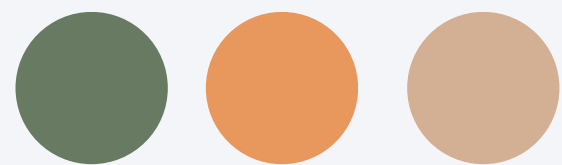


專題報告

11/14

b1228006張鴻毅
b1228022梁釗豪
b1228029蘇琪文
b1228032張紋菱



內容



網站製作



Git使用



資料前處理

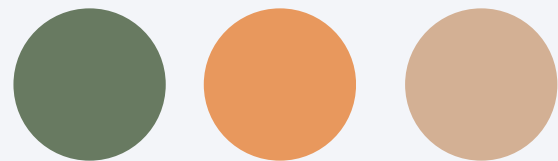
網站製作

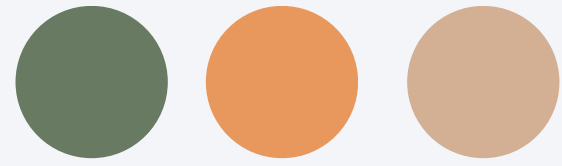
GitHub

https://marcoleung052.github.io/NursingRecordCompletion_train/step7/%E8%AD%B7%E7%90%86%E7%B4%80%E9%8C%84%E7%B3%BB%E7%B5%B1demo1.html

Apps Script

https://script.google.com/macros/s/AKfycbwDy-K2fpkm1knIKzbgSCmbmAVHsNKsbtQ2QVR_x71O7xEcg706GpK2OM8Oz4KFxL_RCw/exec





核心組件



前端

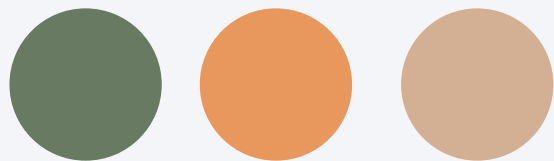


後端

前端

HTML / JS / CSS

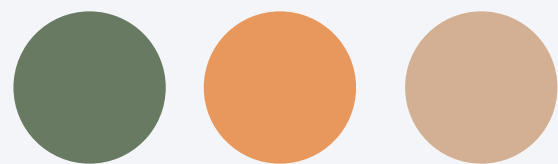
**負責用戶界面 (UI)、Copilot 觸發、以及呼叫 API。
這是用戶直接互動的部分。**

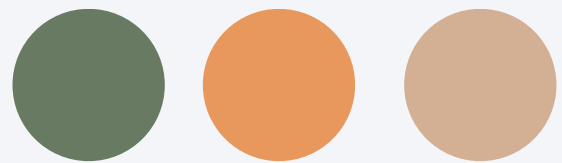


後端

Python / FastAPI / GPT-2

**負責運行 AI 模型 (GPT-2)、生成文本、
並透過 API 接口提供服務。**

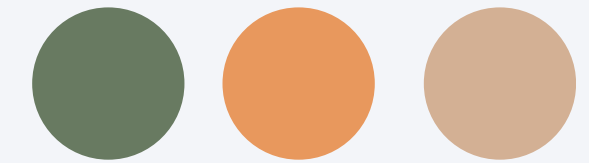




如何將 PYTHON AI 模型伺服器
與前端網頁結合，使其在網際
網路上 24/7 運行？



步驟一：部署 AI 後端



- 選擇平台

我們選擇了 Hugging Face Spaces，它專為託管 AI/ML 模型設計。

- 部署方式

使用 Docker 進行部署，以確保 Python、PyTorch 和所有依賴項的環境一致性。

- 檔案

我們準備了 `api_server.py` (應用程式), `requirements.txt` (依賴) 和 `Dockerfile` (環境藍圖)。

- 硬體選擇

部署在 CPU 實例上，並確認模型 (GPT-2) 成功加載。

步驟二：部署前端介面



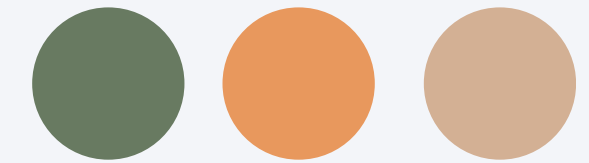
平台：GitHub Pages

我們選擇了 GitHub Pages 來託管靜態前端 (HTML, JS)。

- 免費且快速
- 與 Git 倉庫完美整合
- 輕鬆發布和更新

我們也處理了將網站放在專案子資料夾中需要注意的 CORS 配置問題。

步驟三：打通前後端連線



- **CORS 政策**

在 `api_server.py` 中更新 `origins` 列表，明確允許 `github.io` 域名的存取。

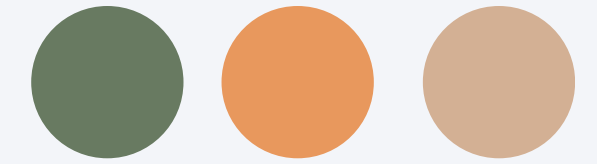
- **API 路由修正**

修正 `model_api.js` 中的 URL，從 `/predict` 改為後端真實的 `/api/predict` 路由。

- **JSON 格式**

修正 `model_api.js`，使其正確解析後端返回的 `{"completions": [...]}` 鍵值。

最終成功架構

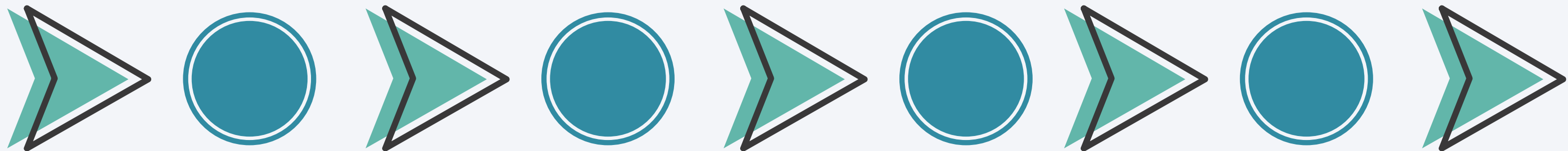


前端

model_api.js 發送
fetch 請求

AI 回應

返回 {"completions":
[...]} JSON 資料



用戶

訪問 GitHub Pages
網址並輸入文字

後端

FastAPI 接收 /api/predict
請求，GPT-2 生成文本

Git使用

基本操作指令

專案開始

git init (初始化專案)

git clone <remote-url> (從遠端下載專案)

指令	用途	功能說明
git add <file>	新增修改到暫存區	將修改的檔案從工作區加入暫存區。
git commit -m "message"	提交更改	將暫存區內容提交至版本歷史中，需撰寫有意義的訊息。
git status	查看狀態	檢查目前修改、暫存與未追蹤的檔案狀況。
git push origin <branch>	推送到遠端	將本地分支的提交推送到遠端儲存庫。
git pull	拉取遠端更新	從遠端儲存庫下載最新修改並合併到本地分支。

Git使用

分支操作指令

指令	主要用途	功能說明	特點 / 注意事項
git switch	切換或建立分支	切換到指定分支，或使用 -c 建立新分支	專注於分支操作，語意清晰、安全性較高
git checkout	切換分支、版本或單一檔案	可切換分支、回到特定 commit、還原單一檔案	功能多但複雜，容易誤操作導致非預期結果
git merge	合併分支	將其他分支的修改內容合併到當前分支	保留兩條分支的提交歷史，可能產生 merge commit
git rebase	重整提交歷史（變基）	將一個分支的提交重新套用到另一分支上	使歷史線性整潔，但需謹慎使用（避免改動公共分支歷史）

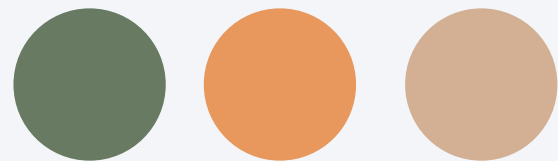
Git使用

版本回復與還原指令

指令	功能說明	特點
git revert	建立一個新的提交，用以反轉指定提交的修改。	保留完整歷史、安全可追蹤。
git reset [mode]	將版本退回到指定提交。	可刪除後續提交。
--soft	保留工作區與暫存區內容。	適合重新提交。
--mixed	保留工作區內容但清除暫存區。	預設模式。
--hard	清除所有修改與暫存。	不可逆，危險操作。

資料前處理

- 多個 Excel 檔案，需批次讀取。
- 跨行儲存格
- 多病患分隔使用「長庚醫療財團法人」分隔。
- 詞彙不統一(病人/病患...)

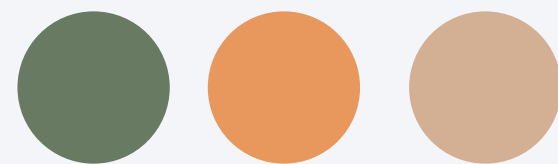


資料前處理

- 1. 讀取與合併將：`HH:MM` 的跨行儲存格合併為單一紀錄。
- 2. 切割與正規化
- 3. Tokenization

使用模型原生的 Tokenizer，並為 CausalLM 任務準備 `labels`

- 4. 儲存與載入



CausalLM

- 使用模型原生 Tokenizer: 採用 `Breeze-7B-Base` 的 Tokenizer，確保輸入一致。
- Labels = Input IDs: 模型要學習預測自己的輸入。
- Padding Masking: 將 `pad_token` 對應的 `labels` 設為 `-100`，使模型專注於學習真實文本，忽略填充符號。

