# Final Project Team3


學生:

A1105505 林彧頎
A1105507 蘇柏諺
A1105521 黎子崴
A1105523 巫柔筠
A1105524 吳雨宣

教授:

黃健峯 教授

# Table of Contents

# 1. Problem1

(1) 方法 1:

使用 Together API 進行 Prompt 分析，自動從使用者的自然語言輸入中提取查詢條件（起始日期、結束日期、查詢項目），基於所提取的關鍵數據，調用適當的爬蟲邏輯，抓取臺灣期貨交易所網站上的目標數據，最後將抓取的數據進行格式化處理，並輸出到 Excel，供後續查詢和分析。

1. 使用 Together API 進行 Prompt 分析:

在這部分我們手動修改 prompt，多次嘗試後才找到提取的關鍵數據最精準的 prompt，以下是 prompt 的優化過程。

```
prompt = f"""
Extract the start date (`start_date`) and end date (`end_date`) in the format YYYY/MM/DD.
Extract the relevant data items mentioned.
Do not provide any explanations or extra information.
Input: {user_input}
"""
```

Prompt 版本 1

```
prompt = f"""
Your task is to extract details from the user's input. Specifically:
1. Extract the start date (`start_date`) and end date (`end_date`) in the format YYYY/MM/DD.
   - The start date should be the earliest date mentioned.
   - The end date should be the latest date mentioned or default to today's date if not provided.
2. Extract the relevant data items mentioned (e.g., "Foreign Long Positions", "Dealer Net Positions", etc.).

Example input:
"Retrieve the open interest data for foreign investors' long positions from 2024/12/01 to 2024/12/15."

Expected output:
start_date:2024/12/01
end_date:2024/12/15
data:[Foreign Long Positions]

Input: {user_input}

Provide the output in the exact format as the example above, without additional explanations:
"""
```

Prompt 版本 2

```
prompt = f"""
    從以下輸入中提取細節：
    1. 找出起始日期（`start_date`）和結束日期（`end_date`），格式為 YYYY/MM/DD。
    - 起始日期為輸入中提到的最早日期。
    - 結束日期為輸入中提到的最晚日期，若未提到則默認為今天的日期。
    2. 找出輸入中提到的查詢項目（例如：「外資多單」、「自營商淨多單」）。

    範例輸入：
    「從 2024/12/01 到 2024/12/15，取得外資多單的未平倉數據。」

    期望的輸出格式：
    start_date:2024/12/01
    end_date:2024/12/15
    data:[外資多單]

    輸入：{user_input}

    請按照上述格式輸出結果(並且不要做多餘的解釋)：
"""
```

Prompt 版本 3

但因為此方法並非全自動生程爬蟲程式並執行，prompt 的部分僅用於分析使用者的輸入，因此我們更新了後續的幾種方法。

(2) 方法 2:

採用全自動化提示工程技術來完成日盤與夜盤數據的爬取、處理和合併，並自動化迭代校正生成程式碼。

1. 動態提示生成:

   根據不同的上下文(目標網址、數據抓取目標及錯誤訊息)，動態生成提示內容。

   在程式中的 generate_crawler_code 函數會根據輸入參數生成適合的爬蟲程式碼提示，重點包括：

   目標網址(日盤、夜盤的 URL)、數據抓取目標（如表格中的「未平倉餘額」)、附加說明（如日期參數）、錯誤訊息回傳(將執行時的錯誤訊息提供給 Gemini，以自動調整生成的程式碼)。

```
prompt = f"""
網址是：{url}
請撰寫 Python 爬蟲代碼，使用 BeautifulSoup 提取 "{target}" 的內容。
{additional_instruction}
請直接提供完整的 Python 代碼，不需要多餘的描述。

另外這是之前的code與錯誤的說明:
{error_code}
"""
```

2. 多次迭代校正:

   在執行爬蟲程式碼時，若發生錯誤，程式會捕捉錯誤訊息，並自動傳回 Gemini 作為下一輪生成提示的基礎，實現完全自動化的程式碼改進過程。

3. 模組化爬取日夜盤數據:

   日夜盤分開爬取後，進行資料驗證（確保抓取數據不為空），以便後續合併。

4. 數據合併與輸出:

   將日盤與夜盤數據合併後，計算「未平倉口數」，並輸出為 Excel 文件。

此全自動化提示工程的優勢:

- 無須人工介入：

  每當程式碼執行失敗時，程式會自動生成改良提示，無需人工手動修正程式碼。

- 上下文記憶：

  提示生成會結合上下文，例如先前的錯誤程式碼與執行結果，進一步提升生成程式碼的準確性。

- 多輪迭代改進：

  確保在多輪修正後達到最優解。

但在使用 Gemini 實現自動化提示工程時，因為 Gemini API 存在使用次數限制， 我們發現迭代到越後面，反而生成的程式碼離我們所預期的結果落差越大，因此更新了方法 3。

(3) 方法 3:

基於方法 2(使用 Gemini 實現自動化提示工程)的做法，因為 Gemini API 存在使用次數限制，必須進一步優化流程以減少不必要的 API 調用。加入使用者回饋機制後，結合錯誤訊息與使用者的回饋，可以更高效地生成符合預期的程式碼。

1. 引入使用者回饋機制:

   每次執行 Gemini 生成的程式碼後，如果發生錯誤，除了捕捉系統自動生成的錯誤訊息外，還允許使用者提供更多具體的回饋，如此一來可以減少 Gemini 的重複調用，因為使用者的回饋能大幅提高下一輪程式碼生成的準確性。

2. 錯誤回饋與動態提示結合:

   程式會整合以下內容到提示中，發送給 Gemini：

   錯誤訊息、目標需求、使用者回饋(對數據格式、處理邏輯等具體需求的補充說明)。

   結合錯誤訊息與使用者回饋，能更準確地生成貼近使用者需求的程式碼。

3. 多輪迭代與精確調整:

   在自動化提示工程中，核心理念是通過「多輪迭代」改進程式碼，而非完全重新生成整段程式碼，因此使用回饋訊息作為輸入，只針

對錯誤部分進行修正，這樣不僅減少 Gemini API 調用次數，還能保持原始程式碼中的正確部分。

此結合使用者回饋的方法之優勢:

- 減少 API 調用次數：
  通過整合錯誤與回饋資訊，生成的程式碼更貼近需求，避免無效的多次重試。
- 結合人類智慧與 AI：
  使用者回饋彌補 Gemini 無法完全理解特殊需求的缺陷。
- 動態調整提示內容：
  提示內容基於執行結果和回饋動態生成，確保每次生成更具針對性。

此基於方法 2 的改進方案，在自動化提示工程中融入了使用者回饋，透過閉環機制（生成—執行—回饋—再生成）和動態提示優化，不僅能有效解決 Gemini 使用上限的問題，還大幅提升生成程式碼的精準度與符合度，最終達成高效完成爬取與處理數據的目標。

(4) 方法 4:

1. 全自動執行：
   此方法為全自動，通過明確的 prompt 設計，避免生成無效或無關的程式碼。不同於方法 2 的全自動，此方法的 prompt 更為精確，並且日夜盤的數據在 prompt 裡一起進行整合。
2. Prompt 設計：
   提供詳細的需求和格式，包含對數據處理、錯誤處理和結果比對的明確說明。
3. 迭代優化：
   每次執行生成的程式碼後，根據失敗結果和錯誤訊息調整 prompt，使用不斷迭代的方法來優化程式碼生成。
4. 程式碼的保存與執行：
   每次生成的程式碼會存為一個 py 檔 (generate_code.py)，執行程式碼後檢查結果，直到程式成功執行。
5. 結果保存：
   程式執行成功後，結果會存成 xlsx 檔 (IR_Final_Prob1.xlsx)。
6. 結果比對：
   將成功的 xlsx 檔跟 hw2 正確的 xlsx 檔 (hw2_A1105505.xlsx)進行數據比對。數據若一致，顯示成功消息，反之，繼續迭代以修正程式碼。

由於在程式執行成功後會將結果保存為 Excel 文件，並與先前的資料進行數據比對，可以使整個自動化流程更具可靠性與精確性。通過

比對數據，程式可以有效驗證生成程式碼的執行結果是否正確，確保生成式 AI 產生的程式碼符合需求並且正確的處理數據。這樣的比對機制可以降低錯誤發生的可能性，也大幅提升數據處理的準確性，讓自動化過程更加準確。

(5) 方法 5:

使用 RAG 系統撰寫程式碼，搭配 Google Gemini 和 Together API 進行除錯，直到獲得成功的結果。最終程式會按照修正後的程式碼執行，並將結果與範例數據進行比較。如果結果一致，則在 app_log.txt 中記錄成功訊息。考慮到 API 的可能限制，若程式執行錯誤達到 10 次，系統將提供使用者手動校正的功能來協助生成更準確的程式碼。

1. RAG

檔案的檢索跟使用者輸入相關的內容作為生成器的輔助參考。首先，根據 PDF 文件與向量存儲的檢索結果，提供檢索到的內容。若未能檢索到相關內容，提示使用者提供更明確的問題。結果結合 system_prompt 和檢索資料傳入生成器。

- PDF 檔案載入與分割：使用 PyMuPDFLoader 加載 PDF 文件。利用 RecursiveCharacterTextSplitter 將文件內容分塊，保證適合向量檢索與存儲。
- 向量存儲與檢索：透過 InMemoryVectorStore 初始化內存向量存儲。使用向量相似度檢索相關文本，根據使用者輸入挑選內容。
- 生成器輸入構造：構建多輪對話格式 (messages)，包含：
  system 提供系統提示，定義生成器的目標與限制。user 傳入使用者問題。assistant 附加檢索到的參考資料。

2. Prompt 的角色與設計

System Prompt：

設定生成器的角色與目標，確保生成結果符合使用者需求。強調輸出內容為 Python 程式碼，並規範程式結構與功能細節。

User Prompt：

包含使用者的具體需求與問題，指導生成器對需求進行細化。

Assistant Context：

根據檢索結果補充生成器所需的背景資料，提升生成內容的相關性。

3. 自動化特點

全流程自動化：

從需求檢索到程式生成與執行完全自動化。錯誤檢測與迭代生成流程避免人工干預。

容錯設計：

執行失敗時，自動找尋錯誤並且生成改進版的程式碼，提升成功率。

日誌追蹤：

將每一步操作與錯誤狀況記錄到 app_log.txt，方便找出錯誤與問題也可以做監控，成功後也會紀錄在文字檔裡。

使用者輔助：

如果多次執行失敗或數據結果不一致，可讓使用者進行手動修正。

我們試了前四種方法，最後選擇使用這一個版本的原因在於他的穩定性跟擁有較高的成功率。此方法結合 RAG 的檢索能力和生成式 AI 的程式碼生成功能，透過多次的迭代與修正達到正確的結果。除此之外，該程式碼設計錯誤的檢測，因此若在 api 嘗試多次都失敗，使用者也可以手動校正來輔助生成正確的程式碼。不僅可以自動將結果和範例的數據進行比對，來確保準確性，也會在每一次的執行過程進行紀錄。此方法的自動化與人工干預的平衡使得整個流程都更加的靈活，並且可以滿足我們的需求。

# 2. Probem2 Architecture

本題主要的問題是如何設計一個完整的架構，以達到問題的要求，因此最重要的是架構設計，過程中當然有遇到其他的問題，像是 token 量不夠，圖片生成不足等等...，我們都不斷地精煉架構與 prompt 設計，最終達成如下:

首先我想先在這裡，講述整體架構，因為我們使用到 RAG 的部分不僅僅只有自動 Prompt 優化，而是整體架構，包含故事生成，其目的是為了可以使故事更加符合資料庫的內容，而詳細的內容請在後續子章節閱讀。

那首先簡單講解我們的 RAG，是參考 LainChain 主要功能，並結合一些我們的技術，像是我們將 LainChain 傳統直接使用的 LLM(openAI 與 Gemini)改成 Together AI，以提供使用者選擇多模組，並可以免費生成圖片，還有在 prompt 初始設計上下了功夫，詳細資訊可以看 2-2 與 2-3。

接著介紹我們本次的 RAG 流程(如下圖 1):

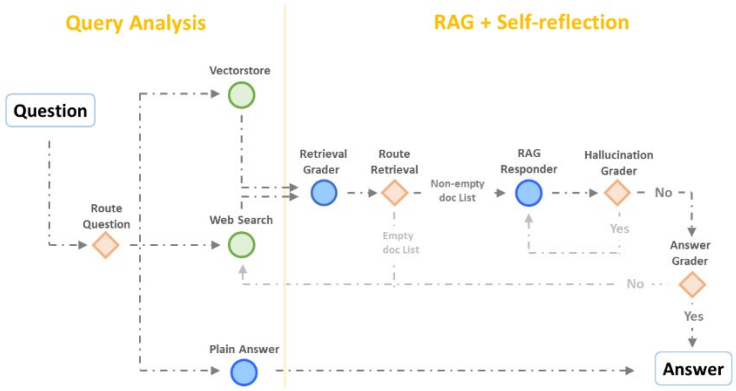



圖 1:RAG(LainChain)架構

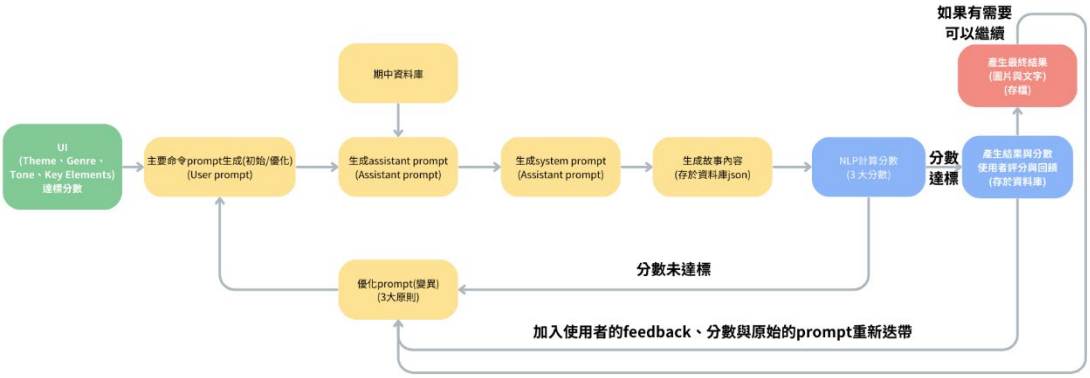
- **圖示定義：**
  綠色圓圈為工具節點，藍色圓圈為 LLM 節點，橘色菱形為 Route 模組，依判斷結果引導至不同節點。
- **RAG 流程：**
  此流程整合 Query Analysis 與 RAG+Self-reflection，並加入多模組處理：

1. **Query Analysis**
   問題經分類器判斷，導向相應工具或模組。實作中依問題面向選擇工具，而非僅依複雜度分流，提升處理效率。
2. **RAG + Self-reflection**
   結合 CRAG 與 Self-RAG 概念，透過多重判斷模組檢查資料提取與生成的有效性，失敗時回到前序模組重試。

- **判斷模組：**

- **Retrieval grader**：過濾與問題無關的搜尋結果，確保 LLM 回應基於有效資訊，無效則重啟搜尋。
- **Hallucination grader**：檢查生成內容是否基於檢索資訊，避免幻覺回應，失敗則重新生成。
- **Answer grader**：確認答案針對性，不符則重啟搜尋流程

而我們結合此 RAG 的概念，時做到本專題之中，我們會於使用到 RAG 模組的部分使用黃色表示。




此流程圖展示了以 **RAG 方法** 為核心的資料處理與故事生成過程，其中**黃色部分代表 RAG 的運作模組**，其餘部分則輔助整個流程的完成。以下是具體說明及與前述 RAG 方法的結合：

**流程概述：**

1. **資料輸入與初始分析(非黃色)**
    - **UI**：輸入主題、風格、語調、key elements 等參數，作為流程的起點。
    - **選擇多模型: ["meta-llama/Meta-Llama-3.1-405B-Instruct-Turbo","Qwen/QwQ-32B-Preview","google/gemma-2-27b-it","microsoft/WizardLM-2-8x22B","togethercomputer/m2-bert-80M-32k-retrieval"]。**
    - **選擇優化時 NLP 分數: coherence, creativity, and relevance。**
    - **主要命令生成**：生成初步的 prompt 作為接下來處理的基礎。
2. **RAG (黃色模組)**
    - **(初始/優化)Prompt**：透過前述選擇的模組，與初始命令產生初始 prompt。後續會自動化不斷地迭帶優化 prompt(詳細內容請參考 2-2)。
    - **資料檢索**：連結期中資料庫，生成 assistant prompt 和 system prompt。
    - **資料生成與處理**：基於檢索內容生成故事內容，並存儲為 JSON 格式。

3. **結果評估與迭代**

- **分數計算 (NLP 評估)**：通過三大分數 (coherence, creativity, and relevance) 判斷結果是否達標。
- **分數未達標時**：根據使用者的反饋與分數調整 prompt，再次進行生成。
- **分數達標時**：輸出最終結果，包括圖片與文字，並將結果與評分存儲至資料庫。

**與 RAG 方法的結合：**

1. **檢索模組 (Retrieval)**

- 流程中檢索期中資料庫並生成相關 prompt，類似於 RAG 的核心檢索機制。
- 可加入 Retrieval grader，對檢索結果進行篩選，確保數據相關性。

2. **回應生成 (Response)**

- Assistant 和 system prompt 的生成與故事內容產生相當於 RAG 的回答模組 (Responder)。
- 可利用 Hallucination grader，確保生成內容基於檢索數據，而非模型幻覺。

3. **自我反思 (Self-reflection)**

- 在結果未達標時，通過使用者反饋進行 prompt 優化，與 RAG 中的反思與迭代機制一致。

# 3. Probem2-1 User Preference Interface

我們的前端是使用 streamlit 撰寫，在 python 的呼叫中會比較特別，需要參考文件中的 ReadMe 使用，並且請注意要自己去申請 Together API，因為我們這組沒有花錢，已經用完許多帳號的免費方案了，需要再麻煩老師或助教要測試的話要注意，謝謝。

本平台在最初會提供使用者設定偏好，如下說明，並最後展示如後續的圖片:

**A. 輸入使用者偏好 (User Preferences)**

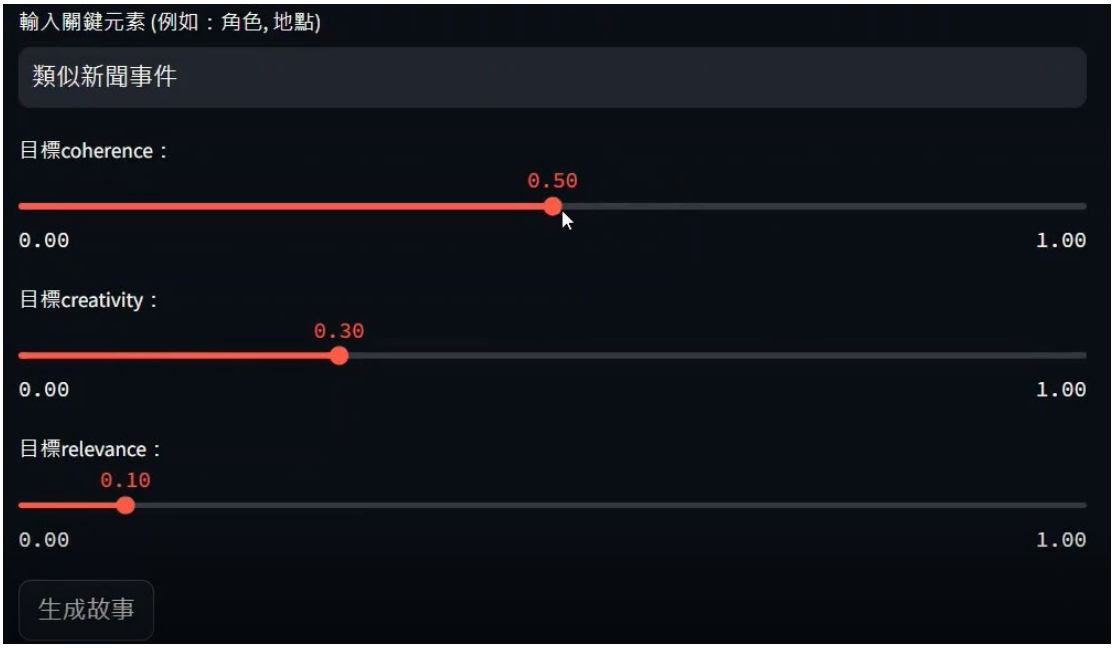
- **主題 (Theme)**: 使用者可選擇以下主題：
  - **Stock (股票)**
  - **Health (健康)**
  - **Sport (運動)**
- **類型 (Genre)**: 使用者可選擇以下內容形式：
  - **短篇故事 (Short Story)**
  - **互動故事 (Interactive Story)**
- **語調 (Tone)**: 使用者可選擇故事語調：
  - **正式 (Formal)**
  - **幽默 (Humorous)**
  - **悲傷 (Sad)**
- **關鍵元素 (Key Elements)**:
  使用者可自行輸入故事中需要包含的關鍵元素 (例如：角色、背景、特殊事件等)。

**B. 選擇多模型 (Model Selection)**

- 使用者可選擇不同模型進行生成，模型列表如下：
  - **meta-llama/Meta-Llama-3.1-405B-Instruct-Turbo**
  - **Qwen/QwQ-32B-Preview**
  - **google/gemma-2-27b-it**
  - **microsoft/WizardLM-2-8x22B**
  - **togethercomputer/m2-bert-80M-32k-retrieval**

**C. 選擇優化指標 (Optimization Metrics)**

- 提供下拉選單讓使用者選擇優化時的 NLP 分數指標：
  - **Coherence (連貫性)**
  - **Creativity (創意性)**
  - **Relevance (相關性)**


# 4. Probem2-2 Dynamic Prompt Engineering Module

### RAG 流程說明

**A. RAG (Retrieval-Augmented Generation) 流程整體描述**

本流程整合了 **Query Analysis** 與 **RAG+Self-reflection**，並進一步加入多模組處理，以提升處理效率與生成內容的精確性。

- **Query Analysis**
  使用分類器對問題進行判斷，導向相應工具或模組。選擇工具的依據是問題的面向，而非僅依問題的複雜度分流，從而優化處理效率。
- **RAG + Self-reflection**
  結合 **CRAG (Controlled RAG)** 與 **Self-RAG** 概念，透過多重判斷模組檢查資料提取與生成的有效性。如判斷失敗，流程將返回前序模組重試，確保結果品質。

**2. 判斷模組介紹**

- **Retrieval Grader**:
  過濾無關的搜尋結果，確保生成回應基於有效檢索資料，若資料無效則重新啟動搜尋。
- **Hallucination Grader**:
  驗證生成內容是否基於檢索資料，避免出現幻覺式回答，如有錯誤則重新生成。
- **Answer Grader**:
  確認答案是否針對問題，若不符則重啟搜尋流程以提升準確性。

**主要技術與應用**

**Dynamic Prompt Engineering 與 LangChain 結合**
本專題透過 **LangChain 模組** 和 **Google Generative AI 的 Embedding 模型**，並使用 **RAG 技術**，實現根據使用者輸入檢索相關文檔，進而生成準確的故事內容。
**具體步驟：**

1. **文檔處理**

   - 使用 **LangChain 的 PyMuPDFLoader** 將 PDF 轉換為文本。
   - 利用 **RecursiveCharacterTextSplitter** 將文本切分為上下文相關的段落。
   - 使用 **Google Generative AI 的 Embedding 模型** 將文本轉換為向量，存儲於 **InMemoryVectorStore**。

2. **內容檢索**

   - 根據使用者輸入進行向量檢索。例如，輸入「生成一個關於運動員的故事」，系統會找到最相關的文本，並整合為生成內容的參考資料，提高準確性與關聯性。

3. **初始與優化 Prompt**

- **Prompt Rephrasing (字詞重構)**

  透過語意調整與字詞變換，提升表達的精準性，我們這裡使用到類似 GA 中變異的概念，使 prompt 再不失去原有的目標的情況下，自動化優化生成新的 prompt：

  • 範例：

```
1. messages = [{"role": "user", "content": f"變異這個指令內容，並保留語意：{question}\n 只要回答變體的指令內容即可"}]
```

**2. Chain-of-Thought (CoT) 分析**

將複雜需求拆解為子問題，並組織成多層次的 Prompt：

• 範例：

```
1. ans = asyncio.run(chat_mutate(f'''請參考目前 prompt 所得文章分數
2.                               coherence:{coherence}(最終須大於{st.session_state.coh})、
3.                               creativity:{creativity}(最終須大於
{st.session_state.cre})、
4.                               relevance:{relevance}(最終須大於{st.session_state.rel})，
5.                               用以下 prompt{ans}繼續變異''',
st.session_state.modelType))
```

**3. Iterative Refinement (反饋迭代)**

根據生成結果進行反饋，迭代改進 Prompt：

• 範例：

```
 1. messages = [
 2.     {"role": "system", "content": system_prompt},
 3.     {"role": "user", "content": user_prompt},
 4.     {"role": "assistant", "content": f"這是參考新聞資料：\n{retrieved_content}"},
 5. ]
 6. ans = asyncio.run(chat_mutate(f'''請參考目前 prompt 所得文章分數
 7.                               coherence:{coherence}(最終須大於{st.session_state.coh})、
 8.                               creativity:{creativity}(最終須大於
{st.session_state.cre})、
 9.                               relevance:{relevance}(最終須大於{st.session_state.rel})，
```

```
10.                                 用以下 prompt{ans}繼續變異''',
st.session_state.modelType))
```

通過結合 RAG 與 Dynamic Prompt Engineering，整合 Query 分析、向量檢索與生成判斷模組的多層次架構，實現高效生成與精準調優，生成內容具備更高的連貫性、創意性與相關性。

# 5. Probem2-3 Generative AI Integration

專題中使用的語言模型，我們採用 Together API 來調用多種生成式 AI 模型，包括 Meta-Llama-3.1-70B 和 Google/GEMMA-2-27B-IT 等…(可以參考前面架構中有講解使用到的模型)，以實現語言生成的多樣性與精準性。這些模型通過 Together API 的統一接口調用，也讓程式的穩定性提升。

而我們設計的 Together API messages 格式包括三部分：system_prompt 定義生成內容的行為規則，例如生成內容需參考 RAG 的資料；user prompt 包含用戶輸入的生成需求(優化或初始的 prompt)；assistant 則 RAG 根據用戶輸入檢索到的新聞內容作為參考資料。

最後，程式通過 Together API 調用指定的語言模型，傳入整理好的 messages，並控制生成參數 max_tokens 和 temperature，確保輸出符合用戶需求。

# 6. Probem2-4 Automatic Content Evaluation & Feedback Loop

在評估生成內容的自動化機制中，我們先通過應用先進的自然語言處理（NLP）技術與基於 BERT 的 Sentence-BERT 模型，有效提供了對文本輸出的結構化評估。該機制注重文本的連貫性、創意性和相關性，目的在為文本生成系統提供清晰的量化指標，是一種專門用於生成句子級向量表示的深度學習模型。

**評估指標**

1. 連貫性（Coherence）：
   - 衡量生成的故事與提示文本之間的語義一致性。該指標使用餘弦相似度來評估兩者的語義相似度。
2. 創意性（Creativity）：
   - 衡量故事的詞彙多樣性。該指標使用 Type-Token Ratio（TTR）來

計算詞彙多樣性，即文本中不同詞彙的比例。

3. 相關性（Relevance）：
   - 衡量故事與參考文本的相關性。若有提供參考文本，則計算故事與該參考文本的餘弦相似度。

**評分流程**

1. 初始化模型：首先，通過 SentenceTransformer 加載指定的 Sentence-BERT 模型。
2. 文本處理：
   - 計算連貫性：通過將生成的故事與提示文本進行比較，計算兩者的餘弦相似度，得出連貫性評分。
   - 計算創意性：使用 jieba 分詞工具將故事文本進行分詞，計算故事的詞彙多樣性，即 Type-Token Ratio（TTR）。
   - 計算相關性：如果提供了參考文本，則計算故事與參考文本的餘弦相似度，得出相關性評分。
3. 返回評分結果：將連貫性、創意性和相關性分別計算並返回，形成一個包含各評分指標的字典。

**核心技術與工具**

1. **Sentence-BERT (SBERT)**
   SBERT 是基於 BERT 的句子表示模型，專門用於生成固定長度的句子向量，以高效計算句子之間的語義相似性。
   - 使用 sentence-transformers 庫加載預訓練模型 paraphrase-multilingual-MiniLM-L12-v2，適合多語言（包含中文）任務。
2. **Jieba 分詞**：
   jieba 是中文分詞工具，用於對中文文本進行分詞，將長文本切分為獨立詞語序列。
   - 計算詞類-詞形比（TTR），衡量詞彙的多樣性。
3. **Cosine Similarity（餘弦相似度）**
   餘弦相似度是衡量兩個向量之間夾角餘弦值的指標，用於計算文本語義相似性。
   - 通過 Scikit-learn 的 cosine_similarity 函數計算句子嵌入向量之間的相似性。
4. **詞彙多樣性指標**
   創意性評估依據 Type-Token Ratio (TTR)，公式如下：
   TTR=total words / unique words
   - unique words：文本中不重複的詞彙數量。
   - total words：文本中所有詞彙的數量。

上述內容，我們於程式碼中，也會不斷地讓我們的自動提示工程，認識到這些數據，幫助 prompt 可以努力朝未達成的指標邁進，如下所示:

```
1. ans = asyncio.run(chat_mutate(f'''請參考目前 prompt 所得文章分數
2.                               coherence:{coherence}(最終須大於{st.session_state.coh}) 、
3.                               creativity:{creativity}(最終須大於
{st.session_state.cre})、
4.                               relevance:{relevance}(最終須大於{st.session_state.rel}) ，
用以下 prompt{ans}繼續變異
5.                                    ''',st.session_state.modelType))
```

後續當 NLP 產生的分數達到使用者初始設定的要求後，我們會讓使用者進行評分與回饋，並將相關的內容，依照使用者是否繼續優化的請求繼續操作，如果使用者決定繼續優化，則我們會提供使用者以下介面(如圖所示)，讓使用者評分與寫回饋:

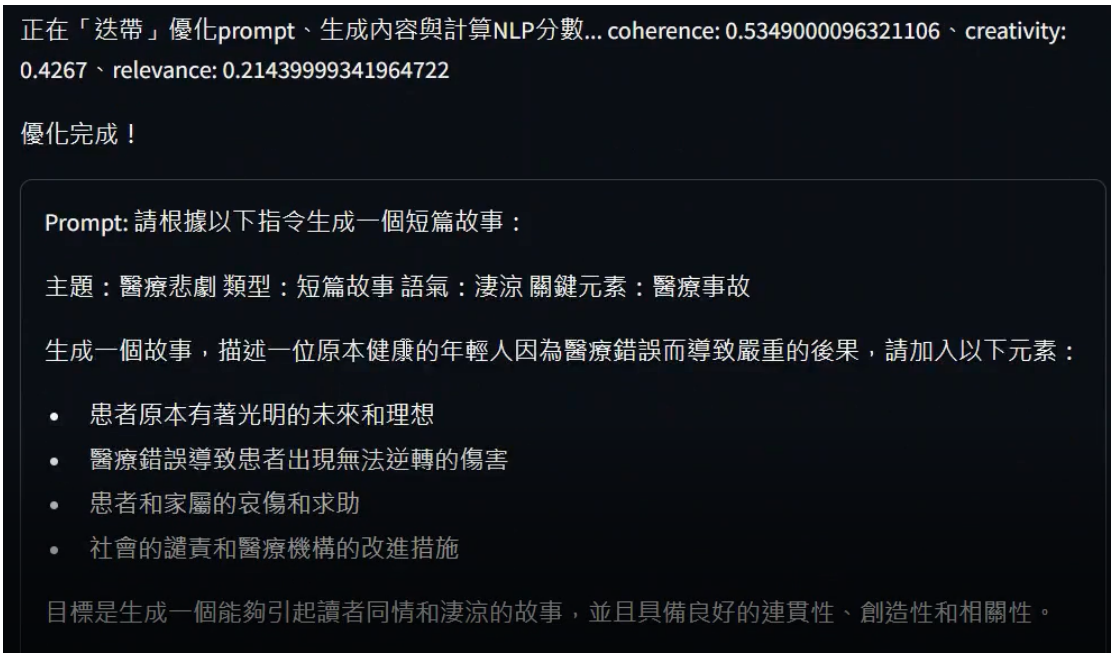

接著再將當前的 prompt 與回饋內容與分數繼續迭帶優化，如下所示:

```
1. ans = f"請參考使用者的回饋分數:{st.session_state.user_score} 和回饋評
語:{st.session_state.feedback}，用以下 prompt {last_entry['prompt']} 繼續變異。"
```

## 7. Probem2-5 Output Visualization and Personalization

生成結果的部分，我們可以不斷地迭帶優化，初始時會顯示「開始」優化，並產生後續「迭帶」結果，如下圖所示:


最終迭帶成功(分數達標)，會產生結果，如下:




接著會讓使用者決定是否繼續優化(需要撰寫回饋與評分):

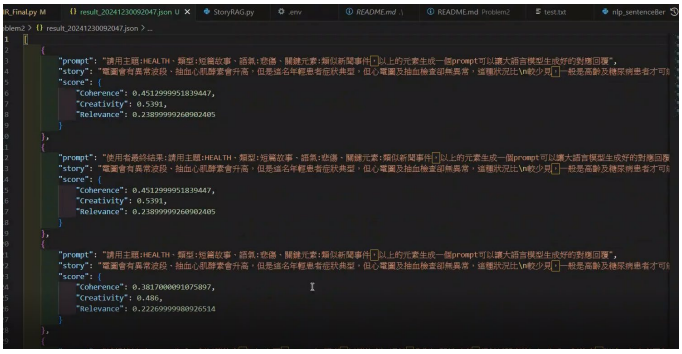

如果選擇繼續優化，則會顯示「初始迭帶」，此為表示有帶著使用者的

回饋與過去 prompt 繼續優化，後續優化會顯示「繼續迭帶」，如下圖所示:

```
正在「初始迭帶」優化prompt、生成內容與計算NLP分數... coherence: 0.23989999294281006、
creativity: 0.5123、relevance: 0.0723000019788742

正在「繼續迭帶」優化prompt、生成內容與計算NLP分數... coherence: 0.4487999975681305、
creativity: 0.4401、relevance: 0.1290999948978424
```

如果選擇產生結果，會匯出資料庫 json 檔與圖片，並且如果不滿意，還是可以繼續迭帶生成優化，如下圖所示:

```
生成的圖片

圖片已成功生成並保存為 'C:\Users\USER\Python_code\ir_genAI\NUK_IR-
GenAI_Final_Project\Problem2/result_20241230092047.png'

結果已保存到 C:\Users\USER\Python_code\ir_genAI\NUK_IR-
GenAI_Final_Project\Problem2/result_20241230092047.json，請重新操作。
```



上述的圖比較是關於故事中的男女，夏墅展現的是，更加符合故事的圖片結果，並且文章生成的文字皆是來自於資料庫，與一些些 LLM 的資料庫輔助生成(如下圖所示)。

以下是一篇新的體育新聞報導：

運動員的堅韌不拔和創紀錄的歷程，是令人激勵和讚嘆的。就以世界著名的田徑運動員，埃利奧特·基普喬蓋（Eliud Kipchoge）為例。他是肯亞的長跑運動員，曾經在2018年創下馬拉松的世界紀錄，時間是2小時1分39秒。

基普喬蓋的成就不是偶然的，他的堅韌不拔和辛苦的訓練是他成功的關鍵。從小時候開始，他就對田徑運動有濃厚的興趣，並且一直堅持訓練。在成為職業運動員後，他每天都會進行嚴格的訓練，以保持自己的體能和技術。

基普喬蓋的成功也得益於他的創新能力。他總是試圖改善自己的訓練方法，並且在比賽中嘗試新的策略。他對自己的體能和技術有著深刻的理解，這使得他能夠在比賽中做出正確的判斷和調整。

基普喬蓋的堅韌不拔和創新能力，不僅使他成為了一名優秀的運動員，也激勵了無數的人。他是

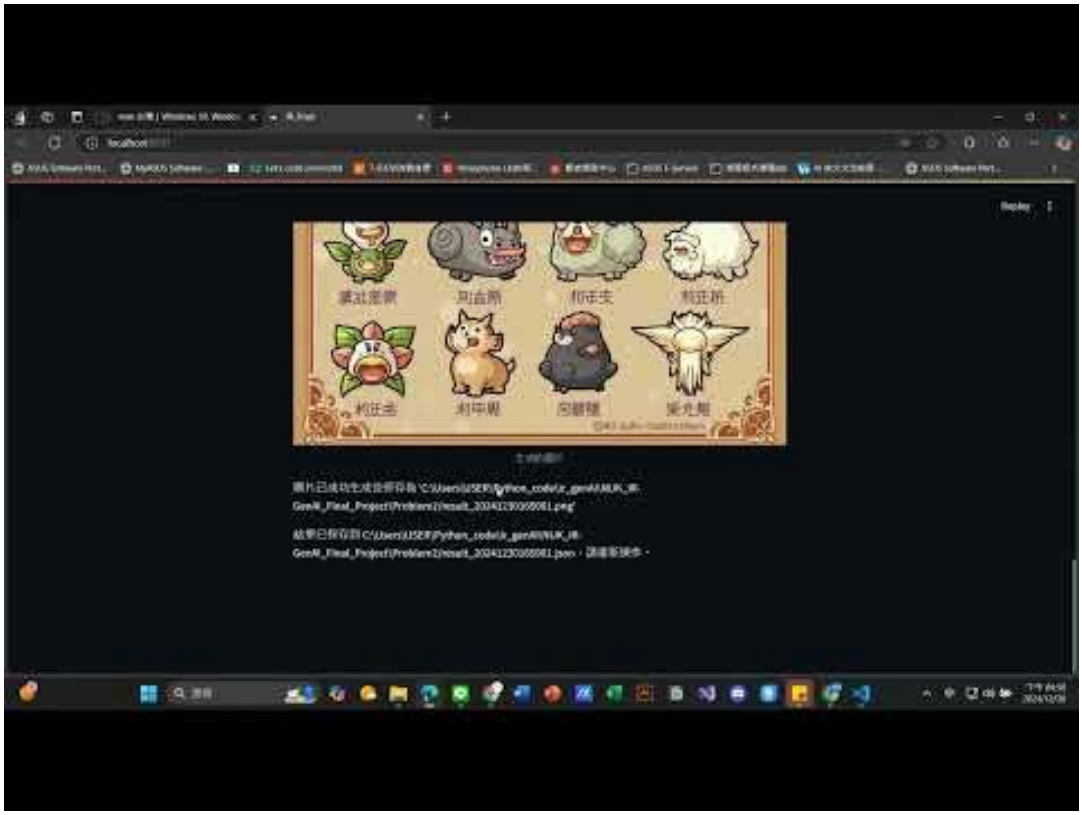
生成的圖片

圖片已成功生成並保存為 'C:\Users\USER\Python_code\ir_genAI\NUK_IR-GenAI_Final_Project\Problem2/result_20241230165403.png'

(下頁繼續)

**影片 1:** (https://youtu.be/SEkmVnCEPos)

**影片 2:** (https://youtu.be/axzKIcIC_zs)



**影片 3: (**https://youtu.be/XiT6RNdnPKM**)**

**一些發現的結論:**

1.Tokens 和 Credits 問題

由於我們沒有錢，我們使用的免費方案，生成圖片有時候很好，有時候微差，但我們可以透過不斷地優化生成決定最好的圖片。

2.多模型:

我們相比起題目要求，額外提供多模型選擇，可以幫助調整訓練。

3.產生的結果:

我們在過程中不斷地調整架構與 prompt，讓結果更加符合資料庫的內容，但發現最終結果中愈有趣的內容，就會愈偏離資料庫。因此可以看到愈原始的結果愈符合資料庫!

(下頁繼續)

**分工**:

| | |
|---|---|
| A1105505<br>林彧頎 | 第一題與第二題主要程式撰寫者，與後續微調，影片錄製、ppt 製作、報告撰寫、口頭報告、merge everything。 |
| A1105507<br>蘇柏諺 | 第二題開發 RAG、ppt 製作、報告撰寫、口頭報告。 |
| A1105521<br>黎子崴 | 第二題開發 NLP、ppt 製作、報告撰寫、口頭報告。 |
| A1105523<br>巫柔筠 | 第一題主要負責者、ppt 製作、報告撰寫、口頭報告。 |
| A1105524<br>吳雨宣 | 第一題主要負責者、ppt 製作、報告撰寫、口頭報告。 |