



# Term Project2

Team 3

**Hosted By:**

A1105505 林彧頤

A1105507 蘇柏諺

A1105521 黎子歲

A1105523 巫柔筠

A1105524 吳雨宣

**IR & GenAI**

# Table of Contents

- 01** Problem1
- 02** Probem2-1 User Preference Interface
- 03** Probem2-2 Dynamic Prompt Engineering Module
- 04** Probem2-3 Generative AI Integration
- 05** Probem2-4 Automatic Content Evaluation & Feedback Loop
- 06** Probem2-5 Output Visualization and Personalization

**1.**

# **Problem 1**



# 方法 1

## 用prompt從使用者的自然語言輸入中提取查詢條件

### # 利用 Together API 進行 Prompt 解析

- 將用戶的自然語指令轉化為結構化的查詢條件
- 起始日期與結束日期、要爬取的數據項目

### #據目標爬取網站數據

- 用 BeautifulSoup 解析臺股期計網站中相關表格

### #數據格式化和查詢

- 輸出到Excel，供後續查詢和分析

```
prompt = f"""\n
```

從以下輸入中提取細節：

- 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。
- 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)：

.....

# 方法 2

## 全自動化提示工程技術爬取日夜盤

### # 動態提示生成

- 目標網址、數據抓取目標（如「未平倉餘額」）、日期參數、錯誤訊息

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}

.....

### #多次迭代校正

- 執行爬蟲程式碼時捕捉錯誤訊息，自動生成修正後的提示進行多輪優化

# 方法 3

## 加入使用者回饋的自動化提示工程技術

### # 引入使用者回饋機制

- 執行程式碼後，允許使用者提供具體的改進建議，結合錯誤訊息提升生成提示的精準度

### # 動態提示結合

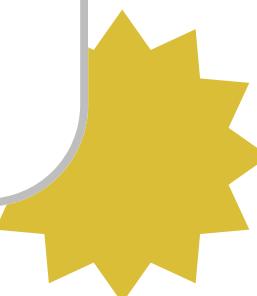
- 結合錯誤訊息、目標需求與使用者回饋

### # 多輪迭代與精確調整

- 僅針對錯誤部分進行修正，保留原始程式碼中的正確部分

### # 跟方法2相比

- 使用者回饋機制補足 Gemini 對需求的理解不足
- 減少 API 調用次數



# 方法 4

## 全自動執行

### # 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，且日夜盤的數據在prompt裡一起進行整合

### #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

### #程式碼的保存與執行

每次生成的程式碼會存為一個py 檔，執行程式碼後檢查結果，直到程式成功執行

### #結果保存

程式執行成功後，結果會存成xlsx檔

### #結果比對

將成功的xlsx檔跟hw2正確的xlsx檔進行數據比對。數據若一致，顯示成功。反之，繼續迭代修正程式碼。

# 方法 5 (最終方法)

## RAG

### # 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

### # 自動化的特點

#### **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：**

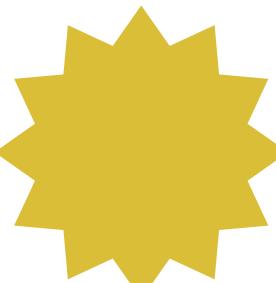
執行失敗時，自動找尋錯誤並且生成改進版的程式碼，提升成功率。

#### **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。

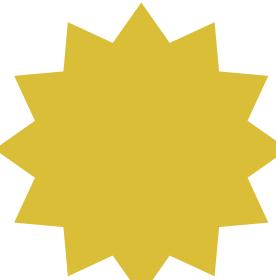


# 方法 5 (最終方法)

## 結論

### # 作為最終版的原因

- 穩穩定性、成功率高
- 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
- 使用者手動校正來輔助生成正確的程式碼  
(擔心API不足，錯誤10次開放此功能)
- 自動比對結果的數據與提供的數據是否一致
- 紀錄每一次的執行過程，方便除錯



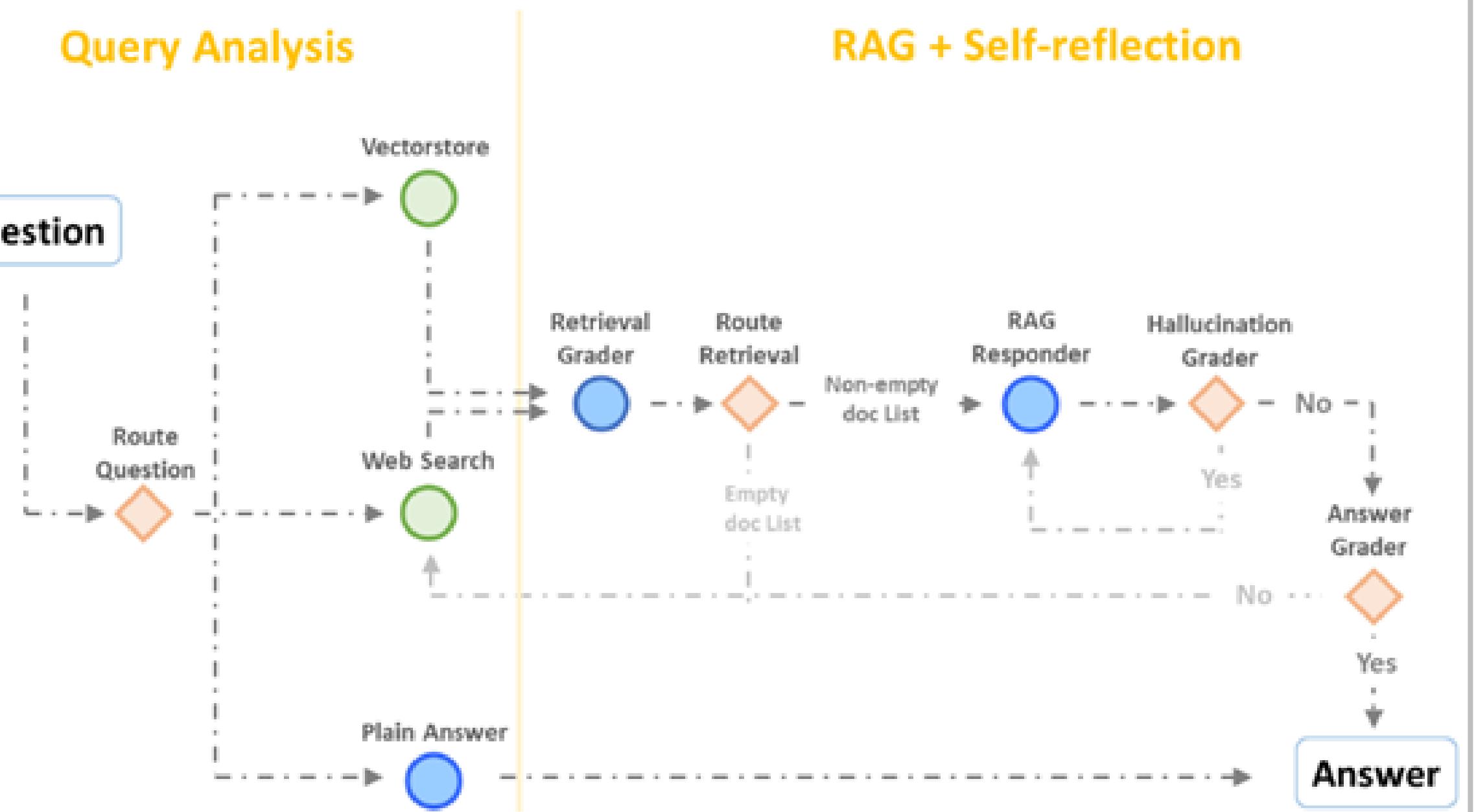
# Problem 2



# Architecture



# RAG 流程



## 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。

### 2.RAG + Self-reflection

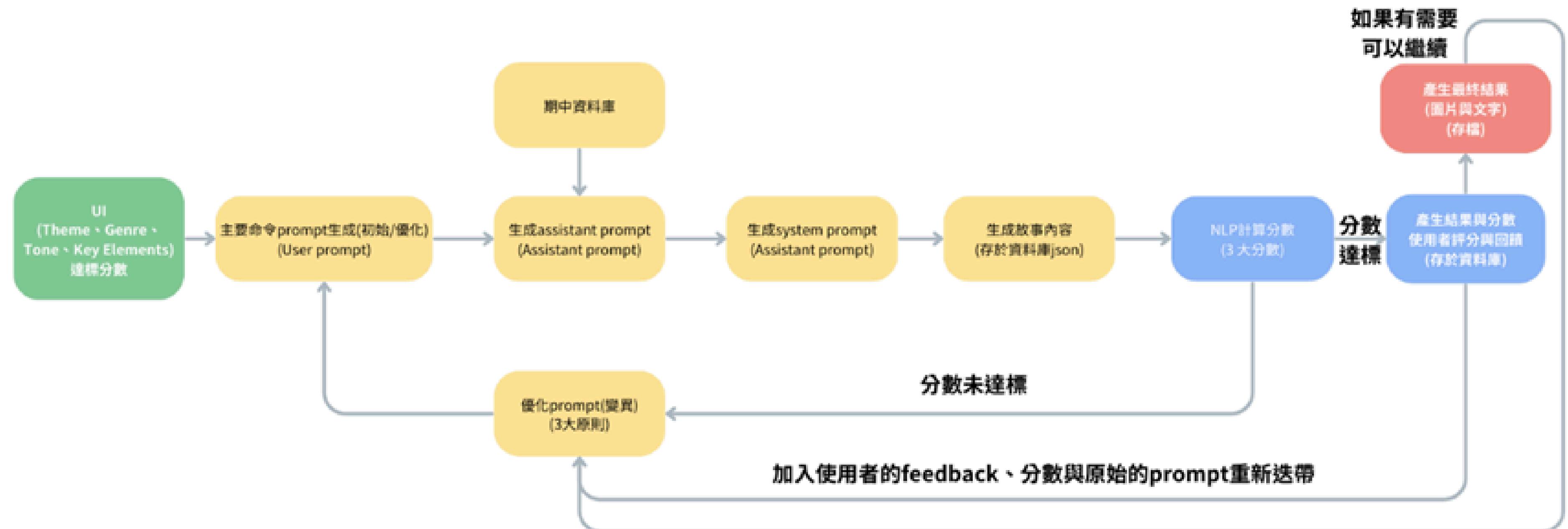
結合 **CRAG** 與 **Self-RAG** 概念，透過多重判斷模組檢查資料提取與生成的有效性，失敗時回到前序模組重試。

## 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

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。

# 整體架構



# 整體架構-說明

- 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。
- 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。
  - **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。
- 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。

# 1. User Preference Interface



# 使用者偏好-說明(1/2)

- 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。

# 使用者偏好-說明(2/2)

- **B. 選擇多模型 (Model Selection)**
  - **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. 選擇多模型 (Model Selection)**
  - **Coherence (連貫性)**
  - **Creativity (創意性)**
  - **Relevance (相關性)**

# AI故事生成平台

選擇模型

meta-llama/Meta-Llama-3.1-405B-Instruct-Turbo

選擇主題

STOCK

選擇類型

短篇故事

選擇語氣

幽默

輸入關鍵元素 (例如：角色, 地點)

類似新聞事件

目標coherence :

0.00

0.50

1.00

目標creativity :

0.00

0.30

1.00

目標relevance :

0.00

0.10

1.00

生成故事

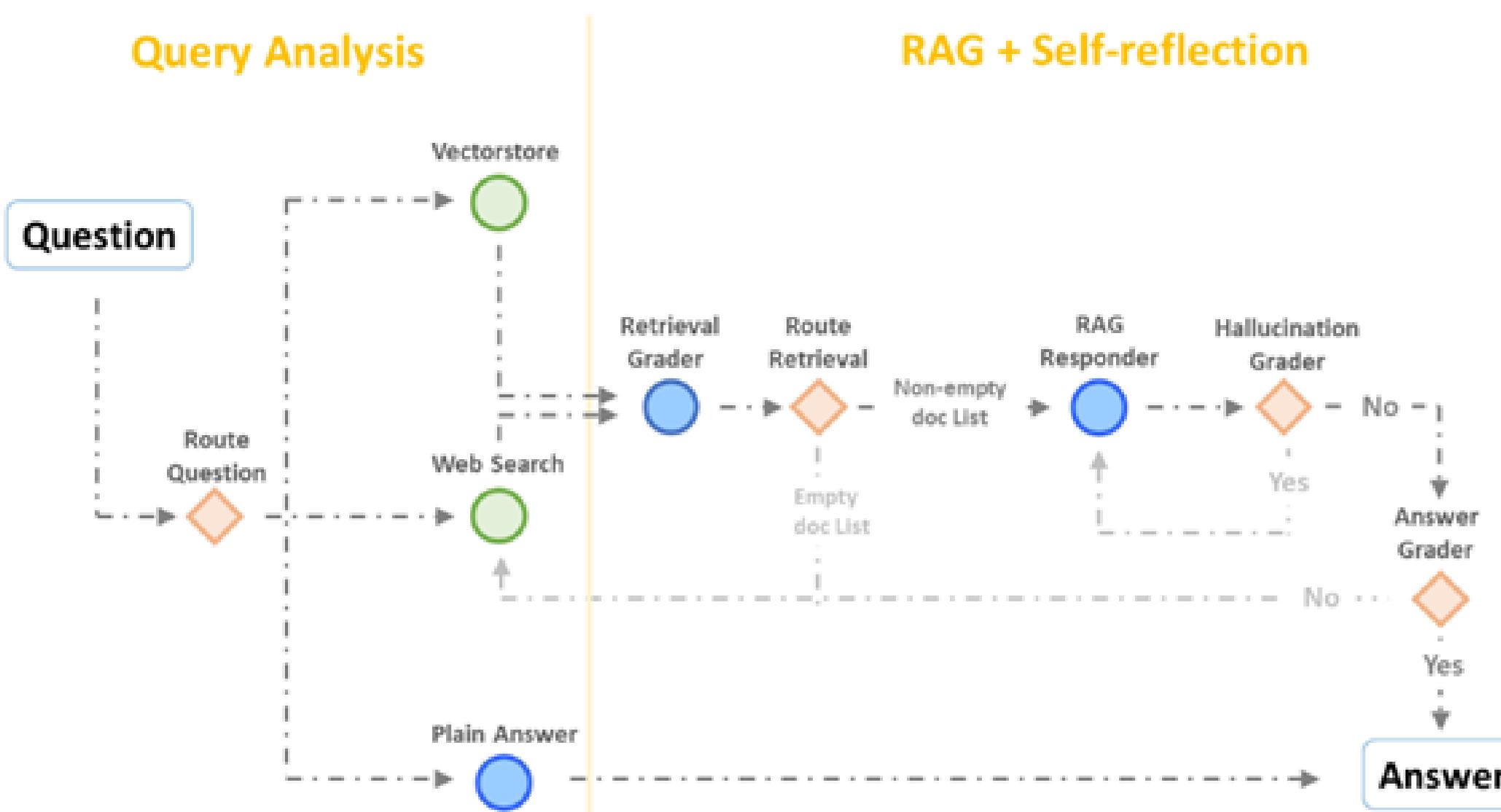
# **2.**

# **Dynamic Prompt Engineering Module**



# 總體說明

1. RAG架構 (如前面架構所述)
2. 主要技術與應用(後續說明)



# 技術架構

## Google Generative AI 的 Embedding 模型 (models/embedding-001)

- 負責將pdf文字轉換為向量形式，以便進行向量檢索和相似性計算

## 使用到的 LangChain 元件

- RecursiveCharacterTextSplitter 將文件進行切分，以便檢索
- InMemoryVectorStore 根據用戶的輸入，進行向量相似度檢索

# RAG 技術的應用

## # PDF 文件加載與處理

```
pdf_content = PyMuPDFLoader(pdf_path).load()
```

## # 文本切分

```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=100)
split_content = text_splitter.split_documents(pdf_content)
```

## #Embedding生成向量並存儲

```
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
```

```
vector_store = InMemoryVectorStore(embeddings)
vector_store.add_documents(documents=split_content)
```

# RAG 技術的應用

# 基於用戶提示檢索

```
retrieved_docs = vector_store.similarity_search(user_prompt)
```

# 整合檢索結果

```
retrieved_content = ""  
for retrieved_doc in retrieved_docs:  
    retrieved_content += retrieved_doc.page_content
```

# RAG 技術的應用-Prompt 優化

使用到類似GA的變異概念：

## # Prompt Rephrasing (字詞重構)

```
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"}]
```

## # Chain-of-Thought (CoT) 分析

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

# RAG 技術的應用-Prompt 優化

## # Iterative Refinement (反饋迭代)

```
messages = [
    {"role": "system", "content": system_prompt},
    {"role": "user", "content": user_prompt},
    {"role": "assistant", "content": f"這是參考新聞資料：\n{retrieved_content}"},  
]  
  
ans = asyncio.run(chat_mutate(f'''請參考目前prompt所得文章分數  
coherence:{coherence}(最終須大於{st.session_state.coh})、  
creativity:{creativity}(最終須大於{st.session_state.cre})、  
relevance:{relevance}(最終須大於{st.session_state.rel})'')，  
用以下 prompt{ans}繼續變異'', st.session_state.modelType))
```

# 3. Generative AI Integration



# 技術架構

**Together API** 用於切換使用以下語言模型

- 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

# 整合prompt 並設定不同的role

# system\_prompt用於說明語言模型的目標

```
system_prompt = """  
你是一個專業的新聞說故事小幫手，  
你的目標是根據retrieved_content的新聞內容，產生故事，但不行偏離參考新聞資料。  
"""
```

#結合system\_prompt + 優化的user prompt + RAG的 retrieved\_content

```
messages = [  
    {"role": "system", "content": system_prompt},  
    {"role": "user", "content": user_prompt},  
    {"role": "assistant", "content": f"這是參考新聞資料：\n{retrieved_content}"},  
]
```

**4.**

# **Automatic Content Evaluation & Feedback Loop**



# 技術架構

## 使用模型 - Sentence-BERT

- 模型：paraphrase-multilingual-MiniLM-L12-v2新聞數據，使用批量查詢以提高效率。
- 功能：生成句子級向量，支援中文及多語言。

## 依賴套件

- sentence-transformers：Sentence-BERT 模型。
- torch：深度學習框架。
- transformers、numpy、scikit-learn：模型及數學工具。
- jieba：中文分詞工具，用於詞彙多樣性分析。

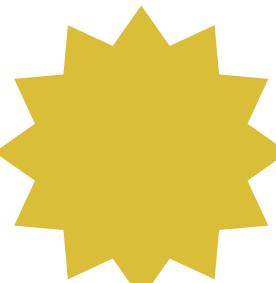
# Sentence-BERT

## 1. 高效語義理解能力

- 基於 BERT 架構，專為句子級別任務設計，能生成高質量語句向量表示。
- 適合語義相似度計算，例如 Coherence 和 Relevance 評估。

## 2. 多語言支持

- 支援中文及多語言處理，適合國際化應用場景。
- 模型如 paraphrase-multilingual-MiniLM-L12-v2 專門針對多語言調校。



# 評估指標

Coherence :

計算生成文本與提示文本的餘弦相似度

Creativity :

- 利用 Type-Token Ratio (TTR) 評估詞彙多樣性。
- 使用 jieba 進行中文分詞後計算：
  - $TTR = \text{唯一詞數} / \text{總詞數}$

Relevance :

- 如果提供 reference\_text，計算生成文本與參考文本的 餘弦相似度。

# 程式碼核心結構

# 相似度計算

```
def calculate_similarity(text1, text2):  
    embeddings = model.encode([text1, text2])  
    return cosine_similarity([embeddings[0]], [embeddings[1]])[0][0]
```

# 創意性計算

```
def calculate_diversity(text):  
    words = jieba.cut(text) words_list = list(words)  
    unique_words = set(words_list)  
    return len(unique_words) / len(words_list) if words_list else 0
```

# 程式碼核心結構

## # 結果輸出

```
return {  
    "scores": {  
        "Coherence": round(coherence, 4),  
        "Creativity": round(creativity, 4),  
        "Relevance": round(relevance, 4),  
    },  
}
```

# 程式碼核心結構-評分迭帶

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

# 使用者回饋

王晨陽的故事是一個悲劇，它提醒我們醫療錯誤的

Score: {'Coherence': 0.5349, 'Creativity': 0.4267, 'Relevance': 0.2144}

請對故事進行評價：

請輸入您的反饋：

我覺ㄉㄉ

Press Enter to submit form

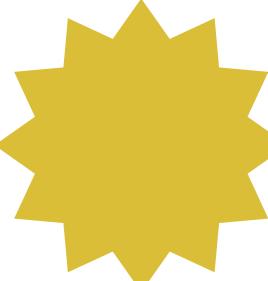
請為故事評分：

0.00

0.00 1.00

產生結果

繼續優化



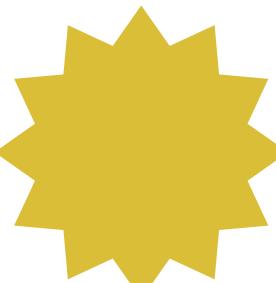
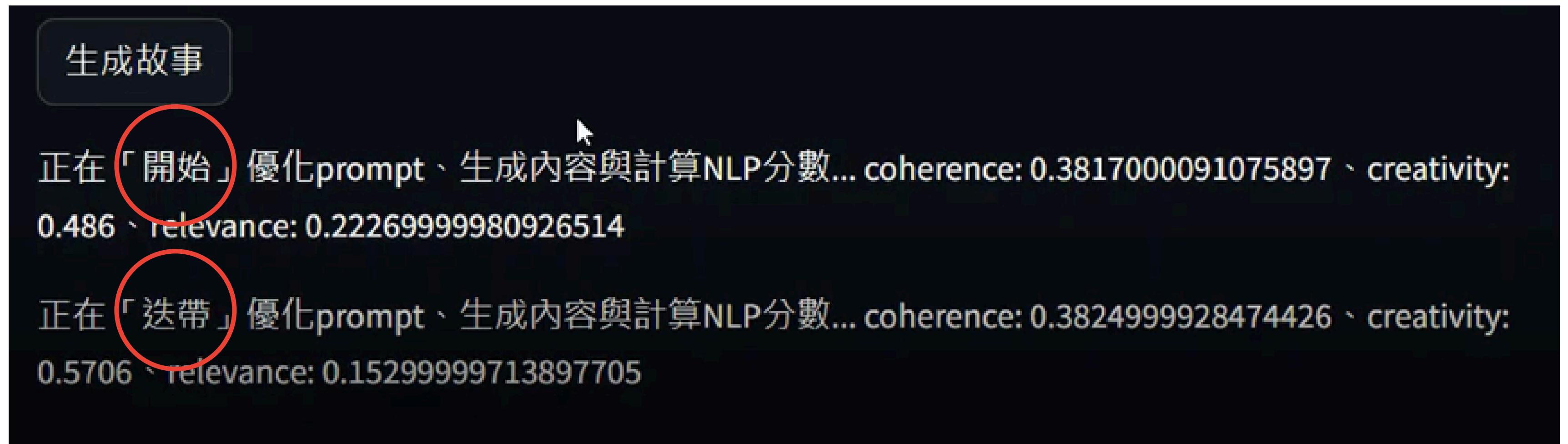
# 程式碼核心結構- 使用者回饋迭帶

**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。"**

# 5. Output Visualization and Personalization



# 初始迭帶



# 初始迭帶-結果

正在「迭帶」優化prompt、生成內容與計算NLP分數... coherence: 0.5349000096321106、creativity: 0.4267、relevance: 0.21439999341964722

優化完成！

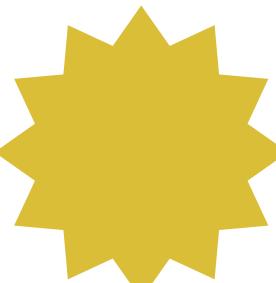
Prompt: 請根據以下指令生成一個短篇故事：

主題：醫療悲劇 類型：短篇故事 語氣：淒涼 關鍵元素：醫療事故

生成一個故事，描述一位原本健康的年輕人因為醫療錯誤而導致嚴重的後果，請加入以下元素：

- 患者原本有著光明的未來和理想
- 醫療錯誤導致患者出現無法逆轉的傷害
- 患者和家屬的哀傷和求助
- 社會的譴責和醫療機構的改進措施

目標是生成一個能夠引起讀者同情和淒涼的故事，並且具備良好的連貫性、創造性和相關性。



# 是否繼續優化(需要撰寫回饋與評分)

王晨陽的故事是一個悲劇，它提醒我們醫療錯誤的

Score: {'Coherence': 0.5349, 'Creativity': 0.4267, 'Relevance': 0.2144}

請對故事進行評價：

請輸入您的反饋：

我覺ㄉㄉ

Press Enter to submit form

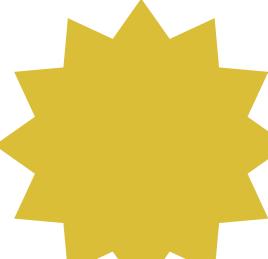
請為故事評分：

0.00

0.00 1.00

產生結果

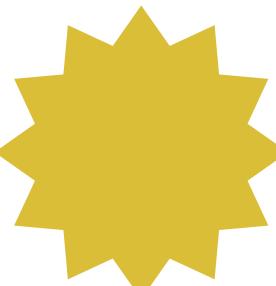
繼續優化



# 繼續優化

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



# 結果生成(存檔)



生成的圖片

```
IR_Final.py M ① result_20241230092047.json U X StoryBAG.py .env ② READMEmd ③ READMEmd Problem2 test.txt ④ nlp_sentencecl ⑤
```

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。

```
[{"prompt": "請用主題:HEALTH、類型:短篇故事、語氣:悲傷、關鍵元素:類似新聞事件\n以上的元素生成一個prompt可以讓大語言模型生成好的對應回覆",
"story": "電圖會有異常波段，抽血心肌酵素會升高，但是這名年輕患者症狀典型，但心電圖及抽血檢查都無異常，這種狀況比較少見\n一般是高齡及糖尿病患者才可",
"score": {
    "Coherence": 0.4512999951839447,
    "Creativity": 0.5391,
    "Relevance": 0.23899999260902405
},
{
    "prompt": "使用者最終結果: 請用主題:HEALTH、類型:短篇故事、語氣:悲傷、關鍵元素:類似新聞事件\n以上的元素生成一個prompt可以讓大語言模型生成好的對應回覆",
    "story": "電圖會有異常波段，抽血心肌酵素會升高，但是這名年輕患者症狀典型，但心電圖及抽血檢查都無異常，這種狀況比較少見\n一般是高齡及糖尿病患者才可",
    "score": {
        "Coherence": 0.4512999951839447,
        "Creativity": 0.5391,
        "Relevance": 0.23899999260902405
    }
},
{
    "prompt": "請用主題:HEALTH、類型:短篇故事、語氣:悲傷、關鍵元素:類似新聞事件\n以上的元素生成一個prompt可以讓大語言模型生成好的對應回覆",
    "story": "電圖會有異常波段，抽血心肌酵素會升高，但是這名年輕患者症狀典型，但心電圖及抽血檢查都無異常，這種狀況比較少見\n一般是高齡及糖尿病患者才可",
    "score": {
        "Coherence": 0.3817600091075897,
        "Creativity": 0.486,
        "Relevance": 0.2226999980936514
    }
},
```

# 結果生成(存檔)



生成的圖片

圖片已成功生成並保存為 'C:\Users\USER\Python\_code\ir\_genAI\GenAI\_Final\_Project\Problem2\result\_20241230165403.png'

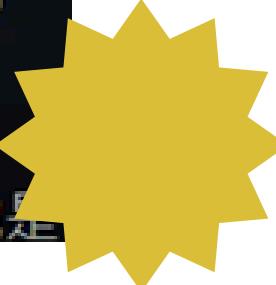
以下是一篇新的體育新聞報導：

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



# 影片示範-1

IR Final Problem2 Example1

RUNNING... Stop Deploy :

Prompt: 請根據以下指令生成一個短篇故事：

主題：醫療悲劇 類型：短篇故事 語氣：淒涼 關鍵元素：醫療事故

生成一個故事，描述一位原本健康的年輕人因為醫療錯誤而導致嚴重的後果，請加入以下元素：

- 患者原本有著光明的未來和理想
- 醫療錯誤導致患者出現無法逆轉的傷害
- 患者和家屬的哀傷和求助
- 社會的譴責和醫療機構的改進措施

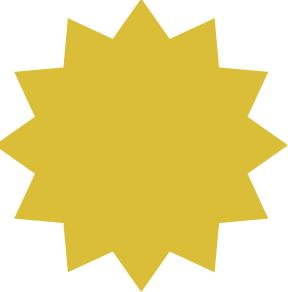
目標是生成一個能夠引起讀者同情和淒涼的故事，並且具備良好的連貫性、創造性和相關性。

Story: 根據您的要求，我們來創造一個故事，描述一位原本健康的年輕人因為醫療錯誤而導致嚴重的後果。

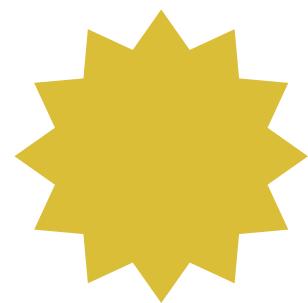
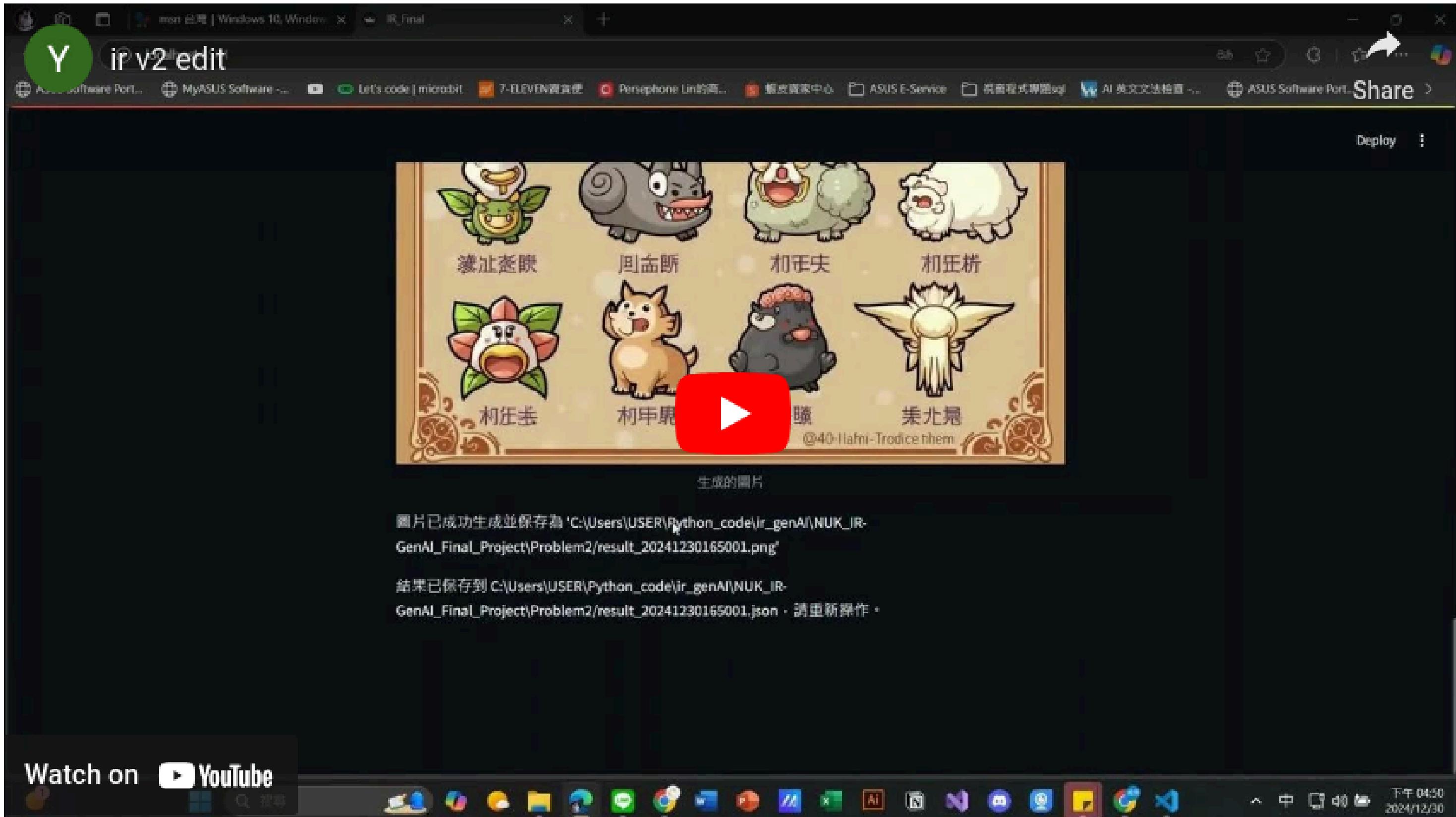
王晨陽是一位22歲的大學生，他的未來充滿希望和理想。他是一位優秀的學生，同時也是一位熱情的運動員。他夢想著成為一位成功的企業家，並且能夠讓他的家人過上幸福的生活。

然而，王晨陽的生活在一天突然發生了變化。他因為嚴重的頭痛而前往醫院求診。醫生告訴他，他的頭痛是由於叢發性頭痛引起的，並且需要進行高濃度氧氣治療與預防用藥調整。

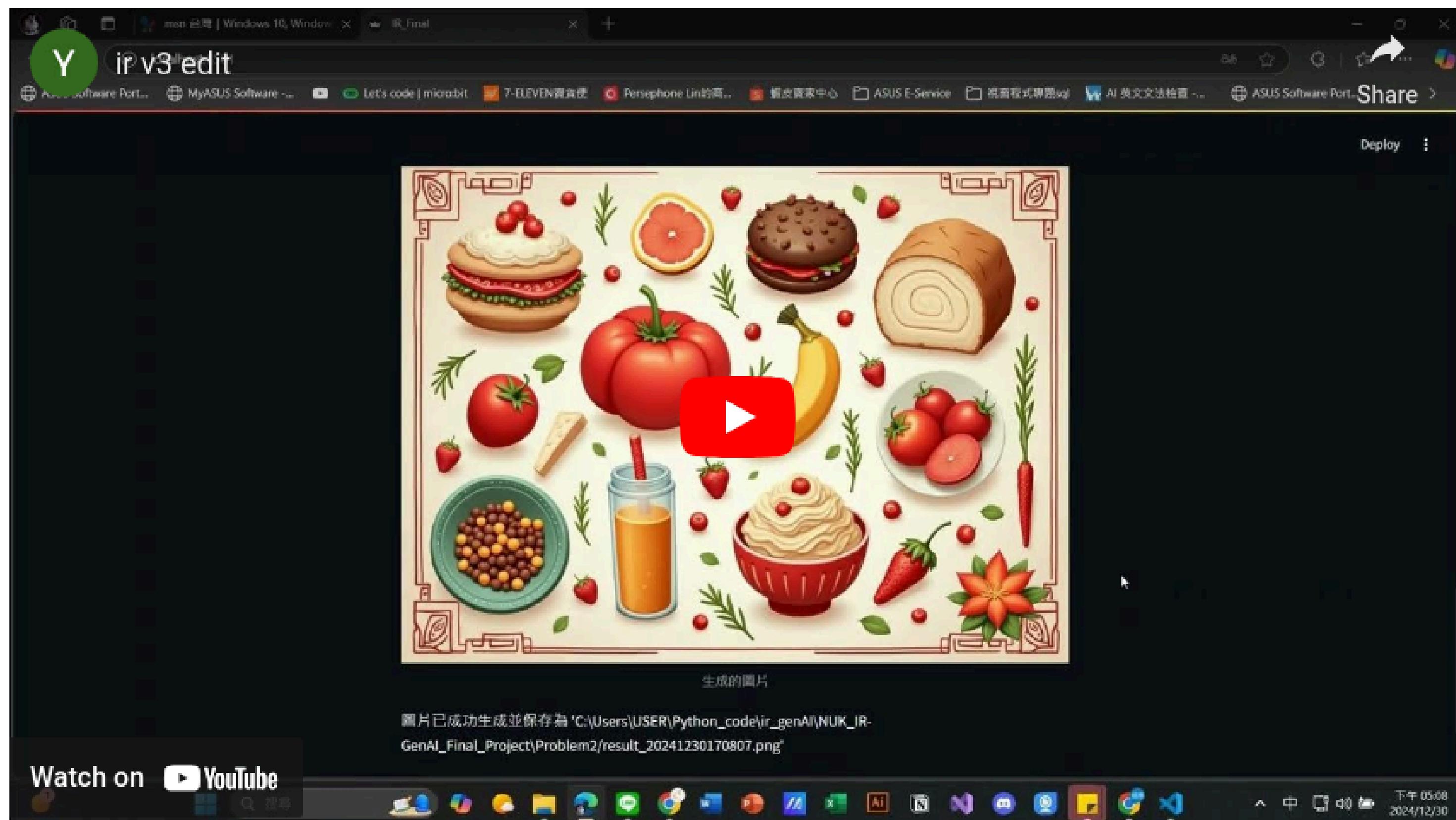
Watch on [YouTube](#)



# 影片示範-2



# 影片示範-3



# Conclusion

## 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。

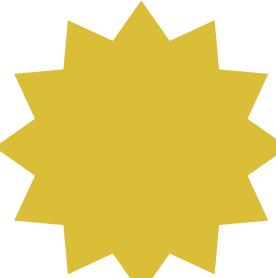
## 2. 多模型：

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!!!



# **Thank you for your listening!**

