專題報告

資訊專題 Lawshield.ai

Python / LLM / RAG (llamaIndex) / elasticsearch

專案成員與教授簽章

指導教授 : 國立政治大學資訊科學系 蔡銘峰 教授

- 國立政治大學資訊科學系 陳肇廷
- 國立政治大學資訊科學系 梁栴睿
- 國立政治大學資訊科學系 蘇胤翔
- 國立政治大學資訊科學系 王冠智

摘要

「Lawshield.ai」專案是一個專為司法領域設計的 AI 法律諮詢平台。本系統運用 Llama 3.1 大型語言模型與 llamaindex - RAG 框架（Retrieval-Augmented Generation）來實現高效的數據檢索和文本生成，確保提供精確的法律建議。在本專案中，我們選用交通事故處理的諮詢服務為開發主軸，以國內公開的法律條文和法院判決書為基礎資料，並進行針對性優化，使其能快速且準確地回應使用者的諮詢需求。

各項開發技術

- **資料集存取及配置 <主要負責人: 梁栴睿、蘇胤翔>**
  - 向量資料庫設計 - 專案中使用 Elasticsearch 作為向量資料庫，善用其特色功能進行高效檢索。我們設計的 RAG pipeline 能根據使用者選擇的諮詢需求，透過 index pattern 與 alias 提取最適合的資料集。在 Elasticsearch 的資料配置上，我們設計特定命名規則以搭配使用 index 標頭查詢功能（如「traffic-*」模式），並不同年份與領域的法律文件分門別類，使系統在歸納大數據資料時能將資料存取與檢索做到最精確的狀態。

- **RAG 系統實作與系統架構設計 <主要負責人: 陳肇廷、梁栴睿、蘇胤翔>**
  - RAG 核心 - RAG 的核心包括 embedding 生成、vector store index 的設計、data retrieve 的實作、chat engine route 與 query response 的處理。該系統整合了多種資料來源，並根據使用者問題進行最佳匹配檢索，優化了查詢結果的相關性。
  - 多模型Route設計 - 為提升模型的精確性與專業性，針對 Llama 3.1 模型進行微調（fine-tuning），以公開的法條文本問答與大量新聞案例作為資料集，優化該模型在法律諮詢場景中的表現。並客製化 fine tuning的資料集，將 Llama 3.1 微調成不同route階段的任務模型，以賦予模型足夠的專一性
  - 多層檢索生成設計 - 我們重新設計 LlamaIndex 的聊天引擎，透過規範制式化的 prompt template 格式設計(如: 生成 boolean)，對於是否進行 RAG 檢索做判斷，並根據 retriever 的提取結果擷取相關性最高的 node。再經由客戶端提供的 context 與系統設定的多項 prompt template 重新配置，使用多模型生成策略，根據不同的模型專業領域分配不同生成任務，最終通過負責總結的模型整合並生成最終回答回傳客戶端，大大提升回應的專業度與準確性。

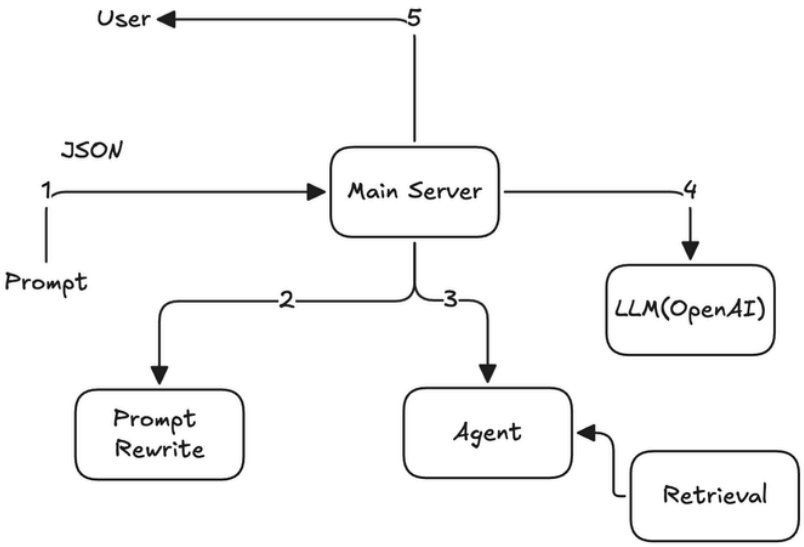
專題報告

資訊專題 Lawshield.ai

Python / LLM / RAG (llamaIndex) / elasticsearch

各項開發技術

- **微調基底模型及準備資料集 <主要負責人: 陳肇廷、王冠智、蘇胤翔>**
  - 大型語言模型 - 為了提升法律相關問答的準確度，我們選擇 LLaMA 3.1 8B 作為基底模型，並針對《道路交通管理處罰條例》等法律進行微調與深度訓練。在 fine tuning 過程中，我們經過測試發現，使用 4-bit 以上量化或 FP16 的模型表現最為穩定且準確。此外，我們以<大模型教小模型>的方式，透過 GPT-4 生成新聞事件問題，並利用 RAG 系統檢索正確答案，再將這些數據用於訓練較小的 LLaMA 3.1 8B 模型，讓模型的回答更貼近日常用語且更加熟悉法律內容，也同時更加能夠針對不同的問題進行回答。
  - 資料集設計 - 讓模型有效學習新知識的關鍵在於具備專業知識的問答資料集。首先，我們對各類道路交通新聞事件及相關判決書進行分類，並模擬使用者詢問頻率最高的問題，設定了四個核心子類別：**事故處理**、**損害賠償**、**肇事責任分析** 及 **罰則**。根據這四個子類別，我們生成了專門用於微調 LLaMA 3.1 8B 的問答資料集，除了這些新聞資料，我們還製作了《道路交通管理處罰條例》、《道路交通安全規則》、《民法》及《刑法》的四個問答資料集以作為模型的基礎知識。



系統架構元件

- **系統採MVC架構設計 <主要負責人: 陳肇廷、梁栴睿、蘇胤翔>**
  - Main Server（Controller） - 是整個系統的核心控制器，負責處理服務之間的通訊，並將使用者的輸入傳送至適當的模型或檢索服務。它同時負責調度不同的 LLM 模型（例如 OpenAI 或微調後的 LLama3.1），依照需求對外部模型進行多次查詢，並根據回應結果進行判斷與整合。
  - Prompt Rewrite Service 提示覆寫服務（Model） - 該服務專門收錄大量與交通事故相關的俗語與日常用語，並利用經過微調的模型重新生成提示，確保 LLM 能更好地理解與處理這些複雜的提問。這一過程會將使用者的問題進行覆寫，以生成適合進一步檢索與生成的語句，提升生成內容的準確度。
  - Agent - 服務負責將改寫後的提示轉發至檢索系統或外部 LLM，並根據檢索結果生成最終回應。這個模組還包括多層檢索功能，根據不同資料來源（例如交通事故新聞、判決案例、法律條文等）進行匹配與提取，確保回應的專業性。

專題報告

資訊專題 Lawshield.ai

Python / LLM / RAG (llamaIndex) / elasticsearch

系統架構元件

- **系統採 MVC 架構設計 <主要負責人: 陳肇廷、梁栴睿、蘇胤翔>**
  - Retrieval - 檢索服務使用預訓練的 Embedding Model，生成向量並在設計好的 vector store index 中進行資料檢索。根據使用者的查詢內容，該服務會提取最相關的節點（text node），以提高回應的精確性。檢索系統的多層設計能夠區分不同的法律資料，進而進行針對性的資料提取。
  - 分散式系統架構設計 - 為了系統的穩定性與可擴展性，我們使用 Kafka 進行系統中各組件的串接，利用 Kafka 的高可用性、高擴展性、高吞吐量的特性，讓我們的系統可以在即時回應用戶的同時，有條不紊的處理每一則使用者的請求
  - 使用者介面 - 使用 Streamlit 開發 Client 端介面，使使用者除了可以與我們的系統互動，並接續之前的對話內容，還可以查看先前聊過的其他話題，有效率完成不同的問題

- **系統流程 <主要負責人: 陳肇廷、梁栴睿、王冠智>**
  - 使用者提交問題：使用者透過前端界面（使用 Streamlit 開發的 Client 端）提交查詢，系統接收到問題後將其傳遞至 Main Server。
  - 提示覆寫（Prompt Rewrite）：主伺服器將問題轉發至 Prompt Rewrite 服務，該服務根據交通事故相關的通俗詞彙進行更改，重新覆寫，確保問題以 LLM 能理解的形式生成。
  - Retrieval：覆寫後的問題進一步傳遞至檢索系統，檢索系統根據改寫後的 prompt 查詢多個index，整理相關性最高的提取文本，並將結果回傳至 Agent 服務。
  - 多模型生成：根據檢索結果，Agent 將問題傳送至不同的 LLM 模型進行生成，並根據模型專業領域進行任務分配，最終由負責總結的模型整合回應。
  - 回應傳回使用者：系統生成的回應透過 Main Server 回傳至 Client，呈現給使用者。

SIDE PROJECT

- **LINE BOT開發 <主要負責人: 蘇胤翔、梁栴睿 >**
  - 官方帳號服務 - 開設官方帳號並將既有的 UI 服務整合到 LINE 聊天室，讓熟悉 LINE 的使用者多一份選擇可以與 Lawshield.ai 對話並獲得有如真實律師或法律顧問的法律協助
  - Rich Menu 設計 - 將平台服務功能整合到 LINE 提供的 Rich Menu 上，讓使用者在聊天室更容易操作平台功能、針對不同的問題進行詢問，也簡化系統的複雜程度(將 LINE mini app 作為 UI 使用，不需要自行維護前端 server)

- **網頁爬蟲 <主要負責人: 蘇胤翔、梁栴睿 >**
  - 每日新聞追蹤爬蟲 - 利用 selenium 每日定時搜索多個網站當日新聞資料，擷取重點後自動上傳 google drive 作為儲存空間使用
  - 相關法律文件更新爬取 - 架設一執行實體每週定時自動檢查相關網頁變化，若有文件更新則立即下載或利用request爬取
  - 法律判決書爬蟲 - 利用 puppeteer 全自動抓取司法院判決書系統上記載的判決書，從系統開始有紀錄的 85 年一直爬到 113 年中，獲得了超過 150 萬篇的判決書