了比預期還多的時間，但在這個過程中對於這些 python 的套件有更深刻的理解，也更期待接下來能從論文中學到可以讓 RAG 系統變的更強大的技術。

2. 延伸方向

以下我將從「整體應用框架」、「資料庫」與「核心技術」三個面向，提出一些延伸或替代方向：

2.1 整體應用框架

除了自己用 LangChain 與 Streamlit 掌握所有細節，其實以這個 RAG 個人助手來說，使用其他工具的開發效率可能更高：

- **n8n**：是工作流自動化工具。我們可以將 RAG 助手作為 n8n 的一個節點，與 Google Drive、Notion 串接，打造自動更新資料庫的流程。
- **Dify.AI**：一個開源的 LLM 應用開發平台，提供了視覺化的操作介面來建構、部署和管理 AI 應用。它內 RAG 的工作流，非常適合 RAG 個人助手這個應用
- **AnythingLLM**：一個開源、可自行部署的 RAG 解決方案。有現成的聊天介面，簡單的將文件拖入系統，就可以開始在本地使用 LLM-RAG 系統。

2.2 資料庫

我們的 Demo 使用本地存放的 FAISS，這在原型開發階段很方便。但若要走向正式應用，資料庫的管理需有更多著墨：

- **雲端向量資料庫 (Pinecone, Weaviate, Milvus)**：相較於本地 FAISS，Pinecone 這類的託管服務提供了更好的擴展性、穩定性與管理功能。它們通常也支援更進階的篩選（例如 Metadata Filtering）與 Hybrid Search 功能。
- **用 SQLite 等資料庫管理 Raw Data**：目前原始資料簡單的存放在一個資料夾，更穩健的做法是，使用 SQLite 或 PostgreSQL 等傳統資料庫來管理原始文件與其 metadata。如此一來，當需要更新或刪除文件時，我們可以輕易地透過資料庫操作，找到對應的 `chunks` 及其在向量資料庫中的 `ids`。

2.3 技術

在 Day 2 我們提到 RAG 的優化方向，現在我們有了實作經驗，更能體會這些技術的重要性。

- **Chunking**：我們使用了基於字元與分隔符號的基礎分塊策略。而這樣的策略當然有不少進步的空間，在實際使用上，還可以根據不同的資料類型與風格設計策略。
- **Retrieve**：我們的 Demo 只用了基本的相似度搜尋。一個常見的優化是**混合搜尋 (Hybrid Search)**，它結合了我們所用的「語意搜尋」與傳統的「關鍵字搜尋」（如 BM25，關鍵字匹配），能改善對專有名詞或特定術語的檢索效果（改善 recall）。另一方面，**重排序 (Re-ranking)**，也是常見的作法。初步檢索（例如 k=20）後，使用一個更強大但計算成本高的模型，對這 20 個文件進行**重新排序**，選出最相關的 5 個送入 LLM（改善 precision）。
- **Embedding**：Embedding 的品質直接決定檢索的天花板。通用的 Embedding 模型（如我們用的 `gemini-embedding-001`）在多數場景下表現不錯，但我們也可以多測試看幾個模型，看誰的表現好，例如 `bge-m3`、`E5` 等。而若資料集有特定領域知識，可能就需要考慮微調（Fine Tuning）一個 Embedding。