Problem Statement and Task Definition:

我們的專題是：《LLM 輔助論文綜整與檢索》

因為在找適合的論文閱讀時，我們常常需要一一檢視一份論文的 abstract, introduction, conclusion 等等內容，非常花時間。就算讓 GPT-4 幫忙略讀，也需要一份份地下載 PDF，再丟給他解析。更何況很多人沒有購買 GPT-4 的使用權，常用的 GPT-3.5, Copilot 都沒辦法解析 PDF，GPT-3.5 甚至只有到 2021 年的資料庫。

因此，我們要製作論文整理綜整與檢索的系統，收集論文並解析，再將解析內容用來檢索出符合使用者需求敘述的論文，最後以 Bot 或 Web 的方式製作使用者介面，與使用者互動及呈現資料。

Description of the challenges:

這個系統主要解決的挑戰有 4 個：1. 資料收集、2. 論文解析、3. 論文檢索、4. 使用者介面。我們將透過爬蟲收集論文資料，接著以 python 套件解析 PDF 等檔案，再用 LLM 模型來解析論文，得到我們要的資料並儲存，之後同樣使用 LLM 根據儲存資料，接收使用者要求並檢索適合的論文，最後用 discord bot 或網頁的方式與使用者互動。

Input/Output Behavior with Concrete Example:

舉一個具體的例子：我們先透過爬蟲收集了 CVPR 2015~2023 的論文資料。接著使用者對於 Bot 或是網頁的輸入處，輸入「如何讓 CNN 能增加更多層數以增進效果？」，系統將會參考我們建立的資料庫以及 GPT pre-train 的資料，回應給使用者 2016 年在 CVPR 發表的”Deep Residual Learning for Image Recognition” 及更多相關論文，底下附上論文的簡介: ”殘差連接…”、tag: “圖像、殘差、梯度消失…”等等資料、其他論文推薦: ” Improved Residual Networks for Image and Video Recognition、…”。

Related works:

與這個 project 相關的系統有: $Elicit$[1], R − discovery[2], Consensus[3]等等。

Elicit: 關鍵字搜尋，摘要的摘要，英文介面

R-discovery: mobile App，關鍵字搜尋，推薦相關論文，英文介面

Consensus: 提問式搜尋，摘要的摘要，推薦相關論文，品質 tag，英文介面

我們的系統比較接近於 Consensus。在搜尋部分，一樣使用提問式搜尋論文，比起 Elicit 與 R-discovery 更加人性化，也比較方便使用者精準描述。在摘要部分，使用整篇論文取得摘要的結果，而非像 Elicit, Consensus 取摘要的摘要，能讓使用者快速看到更全面的內容。在相關論文推薦部分，使用 LLM 幫忙推薦，因為不知道其他系統的實際推薦方式，所以無法比較。在 tag 部分，不同於 consensus 的 tag 標註論文是否來自頂會、被大量引用，我們的 tag 標註了關鍵字、關鍵內容，著重輔助使用者了解論文。最後，在介面部分，我們期望中英文皆可使用，並且生成內容所使用的語言是使用者預先決定後，由 LLM 模型所生成。

Methodology:

首先，我們採用 python 撰寫，因為 python 有豐富的 LLM 相關函式庫及其他套件。

在資料取得部分，因為爬蟲泛用性高，可以用在各種網站上，所以採用爬蟲方法，使用 Beautiful soup 等常用爬蟲套件進行，在各大論文網站，以及 Google Scholar 上做搜索。或是使用論文網站的 API 來獲取資料，以避免爬蟲套件在解析 html 所花費的時間。解析 PDF 則用 PyPDF2 等套件進行。

取得資料後，由 GPT、Lamma 等 API 來做解析論文中的自然語言數據，其中，我們會嘗試不同 Prompt Engineering，使其產生我們資料及格式更精準符合需求。

取得解析數據後，將其儲存，並由 LangChain 套件做到讓 LLM 模型參照處理後的資料，搭配 Prompt Engineering、讓 LLM 模型自動生成追加問題等方法，做出輸入輸出。雖然這些方法可能會需要讓使用者輸入更多數據，造成花費時間較多，但是可以釐清使用者本質上的需求，並且得到更精準的結果。

得到結果之後，我們會利用 Discord Bot 或者是 Django 作為後端的網頁來呈現使用者介面。Discord Bot 在實作上比較簡便，可以縮短我們的開發週期，網頁則需要做前後端的開發，會需要比較長時間，但能得到比較好的呈現結果，實際怎麼做由專案狀況決定。

Evaluation Metrics:

LMSYS Chatbot Arena 是一個用來評估現在 LLM 模型表現的重要排行，而他們的評估方式為，使用多個模型盲測，藉由讓使用者評分的方式來評估 model 好壞，透過這項方法可以更貼近使用者所需及所想，而不單純只是照著舊有資料做出評估，所以我們也打算採用這方式作為標準。

評估方式為: 設計類似於 LMSYS Chatbot Arena 的系統，讓受測者測試並得到回饋，針對每個提問，讓 model 推薦 10 篇 paper，由受測者評估這 10 篇的相關度。

評估指標分數定義:
使用者看過摘要後回答:
1. 搜索到的論文，有幾篇與要解決的問題相關
2. 搜索到的論文，有幾篇可以應用在你要解決的問題上
分數評估 = 問題 1 的比例*0.25 + 問題 2 的比例*0.75

Baseline

我們將 google scholar, consensus 的表現作為 baseline，在同樣的第一筆輸入下，取前十筆資料做評分，期望系統可以超越 google scholar，接近於 consensus。

Time Schedule:

1. 完成爬蟲以及網站 API 串接，做到資料收集
2. 解析 PDF 並串接 LLM API 讓其解析資料
3. 將資料進一步儲存並交給 LLM model 來做搜索
4. 實驗不同的 Prompt Engineering 技巧及不同 model 的成效
5. 建立 Discord Bot 或是網頁
6. 部屬 project 到網路上

Discussion:

Discussion Board on hackmd:

https://hackmd.io/@9UfXRWPzS62YaxdR7uwZ0Q/ryxC1N_gC

Repo:

Github repo: https://github.com/NYCU-AI-intro/Intro-AI-Final-Project