TalentLinkerAI - Auto Scrapy Job & Matching Resume Using NLP

1 職缺資料爬取模型

目標

自動化從各大求職網站收集職缺資訊，提取所需技能、工作經驗、教育要求等關鍵特徵。

技術

- **網路爬蟲 (Web Scraping)**：使用 Scrapy 或 BeautifulSoup 自動抓取職缺網站數據，對於動態🄵容則可考慮 Selenium。
- **資訊抽取 (Information Extraction)**：利用自然語言處理 (NLP) 技術（如 Named Entity Recognition，NER），自動辨識🄵提取「技能」、「職位」、「公司」等重要資訊。可以使用 SpaCy 或 BERT 等預訓練模型。
- **資料處理**：清洗🄵標准化抽取到的數據，便於後續媒合。

2 履歷 OCR 檢測模型

目標

將用🄵上傳的 PDF 或圖片格式的履歷表轉🄵🄵可讀取的文字。

技術

- **OCR 引擎**：推薦使用 Tesseract OCR 或 Google Vision API、Amazon Textract 等商業解🄵方案處理不同格式與分辨率的履歷。
- **自然語言處理**：對 OCR 結果進行清洗，移除頁面分隔符及格式轉🄵錯誤。
- **特徵提取**：提取姓名、技能、經歷、學歷等資訊，結合 NER 模型實現自動化標記與提取。

3 職缺與履歷媒合模型

目標

通過分析職缺需求與候選人履歷的特徵，評估兩者的匹配程度。

技術

- **文本相似度模型**：使用 BERT、RoBERTa 或 Sentence-BERT 衡量職缺與履歷的文本相似度。
- **特徵工程**：設計特徵（如技能相似度、工作經驗匹配度等），將兩者轉🄵🄵向量後計算匹配分數。
- **推薦系統**：使用余弦相似度或基於深度學習的匹配模型，根據分數推薦職缺。

4 職缺資料爬取步驟

4.1 確認目標網站🄵獲取資料許可

- 選擇主要求職網站（如 LinkedIn、Indeed、104），確認職缺數據是否可合法抓取。
- 了解網站結構：觀察 URL 規則與 HTML 結構，確認頁面中的關鍵資訊（如公司名稱、工作職稱、技能需求）。

4.2 建立網路爬蟲框架

- 使用 Python 的 Scrapy 設計高效數據抓取框架，或使用 BeautifulSoup 處理小型網頁。
- 若網站使用動態加載，則🄵用 Selenium 模擬用🄵行🄵抓取數據。

4.3 提取所需數據

- 設置代碼提取特定職缺資訊（如職位名稱、公司名稱、技能需求、薪資、地點）。
- 清理數據：移除 HTML 標🄵、🄵除重🄵項目與多余空白字符。

4.4 儲存數據

- 將數據保存🄵 CSV 或 JSON 格式，便於後續處理。
- 設計良好數據結構（如欄位 `job_title`, `company`, `skills_required`, `location`, `salary`），確保一致性。

4.5 初步測試與資料質量檢查

- 測試爬蟲對多個網站與職缺頁面的效果，確保穩定抓取數據。
- 處理錯誤：加入網路超時、網站反爬措施的處理邏輯。