【Day 7】 RAG 架構搭建：從 FAISS 到 QA 問答

在做完前面三天的準備工作，我們可以來進行第一次的 RAG 問答。

1. 載入資料庫

用 `FAISS.load_local()` 來讀取本地的 **FAISS** 資料庫

```
vector_store = FAISS.load_local(
    "../vector_store",
    GoogleGenerativeAIEmbeddings(model="models/gemini-embedding-001"),
    allow_dangerous_deserialization=True
)
```

將資料庫變成 **retriever** 就可以將它串接到我們的 chain 裡面。要定義搜尋的方法以及參數。

`search_type` 有 “similarity”, “mmr”, 和 “similarity_score_threshold” 三種。參數 `k` 表示要檢索的數量，如果使用 “similarity_score_threshold” 方法，還可以再設置 `score_threshold` 來限制相似度的門檻。對於 “mmr” 方法則有 `fetch_k` 和 `lambda_mult` 兩個參數。

```
retriever = vector_store.as_retriever(
    search_type="similarity",
    search_kwargs={"k": 5},
)
```

2. Retrieve Pipeline

我們可用 langchain 提供的兩個函數來建立 RAG Chain，這裡比較複雜，我們有以下幾件事要做：

- 使用 input (**q**) 進行檢索，取得 context（是一個 documents 的列表）
- 將 context 整理成字串 (**P**)

- 將 **q** 和 **P** 一起傳入 prompt、呼叫 LLM、取得 answer
- 回傳 input, context, answer

1. `create_stuff_documents_chain` 將 llm 和 prompt 轉換成一個可以接收多個文件的 **runnable**，預設會傳入到 prompt template 的 **context** 變數裡面。
2. `create_retrieval_chain` 將可以接收多個文件的 runnable 和 retriever 串接在一起，用 **q** 進行檢索並將它傳入到 prompt template 的 **input** 變數。最後回傳 input, context, answer 三個 key。

```
from langchain.chains.combine_documents import create_stuff_documents
_chain
from langchain.chains.retrieval import create_retrieval_chain

prompt = PromptTemplate.from_template(
    "你是 Rasmus 的履歷助手，負責依據資料庫回覆使用者的問題。如果你不知
道，請直接說不知道。\n **context**: {context}\n\n **input**: {input}\n\n"
)

combine_docs_chain = create_stuff_documents_chain(
    llm, prompt
)

retrieval_chain = create_retrieval_chain(retriever, combine_docs_chain)
```

3. 自己用 Runnables 搭建

看到上面的代碼，我們可能會想，這些函數是不是有點過度包裝？或是如果我想在 RAG 的過程加入自定義的邏輯要怎麼做。我們可以自己用 **runnable** 來搭建一次。

3.1 Retrieve 階段

我們實際上要做的事情是增加一個新的 key: **context**，同時保留原本的 query (**q**) 。`RunnablePassthrough().assign()` 剛好適合用在這：

```
Input:
{
    "input": q
}
```

```
Output:
{
   "input": q,
   "context": retriever(q)
}
```

```
from langchain_core.runnables.passthrough import RunnablePassthrough

retrieve = RunnablePassthrough().assign(
      context=lambda x: retriever.invoke(x['input'])
   )
```

3.2 Generate 階段

我們要最終的輸出再多一個 key: **answer**，我們一樣可以用 `RunnablePassthrough().assign()`，但這次要多定義一個函數來整理 context，因為 retrieve 來的 context 是包含 Documents 的列表，要把它轉成字串。這邊我們取出 `page_content`，再簡單的用換行符隔開。

最後，把 `retrieve` 跟 `generate` chain 起來。

```
Input:
{
   "input": q,
   "context": [doc1, doc2, ..., doc5]
}

Output:
{
   "input": q,
   "context": [doc1, doc2, ..., doc5]
   "answer": llm(q, concat(context))
}
```

```
concat = lambda x: "\n\n".join(c.page_content for c in x)
llm_chain = prompt | llm

generate = RunnablePassthrough().assign(
    answer=lambda x: llm_chain.invoke(
        {
            "input": x["input"],
            "context": concat(x["context"])
        }
    )
)

rag_chain = retrieve | generate
```

4. 結果展示

Input:

Rasmus 如何開啟對行銷的興趣？

Context:

- ---h1--- 關於 Rasmus Zhu：從會計數字到品牌敘事
- ---h2--- 我的故事：在理性與感性之間成長\n我叫朱昀熙，英文名 Rasmus，這個名字背後有個有趣的故事。我的父親是位嚴謹的會計師，從小教育我「數字不會說謊」；母親則是位熱愛文學與藝術的老師，鼓勵我「故事能感動人心」。\n我成長在這樣一個理性與感性交織的家庭，學會了用嚴謹的眼光觀察世界，也學會了用溫暖的筆觸描繪生活。或許正是因為如此，我在面對會計系密密麻麻的數字時，總能想起母親教我的故事結構；而在撰寫行銷文案時，父親的數字思維又能幫助我評估成效。'
- ---h3--- 會計背景的起點\n會計系是我大學聯考時的選擇，當時我認為這是最「務實」的科系，能為我打下紮實的商業基礎。我並不排斥數字，反倒享受從海量資料中抽絲剝繭的過程。\n---h3--- 發現行銷的轉捩點\n大二那年，我開始感到一絲迷惘。在一次實習中，我看到許多企業在財報上表現亮眼，卻在市場上聲量低迷，我才意識到，**一個品牌的成功不單單是財務上的健全，更需要動人的故事與精準的溝通**。從那時起，我開始積極探索行銷領域，希望能將我的數字分析能力，與品牌敘事結合。

- ---h2--- 跨出舒適圈：從會計人到提案者\n除了自媒體，我也積極參與各種行銷競賽與課程專案，將我的行銷思維推向實戰。
- {'question': '您是如何選擇目前的學習領域或職涯方向的？有什麼關鍵影響因素？', 'answer': '我的職涯選擇源於一個跨領域的探索過程。我來自一個『理性與感性交織』的家庭，父親是會計師，母親是老師，因此我選擇了會計系作為務實的起點。然而，在學習過程中，我發現自己對運用『創意敘事』來影響他人的行銷領域產生了濃厚的興趣。我透過創立IG專頁、參與課程專案和行銷競賽來探索這個領域，最終發現自己能將會計的嚴謹數字思維與行銷的創意敘事完美結合，這成為我選擇行銷作為未來職涯方向的關鍵因素。', 'category': 'school'}

Answer:

'Rasmus 在大二那年，透過一次實習發現許多財務表現亮眼的企業在市場上聲量卻不高，這讓他意識到品牌的成功不僅需要財務健全，更需要動人的故事與精準的溝通。從那時起，他開始積極探索行銷領域。'