靜 宜 大 學

資訊工程學系

畢 業 專 題 成 果 報 告 書

語音互動式校園智慧助理

指導教授：蔡英德

學生：

資工四B　411100621　蔡頌望

資工四B　411147835　侯書濠

資工四B　411147283　謝景勛

資工四B　411147128　余佳俊

資工四B　411147348　陳柏諺

西 元 二 〇 二 五 年 十 二 月 四 日

摘要

隨著大型語言模型與語音合成技術的進步，語音對話式服務已逐漸成為人機互動的新型態。本專題設計並實作一套「語音互動式校園智慧助理」，以 LLM（Large Language Model） + TTS（Text-To-Speech） 課程介紹系統為核心，結合網頁前端、語音辨識、檢索式知識庫與 Unity 虛擬角色，提供使用者以自然語音或文字詢問靜宜大學課程相關資訊的服務。系統端透過程式化爬蟲批次蒐集並整理課程綱要與開課資訊，利用中文向量嵌入模型建置向量資料庫，再以 RAG（Retrieval-Augmented Generation）架構，在回答前先從資料庫中擷取與問題語意最接近的內容，降低大型語言模型產生不實回答的機率，並提升回覆與實際課程資訊的一致性。

後端主要採用本地部署的 Breeze-7B-Instruct 語言模型，搭配 LoRA 進行風格微調，使回答方式更貼近校園導覽情境；語音部分則整合 Whisper 語音辨識與 CosyVoice2 語音合成，讓使用者可以以說話的方式提問並聆聽的方式取得回覆。系統運作流程為：前端網頁或行動裝置將文字／語音請求送至 FastAPI 伺服器，由伺服器呼叫 LLM、RAG 與 TTS 模組完成理解、檢索與生成，並將產生的語音檔與字幕以串流方式主動推送至 Unity。Unity 端的 Live2D 虛擬人物再依序播放語音、顯示字幕，並根據音量驅動嘴型與表情變化，使其具備更多的互動感。

本專題的整體架構採伺服器－客戶端模式：LLM、RAG、STT 與 TTS 均集中部署於本地伺服器，而 Unity 虛擬角色與網頁介面可獨立運行於另一台電腦或 Android 裝置，只要能透過區域網路或網際網路與伺服器連線即可使用。此設計可確保模型運算效能，也提升系統在實際校園導覽情境中佈署與擴充的彈性，未來可進一步延伸至系所介紹、新生輔導或校園活動宣傳等不同應用場域。

目錄

圖目錄

第一章　緒論

近年來，隨著人工智慧與深度學習技術的快速發展，大型語言模型（Large Language Model, LLM）在自然語言理解與生成上的表現有顯著提升。搭配語音辨識（Speech-to-Text, STT）與語音合成（Text-to-Speech, TTS）技術，人機互動的模式已不再侷限於鍵盤輸入與螢幕文字顯示，而是逐漸朝向更加多元的互動方式發展。本專題希望使用對話的互動方式構建系統，使用者只要像與真人對話一樣提出問題，系統便能理解語意、查詢相關資料並回應，帶來更直覺與自然的操作體驗。

在大學校園中，各個系所的課程資訊及各類學習資源日益多元，學生經常需要在選課前後查詢大量資訊，例如課程內容、授課老師、上課時間與地點、評分方式等。雖然學校有提供選課系統與課程綱要的查詢網站，卻往往以表格與靜態文字呈現，介面也多為傳統資訊查詢系統的設計，對於不熟悉操作流程或臨時需要快速查詢的使用者而言，仍需要花費一定時間搜尋與比對。若學生想詢問較具彈性的問題，例如「有哪些和動畫或 3D 設計相關的課？」或「想走某種職涯方向可以選哪些課？」時，現行系統也較難直接給出整合性的建議。

在此背景之下，本專題希望結合大型語言模型、檢索增強生成（Retrieval-Augmented Generation, RAG）、語音辨識與語音合成等技術，建置一套「語音互動式校園智慧助理」。透過課程資料的整理與知識庫化，讓系統能理解使用者以自然語言提出的問題，並從課程相關資料中找出合適內容，再由 LLM 調整成易懂的回答。同時，系統再搭配 TTS 將文字回覆轉成語音，並透過 Unity 中的虛擬角色（Live2D）進行播放與表情呈現，讓使用者感覺像是在對話的互動方式，而不只是面對一個文字介面。

本專題的系統架構採伺服器－客戶端模式。後端伺服器集中負責課程資料爬取與整理、向量資料庫建置、LLM 推論生成、RAG 檢索、語音辨識與語音合成等較耗資源的工作；前端則包含兩個主要互動方式：其一為網頁介面，讓使用者可以在瀏覽器中輸入文字或上傳錄音檔進行提問；其二則是 Unity 製作的虛擬角色介面，用於展示語音回覆與字幕，並依照語音的節奏

控制嘴型和表情。透過這種分工方式，系統可以在維持本地運算穩定性的前提下，仍具備一定的部署彈性與延展性，只要網路環境允許，即可在不同裝置上使用。

此外，考量到實際應用情境，本專題也特別重視系統在互動品質與使用體驗上的表現。例如：如何讓 RAG 檢索結果更貼近使用者的真實需求、如何避免模型在找不到資料時勉強「亂答」、如何在語音生成與播放上減少等待時間、如何讓虛擬角色的呈現不會干擾資訊獲取等，都是專題實作過程中需要面對並解決的問題。透過實作與測試，本專題期望提出一套在技術上可行、在實務上也具參考價值的校園智慧助理雛形，作為未來校園導覽系統或智慧客服應用的基礎。

第二章　專題內容與進行方法專題內容與進行方法

2.1 動機與目的

2.1.1 專題動機

近年來，大型語言模型（Large Language Model, LLM）與語音辨識、語音合成等技術的快速發展，讓「可以聽得懂人話、也能開口說話」的互動式系統變得越來越普及。相較於傳統只能透過點選選單、填寫關鍵字搜尋的靜態網頁，使用者只要用自然語言提問，就能更直覺地表達自己的需求，並在短時間內取得與自己需求最相關的資訊。這樣的互動方式，特別適合對系統介面不熟悉或臨時需要查詢課程資訊的使用者。

在校園場域中，多數學校雖然已經提供選課系統與課程查詢網站，但介面往往偏向傳統靜態網頁類型，需要知道課號、開課單位或複雜的篩選條件，才能找到想要的資料。對於沒有課程資訊的使用者來說，現有系統不一定友善。此外，現有網頁多半以文字列表呈現，缺乏互動感無法提供像真人諮詢那樣自然的對話體驗。

因此，本專題希望結合大型語言模型、RAG（Retrieval-Augmented Generation）以及語音技術，打造一個能對話互動的系統。我們希望使用者可以用自然語言提問，例如「想找跟動畫或 3D 設計有關的課」，系統能聽得懂並給出適切的課程建議；也希望透過虛擬人物的呈現，讓使用者有在和一位角色聊天的感受，而不是在面對冰冷的文字介面。

此外，在實作層面上，我們也希望系統一定程度的「機動性」與「部署彈性」。若每一次展示或使用都必須在同一台電腦上安裝完整環境，不但維護成本高，也不利於在各種活動中靈活運用。因此，本專題希望將運算較重的 LLM、RAG 與 TTS 集中在後端伺服器運行，而虛擬人物可部署在不同電腦或手機，使用者可使用手機在 Web 輸入介面進行對話，只要能連線到伺服器，就可以快速啟用。同時，也期望透過這樣的設計，讓未來若要擴充到其他查詢主題（例如校園導覽、行政服務 Q&A 等）時，只需要在後端更新知識庫與模型設定即可。

2.1.2 專題目標

本專題的核心目標，是建置一套結合 RAG、本地 LLM 與 TTS 的對話後端，並搭配 Web 輸入端與 Unity 虛擬人物端，形成一個從「資料蒐集、知識檢索」到「文字與語音回應、虛擬角色呈現」的一體化系統。

首先，在資料面，我們希望透過程式化爬蟲抓取靜宜大學課程相關資訊，並進行整理與前處理，例如統一欄位格式、清理多餘符號、將課程名稱、授課教師、時間地點、課程簡介等內容結構化。這些整理後的資料將被轉換為適合向量化的文字內容，並存放於 RAG 使用的向量資料庫中，作為穩定且可更新的知識來源，以因應學期更替或課程異動。

其次，在對話後端方面，我們的目標是實作一個「先檢索、再生成」的流程：當使用者以文字或語音提出問題時，系統會先利用 RAG 從課程向量資料庫中找出與問題最相關的課程說明或欄位內容，再將這些資料連同使用者提問一併送入本地 LLM，生成較為精準且符合上下文的回答。

第三，在互動介面與體驗上，本專題將整合語音辨識與語音合成，讓使用者不僅可以打字詢問，也能用說話的方式與系統溝通。語音辨識模組會將使用者的語音轉成文字交給後端處理，而 TTS 模組則負責把 LLM 的文字回覆轉成自然的語音輸出，同時驅動 Unity 虛擬人物的嘴型與動作，使回應過程更有真人對話的感覺。

最後，在系統架構與部署方式上，本專題目標是實現「運算集中於後端、跨裝置運行」的模式：後端伺服器負責執行 LLM、RAG 與 TTS 等運算密集的部分；Unity 虛擬人物端則以客戶端形式安裝在另一台電腦或 Android 裝置上，透過網路接收文字與語音回應；Web 端則提供以瀏覽器進行文字或語音輸入與顯示對話紀錄的介面。藉由將三者分工，我們期望系統能可以依各種需求進行配置，只需要攜帶一台可連線的平板與一台後端主機，就能快速架設一個可對話的導覽角色，達成機動性、部署彈性與可擴充性的目標。

2.2 專題相關現有系統回顧與優缺點分析

2.2.1現有系統分析

（一）校內既有課程查詢與選課系統

課程查詢與選課系統以表格與查詢欄位呈現，使用者可以依學年期、開課系所、課程名稱或授課教師等條件篩選，並檢視課程綱要、上課時間、教室與評分方式等資訊。

這類系統的優點在於：

- **資料詳細**
- **欄位結構清楚**

然而，此類系統在實際使用上也有以下幾種限制：

- **操作門檻較高**：使用者必須對課程有一定程度的了解。
- **不支援自然語言提問**：使用者無法以我想修跟動畫相關的課或想了解大二可以選的程式設計課等方式直接詢問。
- **缺乏互動與引導**：系統多為被動查詢，較少提供主動建議或多門課程的比較說明。

（二）聊天機器人與 FAQ 系統

部分學校或單位已經嘗試導入聊天機器人或 FAQ 系統，例如在官方網站或通訊軟體（如 LINE）上提供預設問答，協助解答常見問題。

這類系統的優點在於：

- **常見問題回覆快速**：只要問題有預先建檔，系統便能立即提供標準答案。

- **介面親近**：嵌入常用通訊軟體，學生無須額外安裝或登入系統。

但此類系統也常面臨以下限制：

- **問題類型受限**： FAQ 系統以關鍵字比對為主。

- **缺乏彈性組合回答**：若問題涉及多門課程或跨單位資訊無法提供完整建議。

2.2.2 與本專題定位的比較

傳統課程查詢系統在資料完整性方面具有優勢，但資訊龐大且搜尋方式複雜；聊天機器人與 FAQ 系統則偏重預設問答資訊，對於需要細緻查詢與比較的內容支援較為有限。

本專題嘗試結合上述系統的優點並改善其不足之處，資料方面沿用校內課程資料作為知識庫來源，確保資訊的正確性；透過 LLM 與 RAG 的結合，讓系統得以以自然語言互動並將檢索到的課程資料整理成易理解的說明。

第三章　系統架構與設計

3.1　系統 UML 圖系

- 系統用例圖（Use Case Diagram）

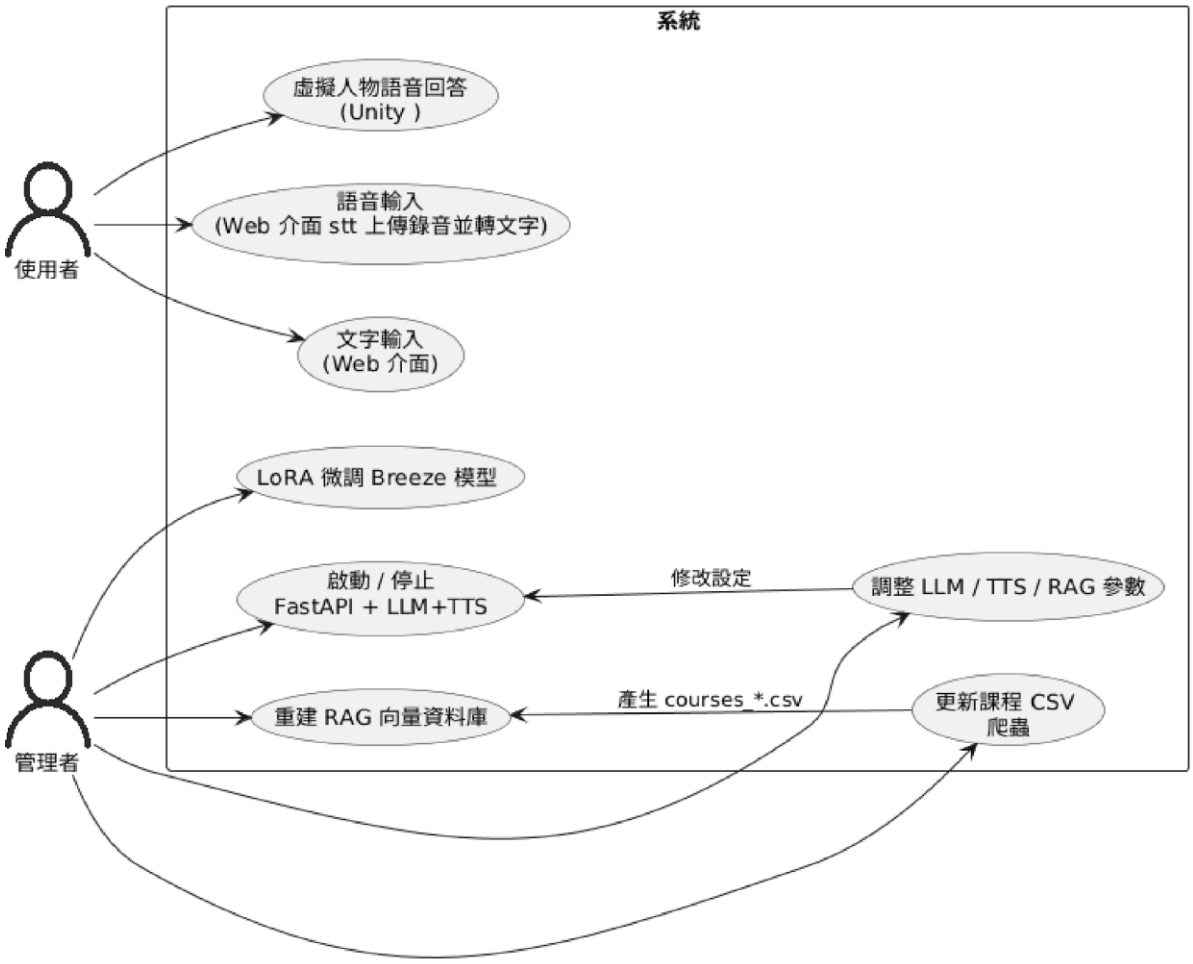

圖 3-1 系統用例圖（Use Case Diagram）

- 元件圖（Component Diagram）

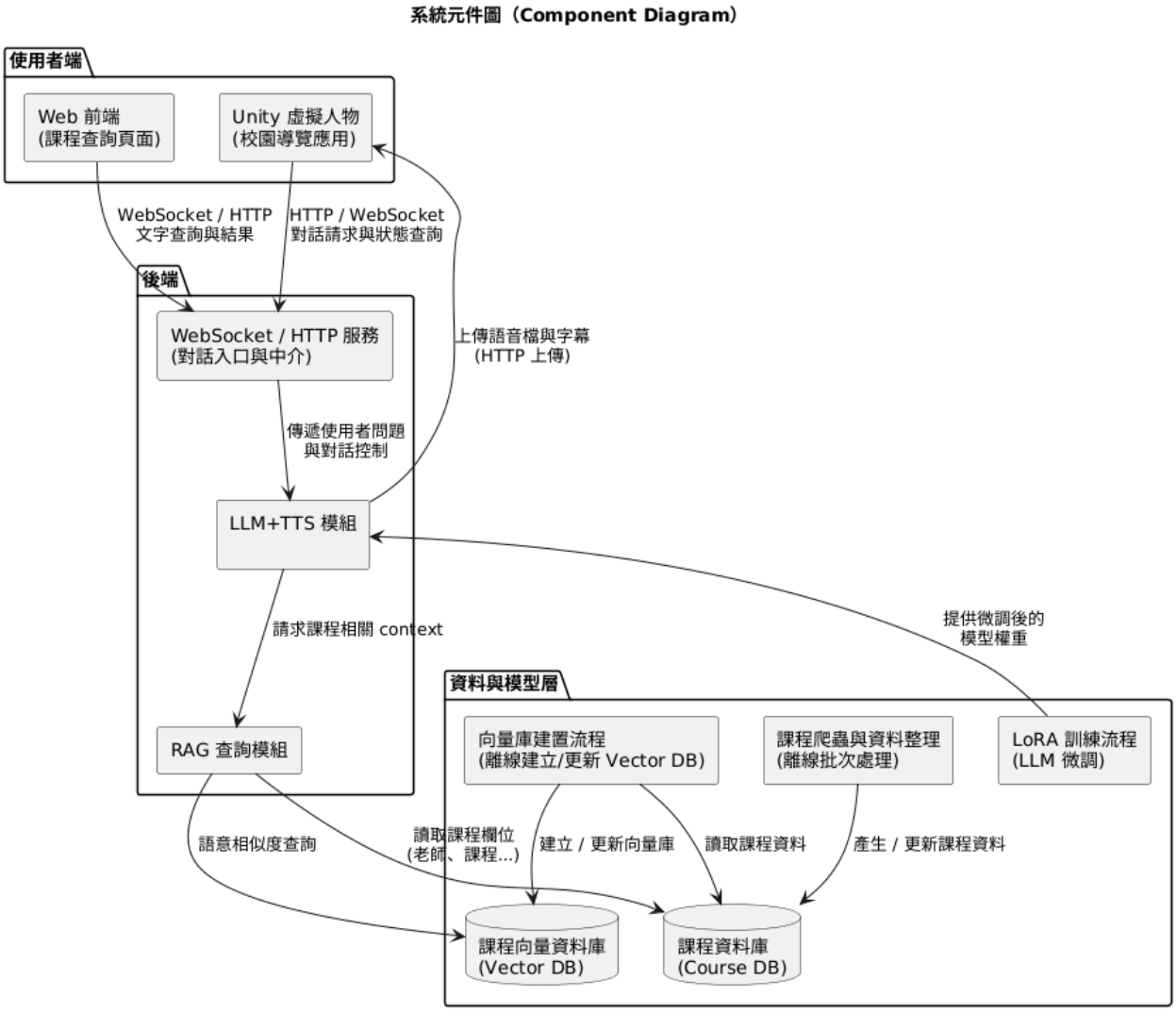

圖 3-2 系統元件圖（Component Diagram）

- Web 文字回覆時序圖（Sequence Diagram）

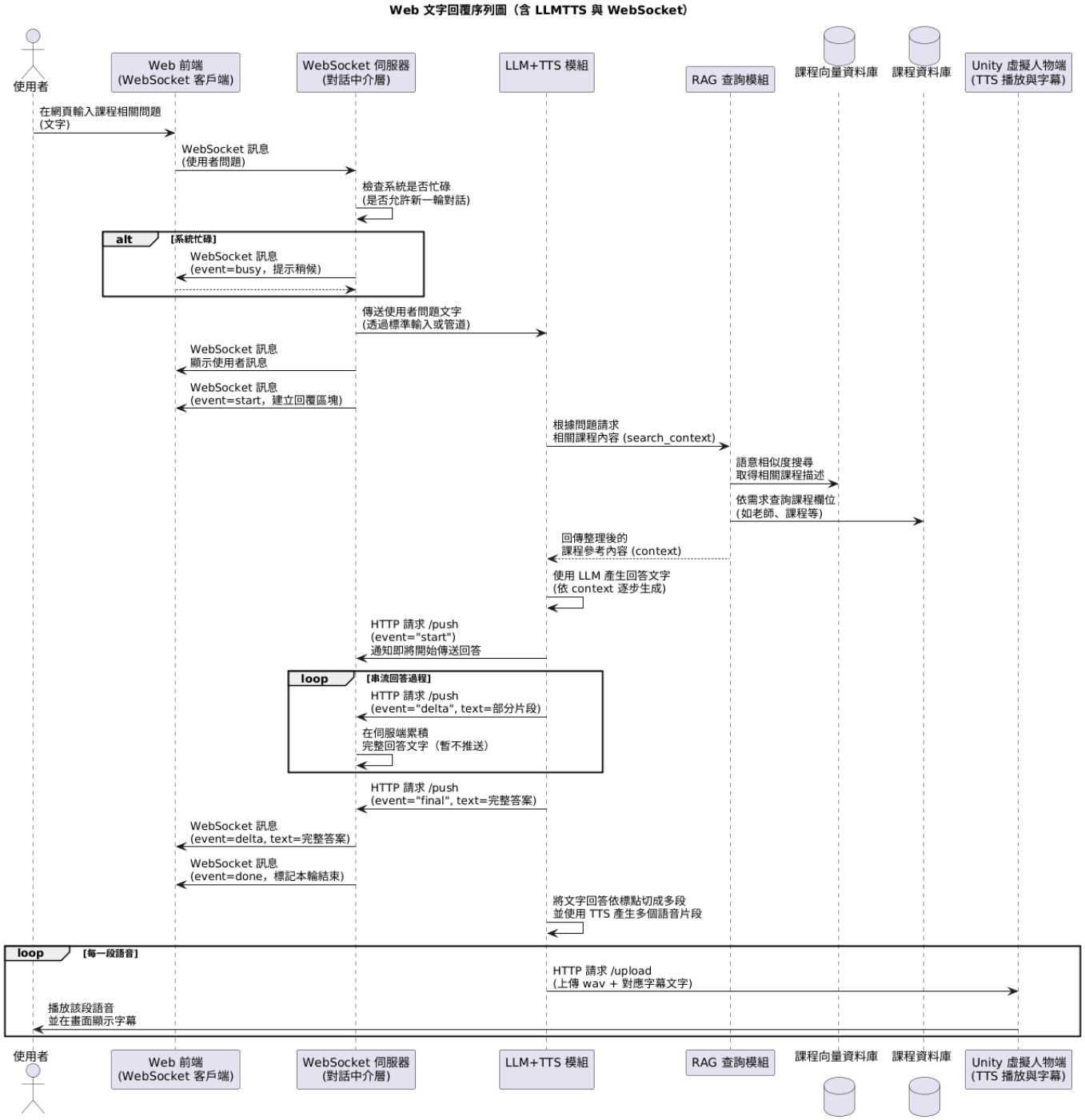

圖 3-3 Web 文字回覆時序圖（Sequence Diagram）

3.2 統架構圖

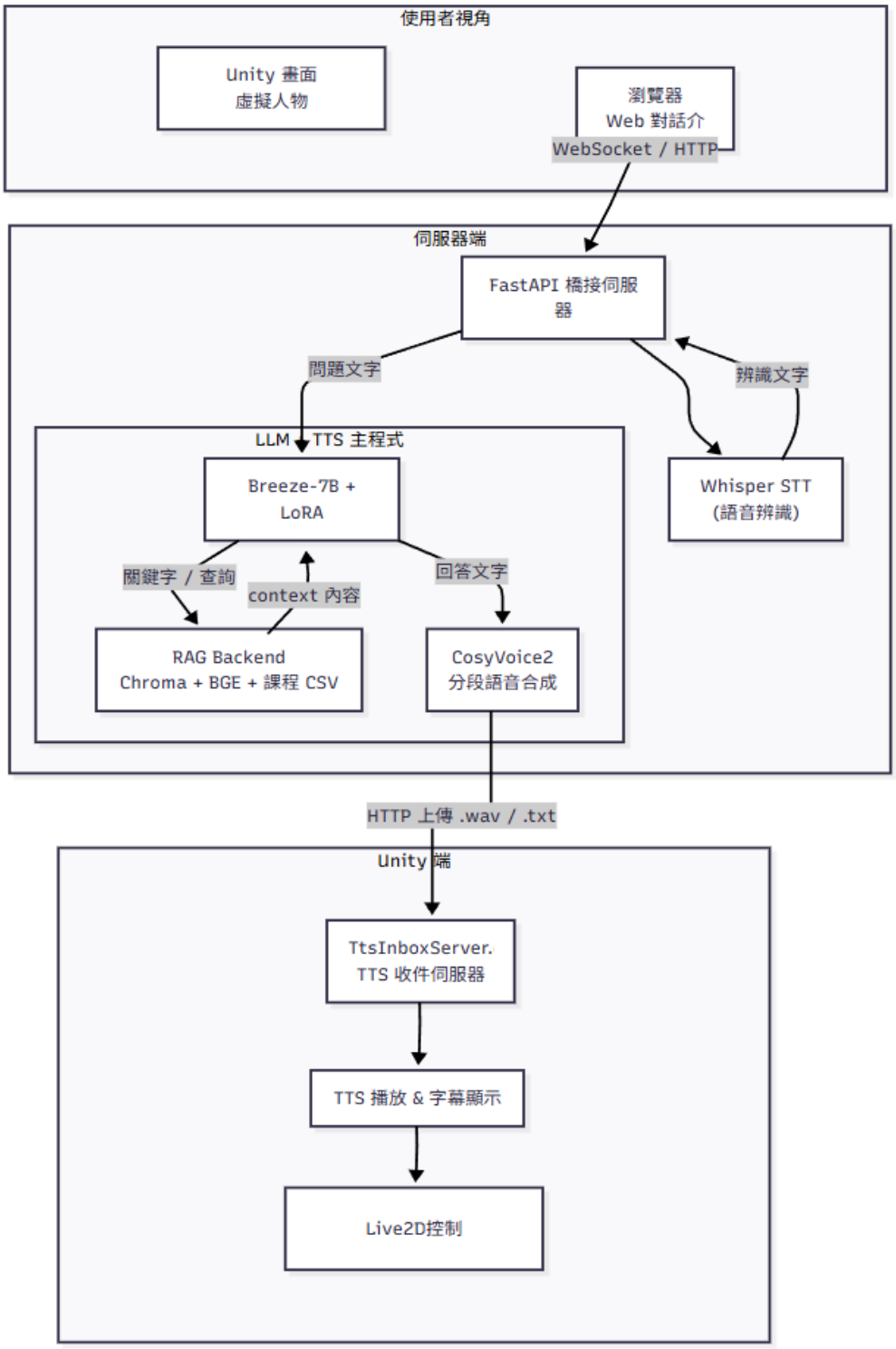

圖 3-4 整體系統架構圖

3.3 系統功能說明

（一）Web 對話介面

使用者透過瀏覽器開啟頁面，在畫面中的輸入框輸入問題，或透過麥克風錄製語音後上傳，並將請求傳送至後端伺服器取得回覆，再於畫面上顯示文字結果(如圖 3-5 所示)。

圖 3-5 Web 使用者輸入介面

（二）FastAPI 伺服器與工作管理

FastAPI 伺服器負責接收前端送來的文字與語音請求，啟動並監控 LLM＋TTS 程式，管理模型是否就緒以及是否正在回覆等狀態，並將回覆的文字透過 WebSocket 傳回前端顯示。

（三）LLM+TTS 主程式

整合大型語言模型與語音生成，根據使用者問題生成文字回答，並將回答內容轉換為語音檔與對應字幕檔，提供後端伺服器推送給 Unity 播放。

（四）RAG

接收到問題後，從向量資料庫中檢索出語意與字面上都較為相關的段落，整理後附加在使用者提問後面，協助語言模型回答課程資訊相關問題。

（五）STT

伺服器提供語音轉文字（STT），將前端上傳的錄音檔交由本地 Whisper 進行辨識，轉成文字後與文字輸入的流程整合，讓使用者可以用說話的方式進行提問。

（六）Unity 端

Unity 利用 HTTP 接收後端主動推送的語音檔與字幕檔，依順序播放音訊並在畫面下方顯示字幕，並透過音量控制 Live2D 人物嘴型、眨眼、呼吸與身體擺動的變化。

圖 3-6 Unity 虛擬人物與字幕顯示畫面

第四章　專題成果介紹

4.1　軟體環境與使用套件

本專題採用 Windows 環境搭配 Python、Unity 及多項開源套件進行開發，主要套件說明如下：

作業系統與開發工具

- 作業系統：Windows 11
- Python 環境：Python 3.10.6（Anaconda 管理虛擬環境）
- 開發工具：Visual Studio Code
- Unity：Unity 6.2
- 瀏覽器：Chrome，用於測試 Web 對話介面。

後端伺服器與 Web 框架

- **FastAPI**：作為後端 Web Framework，負責提供 REST API 與 WebSocket 服務。
- **Uvicorn / Starlette**：搭配 FastAPI 的 ASGI 伺服器，用於非同步請求處理。
- **websockets／requests 等套件**：處理 WebSocket 連線與 HTTP 請求。

大型語言模型與微調相關套件

- **PyTorch**：深度學習框架，負責模型載入與推論。
- **transformers（Hugging Face）**：載入 Breeze-7B-Instruct 模型、Tokenizer 以及 TextIteratorStreamer 等工具。
- **peft**：用於實作 LoRA（Low-Rank Adaptation）微調。
- **bitsandbytes**：支援 4-bit／8-bit 量化，降低模型推論所需記憶體。
- **flash-attn／FlashAttention2**：加速注意力運算以提升推論效能。

RAG 與向量資料庫相關套件

- **langchain-text-splitters**：使用 RecursiveCharacterTextSplitter 將課程文字資料切分為適當大小的段落。
- **langchain-chroma**：與 Chroma 向量資料庫整合，提供向量儲存與搜尋介面。
- **langchain-huggingface / HuggingFaceEmbeddings**：使用 BAAI/bge-small-zh-v1.5 等中文向量模型產生文本向量。

- ChromaDB：作為本地向量資料庫，儲存課程段落向量與相關 metadata。
- OpenCC：用於繁簡體中文轉換，讓資料與模型輸入在文字格式上保持一致。

語音辨識與語音合成相關套件

- Whisper：用於本地語音辨識，將使用者的錄音檔轉為文字。
- CosyVoice2：作為語音合成核心模型，負責將 LLM 產生的文字回覆轉換成自然語音；搭配官方或開源推論程式進行批次或分段合成。
- wave、numpy 等套件：處理 wav 檔案讀寫、音訊陣列運算、分段儲存與檔案命名管理。

Unity 端套件

- Live2D Cubism SDK for Unity：驅動 2D 虛擬人物的模型、表情與參數控制。
- TextMeshPro：顯示中文字幕，並支援多行與自動換行排版。

4.2 相關技術

4.2.1 語言模型與 LoRA 微調

本專題的大型語言模型使用聯發科的 Breeze-7B-Instruct 作為基礎語言模型，藉由指令微調能力，讓模型在回答課程相關問題時，更符合本專題設定的對話方式和語氣。考量到 7B 模型參數數量龐大，若使用全精度訓練將大幅占用 GPU 記憶體，因此我們先透過 bitsandbytes 以 4-bit 量化方式載入模型，並搭配 FlashAttention2 及 FP16 計算，使推論與訓練在單張顯示卡上仍能順利進行。

在微調方法上，我們採用 LoRA（Low-Rank Adaptation）技術，只在注意力與前饋網路中的關鍵線性層（如 q_proj、k_proj、v_proj、o_proj 及 up/down/gate_proj 等）加入低秩矩陣，設定適中的 rank、縮放係數與 dropout，讓可訓練參數僅佔原模型的一小部分。微調程式使用 PEFT 套件將這些 LoRA 層掛載到量化後的模型上，訓練時僅更新新增權重，既降低顯存需求，也讓日後能方便切換或疊加不同風格的 LoRA 模型。

訓練資料以 JSONL 方式儲存，每筆資料包含 system 提示、使用者問題與回覆三種訊息。程式會呼叫模型內建的 chat template 先將對話展開為模型習慣的格式，再在最後接上導覽員回答與結束標記。

為了讓模型專注學習「如何回答」，而不是背誦題目與系統提示，在建立 labels 時，僅對最後一段 assistant 回覆計算損失，其餘前綴 token 一律標記為 -100，使其在損失函數中被遮蔽，不參與後續的梯度反向傳播。

訓練過程中我們設定適中的最大序列長度與學習率，搭配梯度累積與 8-bit AdamW 優化器，完成數個 epoch 後輸出 LoRA 權重，並在部署階段與原始 Breeze-7B-Instruct 合併，形成本專題最終使用的模型。

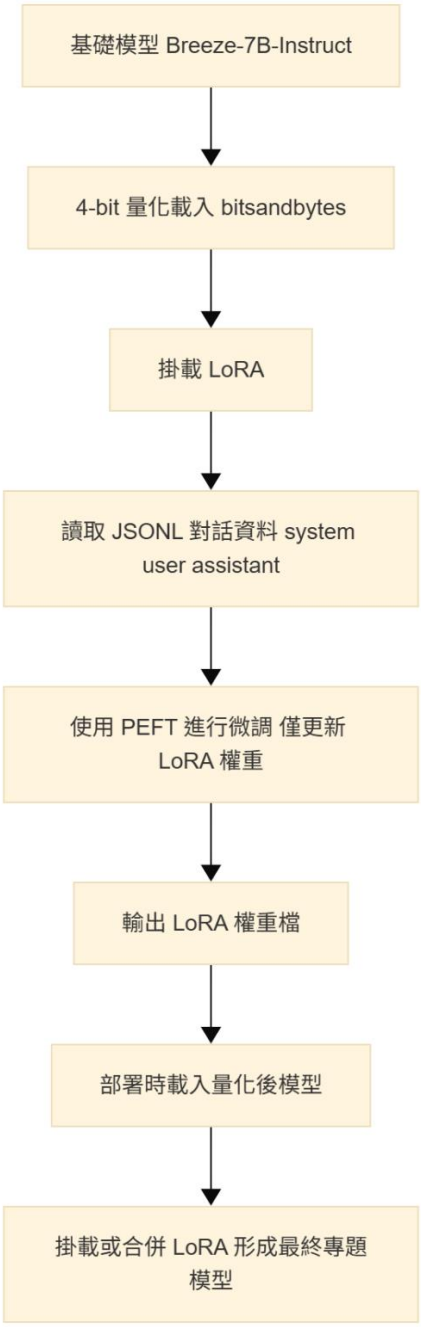

圖 4-1 語言模型與 LoRA 微調流程圖

4.2.2 RAG 與向量資料庫

為了讓 LLM 能回答課程相關問題，系統採用 RAG（Retrieval-Augmented Generation）架構。在建置資料庫的階段，會先掃描資料夾中的

課程 CSV 檔，依照句號與換行將長篇內容切成約四百字的段落，每一段都保留原始檔名作為 metadata，以便回答時標示資料來源。

在向量化階段，系統選用適合中文語意的 BGE 模型 BAAI/bge-small-zh-v1.5，搭配 HuggingFaceEmbeddings 將每個段落轉成高維向量，並使用 Chroma 建立向量資料庫。

系統進入向量檢索流程時，後端會向 Chroma 要求較大量的候選文件與距離分數，接著分別計算「字面重疊比例」與簡化 BM25 分數，字面重疊是將中文直接視為字元集合，比較問題與段落中共有多少字；BM25 則將中文每個字當成 token，統計文件長度與詞彙出現的頻率，計算出適合排序的權重。這兩種分數經過 min-max 正規化後，會與向量相似度依權重混合成 hybrid score。最終挑選出的多段文字會附在使用者問題後面，作為 Breeze 模型生成答案時的提示。

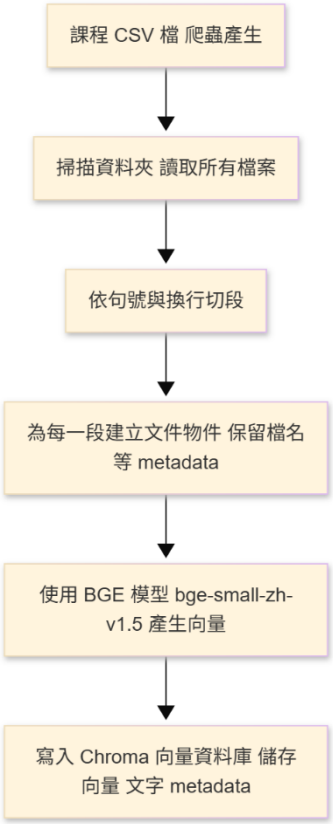

圖 4-2 RAG 資料建置流程圖

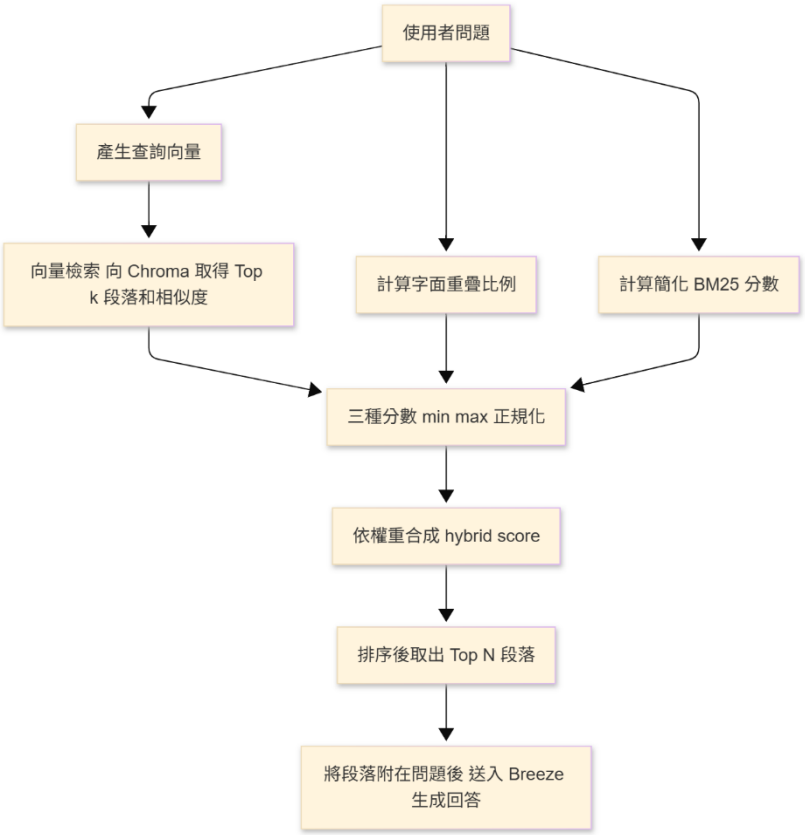

圖 4-3 RAG 查詢與 hybrid 排序流程

4.2.3 課程資料爬蟲

課程相關資料主要透過程式化爬蟲批次蒐集。執行時，程式會先根據指定的學期代碼，取得該學期所有開課的基本資訊，如選課代碼、課程名稱、開課系所與開課別等。接著，依照選課代碼逐門讀取課程教學綱要頁面，擷取課程名稱、授課教師、上課時間與教室、課程簡介、評分方式等欄位，並整理成 CSV 檔供後續 RAG 流程使用。

4.2.4 TTS 語音合成

本專題使用 CosyVoice2 作為語音合成模型。當 LLM 產生出最終回答後，會先經過分句，將過長段落切割成數個適合朗讀的短句，避免單一語音

檔過長，導致必須等待整段生成完畢才能開始播放。每一小段文字會依序送入 TTS，產生對應的語音檔與字幕內容。

合成過程中，同時執行輸入輸出多工處理，每當某一段語音完成，便立即將該段語音與字幕主動傳送到 Unity 端，而不是等全部段落都生成後才一次送出。這種「邊生成、邊傳送邊播放」的設計，讓實際體感等待時間遠短於整段語音生成的總耗時。

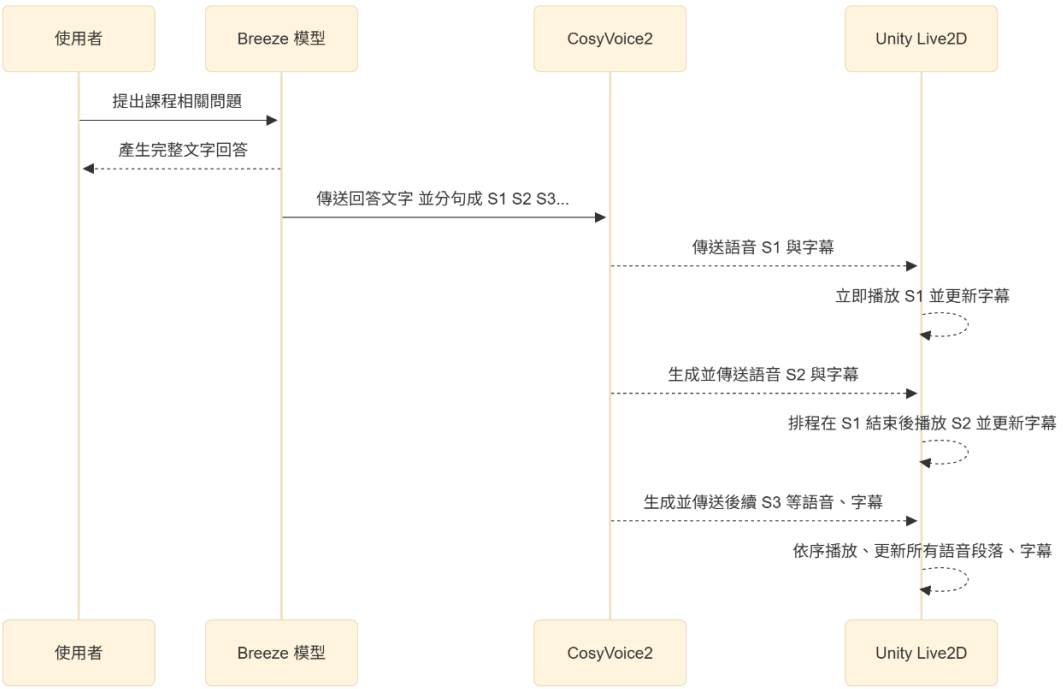

圖 4-4　TTS 分段合成與串流播放時序圖

4.2.5　Unity / Live2D 虛擬人物與 TTS 語音接收

本專題使用 Unity 承載虛擬角色與互動場景，並搭配 Live2D 實現角色表情與肢體動作。Unity 專案中包含兩個部分：一是負責接收後端主動送來語音與字幕檔案；二是將這些語音資料轉化為聲音播放與角色動作的 C#腳本。

TTS 語音接收在 Unity 內以一個常駐的 HTTP 伺服器元件形式存在，開始運行後會在指定的 prot 上監聽特定路徑，等待後端傳送語音資料。資料包含段

落識別用的基底名稱、一段語音檔與對應字幕文字(如圖 4-5 所示)。播放腳本定期檢查接收端，依檔名順序取出新到的語音與字幕，安排播放。

圖 4-5 段落識別用基底名稱、語音檔與對應字幕文字

虛擬角色的動態表現由一支 Live2D C#腳本控制。腳本啟動時會讀取模型中與嘴巴開闔、眼睛張合、眼球視線、身體姿態與頭髮搖晃相關的參數範圍，並建立多組內部狀態。當語音播放時，控制腳本會從音訊來源擷取一小段音訊波形，計算音量強度，經過帶有攻擊與釋放時間的包絡追蹤與平滑處理後，轉換成嘴型開合程度，對應到 Live2D 模型的口型參數，讓角色的嘴型隨語音節奏自然張合。

第五章　專題學習歷程介紹

5.1　專題相關技術學習介紹

（一）Python 爬蟲與資料前處理

在資料來源方面，本專題首先以 Python 為主要開發語言，學習撰寫爬蟲程式，從學校的課程查詢網站批次取得課程清單，最後統一整理成.csv。

透過對 HTTP 請求、查詢參數與回應格式的分析，我們實作出自動化的查詢流程，能依學期代碼逐一取得開課列表，再進一步進入各課程的餘額查詢與教學綱要頁面，擷取課程名稱、授課教師、上課時間與地點、課程簡介、評分方式與課程連結等欄位。

在資料整理階段，我們使用 csv. DictWriter 將擷取到的資訊統一整理為表格檔，並為缺漏欄位與異常值設計機制，避免單一筆資料錯誤導致整批轉檔失敗。這段實作讓我們更熟悉如何處理網頁資料，以及如何把原本分散在多個頁面的內容整合為後續可以使用的資料集。

（二）向量資料庫與 RAG 建置

有了 CSV 之後，我們編寫 build_rag_db.py，使用 pandas 讀入課程表，組合出包含課名、教師、時間地點、課程簡介、評分方式與課程連結的文字，作為之後向量化的基礎內容。

接著透過 RecursiveCharacterTextSplitter 依照句子與段落分段，設定 chunk_size、chunk_overlap 與分隔符號，將原始長文本拆成適合檢索的片段，並將來源檔名寫入 metadata["source"] 中。

在嵌入模型部分，我們使用 BAAI/bge-small-zh-v1.5 中文向量模型搭配 Chroma 建立本地向量資料庫，在實際查詢時,則由 rag_backend.py 負責載 入 Chroma 與 Embeddings,並實作一個「二階段」檢索流程:問題先經過 LLM 的關鍵字整理後先以 similarity_search with score 取得候選段落,再加上自製的字面重疊評分、BM25、向量分數正規化與 hybrid 加權,最後再選

擇分數接近第一名且筆數不超過門檻的文本組成 context。這段實作讓我們學習如何將資訊存入向量資料庫，以及以向量相似度、字面重疊比率、BM25 組成混合排序檢索。

(三)LoRA 微調與模型訓練流程

在語言模型部分,我們撰寫一套以 Breeze-7B-Instruct 為基座模型,建置「messages + chat template + assistant-only LOSS」的 LoRA 微調流程。透過 BitsAndBytesConfig 啟用 4-bit 量化，使 7B 模型能在有限 VRAM 中訓練，使用 tokenizer.apply_chat_template() 將 system／user／assistant 對話轉換對話格式，使用 gradient_accumulation_steps、gradient_checkpointing 與 8-bit 優化器 paged_adamw_8bit 在顯示卡可承受的前提下，盡量放大有效批次大小，以兼顧訓練穩定度與資源使用效率。最終我們將 LoRA adapter 儲存到獨立資料夾，並在推論端由主程式載入再 merge_and_unload，達成推論時只需單一合併後權重即可。

（五）Web 伺服器、狀態管理與語音辨識

為了讓使用者能透過 Web 技術使用本系統，我們撰寫 server.py 以 FastAPI 為核心，負責啟動與管理 LLM+TTS 主程式子行程（child process），並在啟動時預先載入；建立 /ws WebSocket ，維持前端客戶端連線；提供 /push API，接收主程式透過 HTTP 主動回報的 start／delta／final／done 事件，並統一轉換成 WebSocket 訊息送給前端。

另外，我們也整合本地 faster-whisper，在 Web 端點收錄使用者上傳的音檔，存成暫存檔後交給 Whisper 模型進行轉文字，最後再回傳 JSON 給前端。

5.2 專題製作過程遭遇的問題與解決方法

（一）顯示卡記憶體不足與效能調校

在載入 Breeze-7B 模型並加上 LoRA 權重時，最初常因為顯示卡記憶體不足而發生 CUDA OOM。尤其在嘗試同時開啟 FlashAttention2、全精度權重與較長的最大序列長度時，問題更加明顯。

為了在有限的 GPU 資源下完成訓練與推論，我們我們採取了以下做法：

- 訓練時使用 4-bit 量化載入基座模型，再搭配 LoRA 只針對部分層訓練；
- 調整 max_length 與對話歷史，只保留最近幾輪與 RAG context，避免一次塞太多 token；
- 透過 gradient_checkpointing、gradient_accumulation_steps 與較小的 per-device batch size 在訓練中節省記憶體；

（二）RAG 檢索品質與「查不到課」的問題

在 RAG 架構剛完成時，我們發現向量相似度搜尋不一定能滿足所有狀況：

- 查詢老師姓名時，模型無法確保一定抓到正確的教師，而可能選到名字相似但不同人的資料；
- 有些課程明明存在於資料表中，但因為切割方式或嵌入差異，搜尋結果卻未包含相關段落，導致系統錯誤宣稱「查無此課」。

為改善這些情況，我們做了以下調整：

- 對於明確包含教師姓名的問題，優先在原始表格中以字串比對尋找對應課程，再視需要補充向量檢索的結果；
- 在向量相似度之外，加入字面重疊與簡化 BM25 分數，將多種指標混合排序，避免完全依賴單一相似度；

- 設計保守策略：當所有候選結果的分數都偏低時，寧可明確回覆「目前資料中找不到符合條件的課程」，而不是勉強湊出看似合理但實際不存在的答案。

透過這些方法，RAG 回傳給語言模型的內容更接近使用者需求，也降低了誤導性的回答。

（三）語音合成延遲與檔案完整性問題

原本 TTS 流程是「LLM 一次輸出整段回答 → 一次丟給 CosyVoice 合成 → 完成後再播放與上傳」，導致使用者需要等完整音檔產生才能聽到聲音。為改善這些情況，我們我們改成分段合成與多執行緒在文字時先將回答切成多個短句，逐句送入語音合成模型後上傳。

調整後，使用者可以在系統還在生成後面內容時，就先聽到前幾句回答，整體體感等待時間大幅縮短，且播放過程更為穩定。

第六章　結論與未來展望

本專題結合大型語言模型（LLM）、檢索增強生成（RAG）、語音辨識（STT）與語音合成（TTS），以及 Unity Live2D 虛擬角色，實作一套課程介紹系統。透過課程爬蟲與向量資料庫建置，我們讓 LLM 能夠利用實際的課程資料回答與課程相關的問題，並透過 RAG 在回答前檢索內部資料庫，提高回覆的可靠性；同時藉由 TTS 與 Unity 虛擬角色的整合，將原本單調的文字回應轉成具體的語音與角色互動，提升系統的親和力與使用體驗。

在系統架構上，本專題將運算負載較高的 LLM、RAG、STT 與 TTS 集中在後端伺服器，前端則以瀏覽器與 Unity 客戶端兩種形式與後端連接。Unity 虛擬角色可以部署在與伺服器不同的電腦或手機上，只要能連上後端即可運作，充分展現出機動性部署的優點。這種設計模式未來也可以套用到其他應用，例如智慧諮詢櫃台、系所介紹或校慶活動導覽等，並可進一步擴充更多校園資訊來源、加入多語言支援，或改善模型效能與回覆品質。

參考文獻

〔1〕MediaTek Research, “Breeze-7B-Instruct-v1_0,” Hugging Face.

〔2〕E. Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” arXiv:2106.09685, 2021.

〔3〕Hugging Face, “PEFT: Parameter-Efficient Fine-Tuning,” Hugging Face Docs.

〔4〕BAAI, “bge-small-zh-v1.5,” Hugging Face.

〔5〕CosyVoice , “CosyVoice2: A Scalable Text-to-Speech System For Zero-Shot Speaker Adaptation,” 2024.

〔6〕P. Lewis, E. Perez, A. Piktus, et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

〔7〕J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, “BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation,” arXiv:2402.03216, 2024.

〔8〕JackCheang（2024）。〈建構簡易 RAG 系統：以 Hugging Face、LangChain、Chroma DB 和 Google Gemma 為例〉。Medium。