從零到一：用 AI 協作打造智能英文學習平台

以 Linker 為例（FastAPI + Gemini 雙模型 + 個人化複習）

Max Chen | GitHub：MaxChen228/Linker

2025 年 8 月 10 日

摘要

Linker 是一個為個人英文學習打造的智能系統：即時批改、錯誤分類、知識點追蹤與複習排程。本文件整合專案動機、設計決策與技術實作，並以圖解呈現「雙模型架構」與「知識管理」流程，同時附上部署與操作速查。全文可在 Overleaf 以 XeLaTeX 直接編譯。

目錄

1 為什麼做 LINKER？（痛點與目標）

痛點：寫句子缺乏道地度判斷、錯誤無即時與可追蹤回饋。
目標：即時批改 + 精準分析 + 個人化追蹤，不只標記對錯，更要說明「為什麼」和「如何更好」。

設計原則

- **快速驗證**：先 CLI 後 Web，從最小可行功能起步（題目生成＋批改）。
- **成本／品質兼顧**：採 **雙模型**——*Flash* 出題、*Pro* 批改，兼顧速度與準確。
- **可維護的 UI**：以設計令牌（Design Tokens）統一色彩、間距與字級，支援響應式。

2 架構與流程（From CLI to Web）

2.1 模組演進

- **CLI 驗證**：`linker_cli.py`
- **Web 應用**：`web/main.py`（FastAPI + Jinja2）：練習、知識點、文法頁。
- **核心模組**：`core/ai_service.py`（雙模型）、`core/knowledge.py`（知識點管理）、`core/error_types.py`（錯誤分類）、`core/logger.py`（集中日誌）。

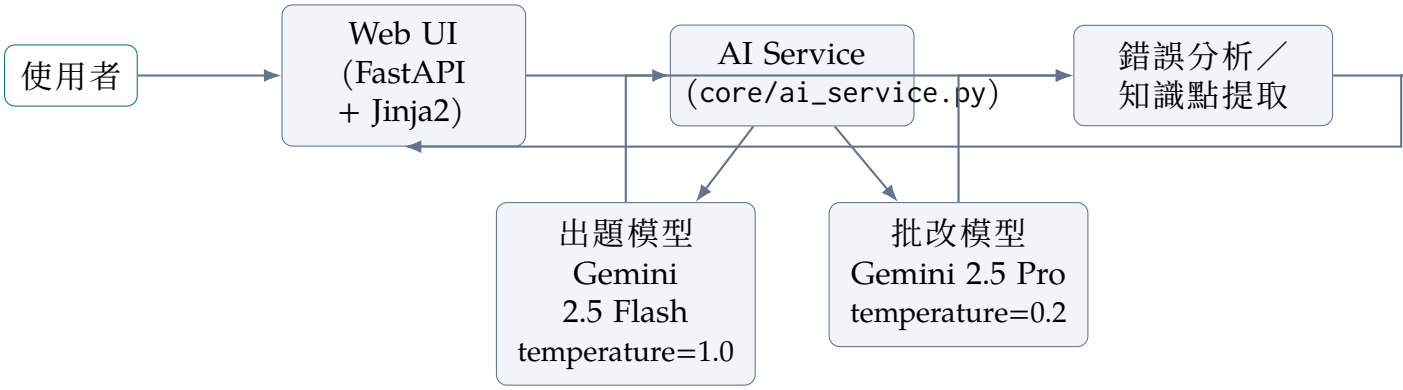

圖 1: LINKER：雙模型與知識管理資料流示意

3 AI 雙模型策略（成本 × 品質）

策略：出題使用 **Gemini 2.5 Flash**（快、創意、低成本），批改使用 **Gemini 2.5 Pro**（準確、專業、高品質），並調整溫度與取樣參數以平衡多樣性與穩定性。

3.1 核心程式片段（AI Service）

程式碼 1: 雙模型初始化與參數（摘錄）

```
# 出題模型（快速）- 提高溫度以增加變化
self.generate_model = genai.GenerativeModel(
    self.generate_model_name,
    generation_config=genai.GenerationConfig(
        response_mime_type="application/json",
        temperature=1.0,
        top_p=0.95,
        top_k=40,
    ),
)

# 批改模型（高品質）
self.grade_model = genai.GenerativeModel(
    self.grade_model_name,
    generation_config=genai.GenerationConfig(
        response_mime_type="application/json",
        temperature=0.2,
        top_p=0.9,
    ),
)
```

4 知識點管理（錯誤類型 × 間隔重複）

4.1 四級錯誤分類

類別	說明與示例
系統性（Systematic）	可規則化避免的錯誤：時態、主被動、片語動詞用法等。
單一性（Isolated）	需個別記憶：固定搭配、片語、慣用語。
可以更好（Enhancement）	文法正確但不地道：自然度、語域、用字更精準。
其他（Other）	漏譯、誤解、語意錯置等。

表 1: 錯誤分類設計

4.2 複習排程（艾賓浩斯導向）

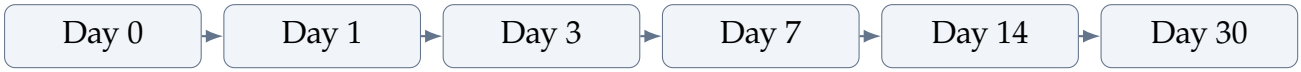

圖 2: 知識點複習時間點（示意，實際依掌握度動態調整）

5 前端設計系統（Design Tokens）

以設計令牌統一色彩、間距與字級，讓樣式跨頁一致、易擴充且利於重構。響應式布局支援桌機與行動。

好處：一致性、可維護、可替換品牌色； **做法**：集中於 `web/static/css/design-system/`，頁面僅引用語意化類別。

6 成果與介面

6.1 核心頁面

- 練習：新題／複習、即時批改與解說。
- 知識點：卡片化瀏覽、依錯誤類型與掌握度篩選。
- 文法句型：內建查詢與示例。

7 部署與啟動（速查）

7.1 本機 Web

程式碼 2: 啟動服務（Linux/Mac）

```
# 1) 設定 API Key
export GEMINI_API_KEY="your-api-key"

# 2) 啟動服務
./run.sh

# 3) 瀏覽器開啟
# 注意：以下 URL 會正確顯示且可複製
# http://localhost:8000
```

7.2 Docker

程式碼 3: Docker 啟動

```
docker-compose up -d
# 之後訪問：
# http://localhost:8000
```

7.3 雲端部署（Render）

程式碼 4: Render 部署流程（摘要）

```
# 1) Fork 專案並連接 GitHub 倉庫
```

```
# 2) 在 Render 設定環境變數 GEMINI_API_KEY
# 3) 觸發部署完成後，即可對外提供服務
```

8 反思與發展路線圖

8.1 經驗

- 與 AI 協作要明確需求、質疑並共創：快速原型 → 針對性優化。
- 設計系統讓前端在功能擴張時保持穩定與一致。
- 後端以事件日誌與知識點資料結構貫穿學習閉環，可量化成效。

8.2 Roadmap

- **短期**：帳號系統與跨裝置同步。
- **中期**：JSON → SQLite 遷移，增強查詢與效能。
- **長期**：社群共學、分享知識點與筆記。

附錄 A：環境變數（示例）

```
# AI 模型設定
GEMINI_API_KEY=your-api-key
GEMINI_GENERATE_MODEL=gemini-2.5-flash
GEMINI_GRADE_MODEL=gemini-2.5-pro
# 日誌
LOG_LEVEL=INFO
LOG_TO_FILE=true
```