**Sentiment News Project Team4**

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. Multi-source Crawling 1](#_Toc181548990)

[2. Real-time Data Collection 3](#_Toc181548991)

[3. Topic-specific Crawling 5](#_Toc181548992)

[4. Sentiment Analysis 8](#_Toc181548993)

[5. Dynamic and Adaptive Crawling 9](#_Toc181548994)

[6. Data Storage 10](#_Toc181548995)

[7. Visualization 25](#_Toc181548996)

[心得 29](#_Toc181548997)

1. Multi-source Crawling(如有需要雲端資料庫請私訊林彧頎提供github，或參考.sql的資料庫)

我們爬取的新聞網站有「中時」、「自由時報」、「tvbs」，以及為了讓新聞來源更豐富，我們使用API(https://newsapi.org/)來爬取更多新聞來源，**請注意我們為了幫助資料庫可以快速爬取，並平衡資料庫，因此我們有設置當日新聞最多爬取三則，因此新聞量不符合實際網頁的結果，是我們僅爬取三筆的結果，但這三筆皆是經過篩選後確定為該類別的新聞**。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 中時 | | | 自由時報 | | | tvbs | | | API | | |
| 時間 | 最多爬到  2年前  (2022/10/24) | | | 最多只能爬500頁  (有些只能爬不到2年，有些最久可爬到2004年) | | | 最多可以爬到295頁(2003/6/3) | | | 最多爬到  1個月前 | | |
| 資料筆數 | H | St | Sp | H | St | Sp | H | St | Sp | H | St | Sp |
| 5068 | 11903 | 15895 | 15268 | 19188 | 13894 | 15952 | 16013 | 30422 | 237 | 373 | 351 |
| Total:32866 | | | Total:48350 | | | Total:62387 | | | Total:961 | | |
| 全部爬蟲量:144564 | | | | | | | | | | | |
| 新聞  來源數 | 1間 | | | 1間 | | | 1間 | | | 24間 | | |

(H:Health的資料筆數/St:Stock的資料筆數/Sp:Sport的資料筆數)

根據上方表格可知，通常當一間新聞網的的新聞量越大，相對的能爬取到的時間跨度越小(當然也有例外狀況，但是是少數，像是tvbs是因為他的sport數較多)。  
(1) 中時:

* User-Agent 模擬:

程式碼使用一組常見的 User-Agent 列表，每次請求隨機選擇一個 User-Agent，以模擬不同裝置和瀏覽器的訪問，防止因固定 User-Agent 而被偵測為爬蟲。

* 隨機延遲:

在每次請求之間加入隨機延遲（1-3秒），可以避免持續、高頻率的請求，降低被反爬蟲機制察覺的可能性。

* 防止 Webdriver 被檢測:

透過 Chrome WebDriver 中的 execute\_cdp\_cmd 方法，將 navigator.webdriver 屬性設定為 undefined，這可以掩蓋 WebDriver 的使用，避免被網站偵測到是自動化工具。

* 多組選擇器(CSS\_SELECTOR):

使用 safe\_find\_element 函數來指定多組選擇器，確保即使網站隨機更改結構，也可以找到目標元素，增加爬蟲的穩定性和兼容性。

* 重試機制:

若在預設的等待時間內無法加載目標元素，會進行最多3次重試，以應對網站的臨時延遲或加載問題。

* 每次抓取完一頁後，要關閉瀏覽器並重新打開網頁，如果直接從已開啟的網頁換頁，會被該網頁阻擋。

(2) 自由時報:

* 延遲請求:

在爬取每個頁面之後，加入time.sleep(1)，讓每次請求之間稍作等待，避免過度頻繁請求觸發網站的rate-limiting

* 例外處理與重新載入:

使用 WebDriverWait 和 TimeoutException 來應對載入超時的情況。當網頁沒有在指定時間內載入完畢，會捕獲 TimeoutException，並重新嘗試載入頁面，用以應對因為伺服器延遲或載入緩慢導致的失敗。

* 過濾重複連結:

因為我們在這間新聞的爬取方式是先將該網頁上的所有連結存入news\_url\_list，但網頁上可能會出現重複的新聞連結，因此我們建立 seen\_hrefs 集合儲存已添加的網址，避免重複爬取同一篇新聞，這可以減少重複請求並提高效率。

(3) tvbs:

* 延遲請求:

使用 time.sleep(2) 在每頁爬取後進行延遲，避免過於頻繁的請求，從而減少被認為是異常行為的可能性。

* 多重選取器容錯:

使用多個選取器查找資料，例如時間和摘要的元素定位，這樣即使網站改變結構，也能增加抓取成功的機會。

* 異常處理:

使用 try-except 機制來捕獲錯誤並忽略單篇文章的解析錯誤，確保爬蟲能繼續執行而不會因為小問題而停止。

1. Real-time Data Collection

我們用了第一題所提及的種種方式處理反爬蟲機制，觀察每間新聞網可爬取的最長時間，並盡可能爬到最多的資料量，接著我們設計一個功能，讓使用者可設定從今日起追溯多久以前的新聞資料。  
在儲存資料的部分，我們使用了supabase，是因為它可以支援即時數據同步，讓組員們可以同時共享及操作資料庫。但因為免費版的用戶，單一資料庫的儲存上限為500MB，而我們爬取的新聞量十分龐大，所以開了兩個資料庫來儲存所有資料。

這兩個資料庫分別命名為 “IR\_GenAI\_Team\_Proj1” 和“IR\_GenAI\_Team\_Proj1\_second\_Database”，每個資料庫包含12個資料表儲存3個topic相關的新聞及情緒分析:

**資料表stock:**

|  |  |
| --- | --- |
| stockID | stock\_name |
| 股票代號 | 股票名稱 |

**資料表 stock\_news:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| stockID | title | date | content | source | id | url |
| 股票代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

**資料表 stock\_news\_API:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| stockID | title | date | content | source | id | url |
| 股票代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

**資料表 stock\_news\_sentiment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | news\_id | sentiment | emotion | star |
| index | 新聞編號 | 情緒分數 | 情緒類型 | 星等 |

**資料表health:**

|  |  |
| --- | --- |
| healthID | health\_name |
| keyword代號 | keyword名稱 |

**資料表 health\_news:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| healthID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

**資料表 health\_news\_API:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| healthID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

**資料表 health\_news\_sentiment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | news\_id | sentiment | emotion | star |
| index | 新聞編號 | 情緒分數 | 情緒類型 | 星等 |

**資料表sport:**

|  |  |
| --- | --- |
| sportID | sport\_name |
| keyword代號 | keyword名稱 |

**資料表 sport\_news:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| sportID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

**資料表 sport\_news\_API:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| sportID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

**資料表 sport\_news\_sentiment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | news\_id | sentiment | emotion | star |
| index | 新聞編號 | 情緒分數 | 情緒類型 | 星等 |

在資料表 stock\_news, health\_news, sport\_news 中，content 欄位因為包含大量文字且也屬於主鍵之一，在進行全文檢索時，易導致查詢效率降低。因此，這裡使用了 PostgreSQL 的 GIN（Generalized Inverted Index）索引來加速這類查詢。

CREATE INDEX sport\_news\_content\_gin\_idx ON sport\_news USING GIN (to\_tsvector('simple', content));

建立了一個名為 ‘sport\_news\_content\_gin\_idx’ 的 GIN 索引，用於加速 ‘sport\_news’ 表中 ‘content’ 欄位的全文檢索。‘USING GIN’ 指定使用 GIN 索引類型，適合處理大量文本資料，並生成inverted index以加速關鍵字查找。‘to\_tsvector('simple', content)’ 將 ‘content’ 欄位轉換為文本向量格式，‘simple’ 配置不區分詞幹、停用字，增加查詢的靈活性，適合快速搜尋。

1. Topic-specific Crawling

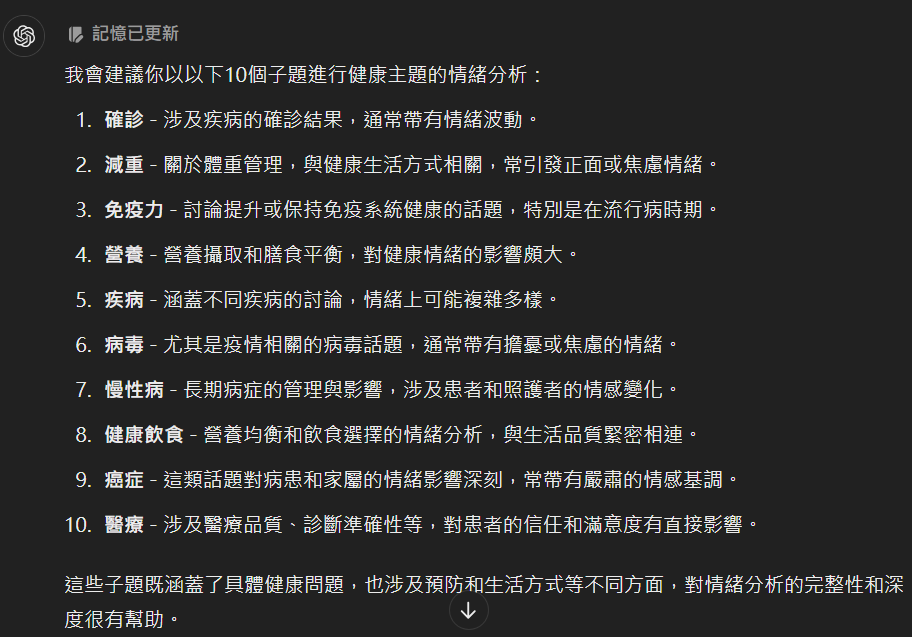
本專題中，主要選定的主題為股市、健康與運動，這三主題可以說是現代人日常生活中最關心的三大議題，我們選定主題後，並不是直接將主題的名詞直接當作我們過濾新聞的主要指標，因為我們發現，如果僅單純定義某個主題的定義為關鍵字，雖然範圍很大可能以單一來看資料量多，但是該主題下許多小主題相加起來的資料量更多(如下圖比較所示)。





**(由上述兩張圖可看出，設定出單一類別就會比大方向的資料來的多)**

因此我們決定為每一個主題都定義十個子題，這些子題會跟主題有巨大的關聯，並且我們也利用ChatGPT來回答我們最好的十個子題做參考(如下一頁圖所示)。



然後我們也參考資料量相對多的中時，來判斷該子題的新聞量多寡，決定這些子題的可靠性與資料量，最後我們得出3個主題與各十個子題，最後有30個主題可以獲得，此30個主題請參閱第二節或第六節的資歷庫介紹。

以下表格將介紹各主題與各子題的類別含意:

Stock(股市)

|  |  |
| --- | --- |
| 統一 | 追蹤統一企業於市場的營運狀況與投資機會。 |
| 鴻海 | 了解鴻海供應鏈及全球市場的科技影響力。 |
| 台積電 | 探索台積電於半導體領域的競爭優勢。 |
| 廣達 | 分析廣達電子製造業務的成長趨勢與合作夥伴。 |
| 中華電 | 追蹤中華電在電信業的發展和技術創新。 |
| 聯發科 | 探究聯發科在半導體市場的產品競爭力。 |
| 長榮 | 觀察長榮於運輸和物流業的業績走勢。 |
| 富邦金 | 了解富邦金於金融保險市場的佈局策略。 |
| 長榮航 | 分析長榮航空的航線拓展及旅遊趨勢。 |
| 元大金 | 研究元大金於資產管理及金融市場的角色。 |

Health(健康)

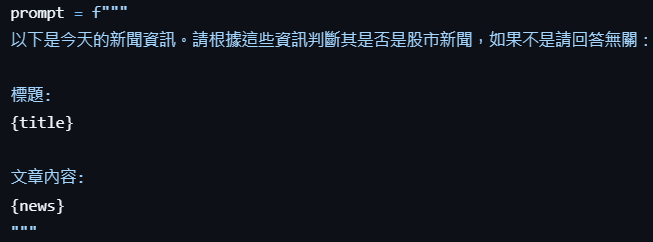
|  |  |
| --- | --- |
| 確診 | 追蹤疫情確診數據以評估健康風險。 |
| 減重 | 探索減重方法及其對健康的影響。 |
| 免疫力 | 了解增強免疫力的方法和相關保健品。 |
| 營養 | 分析營養對健康的益處與飲食建議。 |
| 疾病 | 探討常見疾病的預防和治療方法。 |
| 病毒 | 研究病毒傳播途徑與防護措施。 |
| 慢性病 | 了解慢性病的成因及健康管理策略。 |
| 健康飲食 | 介紹健康飲食的要素與對身體的益處。 |
| 癌症 | 探索癌症的最新治療和預防進展。 |
| 醫療 | 追蹤醫療科技發展及對健康的影響。 |

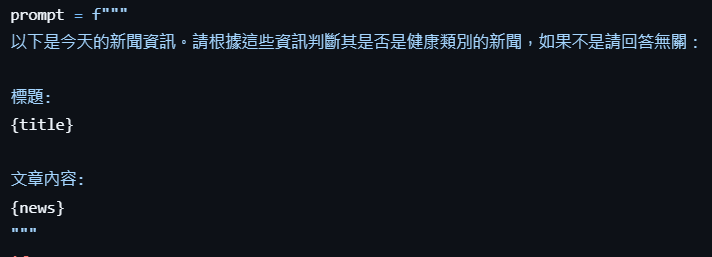
Sport(運動)

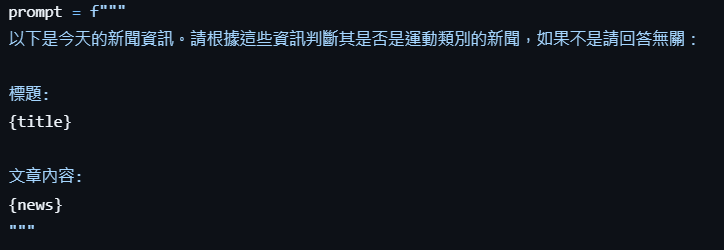
|  |  |
| --- | --- |
| 賽事 | 追蹤各類運動賽事的賽程及結果。 |
| 球員 | 研究球員的表現和職業生涯發展。 |
| 聯賽 | 分析各大聯賽的競爭趨勢及影響力。 |
| 世界盃 | 探討世界盃比賽及球隊的參賽表現。 |
| 籃球 | 追蹤籃球比賽及其全球影響力。 |
| 奧運 | 研究奧運賽事及選手表現的亮點。 |
| NBA | 探索NBA的明星球員與聯賽動態。 |
| MLB | 了解MLB的賽季變化及選手動向。 |
| 棒球 | 研究棒球運動的發展和國際賽事。 |
| 中職 | 追蹤中華職棒的賽事及球員動態。 |

另外關於處理不相關的資料，我們主要透過三項技術來做文字過濾，第一項是於爬蟲中，我們使用關鍵字，於該網也搜尋相關新聞，第二項是利用Gemini，使用以下的prompt(會依照不同的主題做改變)，進行文字過濾，第三則是如果於Gemini的API不可用時，會改成利用keyword的方式檢查是否為該類別的文章，這樣可以有效的過濾非該類別或是廣告的資料。

以下是prompt:







1. Sentiment Analysis

情緒分析部分，我們採用 BERT 模型 "nlptown/bert-base-multilingual-uncased-sentiment" 來針對新聞文本進行自動評分。這個模型以多語言的 BERT 為基礎，專門訓練來處理多語言的情緒分析任務，能夠為每一篇新聞內容給出 1 到 5 顆星的評分。透過星級評分，我們可以進一步將情緒分類成三種主要類型：

1. **負面**：如果模型給出的星級評分為 1 星或 2 星，表示新聞情緒較為負面，反映出該新聞可能包含負面的詞彙或情緒，例如批評、爭議等。
2. **中立**：評分為 3 星的新聞被認為是中立的，意味著內容可能是客觀描述，缺乏強烈的情緒色彩，或包含正負情緒較平衡的內容。
3. **正面**：當新聞的評分為 4 星或 5 星時，代表該新聞具備正面的情緒特徵，可能帶有支持性、樂觀的語氣或偏向積極的觀點。

為了統一數據的結構與便於後續分析，我們將這些情緒結果整理並儲存在 sentiment table 中。儲存的內容包含：

1. **情緒類型**：以 "positive"、"neutral" 和 "negative" 來區分每篇新聞的情緒性質。
2. **星級評分**：將原始星級評分 1-5 直接記錄在資料庫中，以便細分分析使用。

**(後續第7節將介紹分析的結果)**

1. Dynamic and Adaptive Crawling

針對Dynamic crawling與Adaptive crawling我們主要的操作如下:

1. Dynamic Crawling 動態爬蟲

Dynamic Crawling 是一種靈活應對網頁變動的爬蟲策略，適合處理頻繁變動的網站。具體技巧如下：

* Selenium: 用於自動化瀏覽器行為，特別適合處理需要互動（如點擊、滾動、等待動態加載）的網頁。透過模擬瀏覽器行為，我們可以從網頁中獲取經 JavaScript 渲染的內容。
* BeautifulSoup: 配合 Selenium 抓取靜態內容，解析網頁 HTML 結構。BeautifulSoup 提供簡單直觀的接口，用來定位網頁中的特定元素，利於資料提取。
* API: 有些網站提供 API 以供程式化訪問資料。直接訪問 API 相比網頁爬取更加穩定和高效，可以避免不必要的渲染和頁面加載過程。
* User Agent: 設置合適的 User Agent 以模擬真實的瀏覽器訪問。不同的網站對於 User Agent 的要求可能不同，隨時切換適當的 User Agent 可以減少封鎖風險。
* Headers: 根據網站的要求自訂 Headers，這包括設定 Referer、Accept、Accept-Language 等，讓請求看起來像是從瀏覽器發出的。這樣可以增強訪問的真實性，避免被封鎖。
* 模擬人類行為: 觀察各網站的封鎖機制，根據網站的反爬行措施動態調整，例如隨機停頓（TIMER）來模擬正常瀏覽速度，並防止 Webdriver 被檢測，或是不要連續點過多網頁，而適當關閉與開啟網頁，避免不正常的過九訪問。
* TIMER: 動態設置隨機停頓時間，讓爬蟲行為更接近人類操作，減少網站封鎖的機率。
* 生成式 AI: 使用生成式 AI 輔助過濾不必要的新聞，且也可以加入生成式AI於給出網頁爬蟲的改良建議，想適架構改變時，適時將網頁資訊與目前的爬蟲，提供給生成式AI詢問修改方向。

1. Adaptive Crawling 自適應爬蟲

Adaptive Crawling 則更著重於錯誤處理和動態適應不同網頁結構的改變。具體技巧如下：

* CSS Selector: 利用 CSS 選擇器定位網頁元素。當網站結構發生輕微變動時，可以透過調整選擇器來適應新結構，減少爬蟲中斷的可能性。
* 正規表達式處理結構錯誤: 針對結構錯誤或網頁資料的變動，使用正規表達式提取關鍵資料，避免依賴特定 HTML 結構，提高代碼的適應性。
* 類似元素辨識: 當結構變化時，透過觀察相似元素的標籤或屬性模式來找到相對應的資料。例如，如果某個元素的 class 名稱改變，可以根據其他不變的屬性辨識出目標資料。
* User Agent: 在 Adaptive Crawling 中持續更改 User Agent，應對網站的反爬機制。自適應爬蟲會根據不同的網站需求調整 User Agent，並在檢測到異常時更改其內容。
* 生成式 AI 輔助撰寫代碼: 生成式 AI 可協助撰寫調整過後的爬蟲代碼。例如，用生成式 AI 輔助過濾不必要的新聞，且也可以加入生成式AI於給出網頁爬蟲的改良建議，想適架構改變時，適時將網頁資訊與目前的爬蟲，提供給生成式AI詢問修改方向。
* 錯誤通知與自動重試: 當爬蟲遇到異常時會自動通知，並根據預設的重試策略重新執行爬取作業。這樣可以有效避免因暫時性錯誤而導致的數據缺失。

Adaptive Crawling 的核心在於適應結構變化和處理異常，透過結合正規表達式、AI 輔助代碼生成和自動錯誤處理機制，使爬蟲在面對動態變化的網頁時能夠保持穩定高效。

1. Data Storage

我們使用了兩個資料庫，分別命名為“IR\_GenAI\_Team\_Proj1”和“IR\_GenAI\_Team\_Proj1\_second\_Database”，每個資料庫包含12個資料表儲存3個topic相關的新聞及情緒分析:

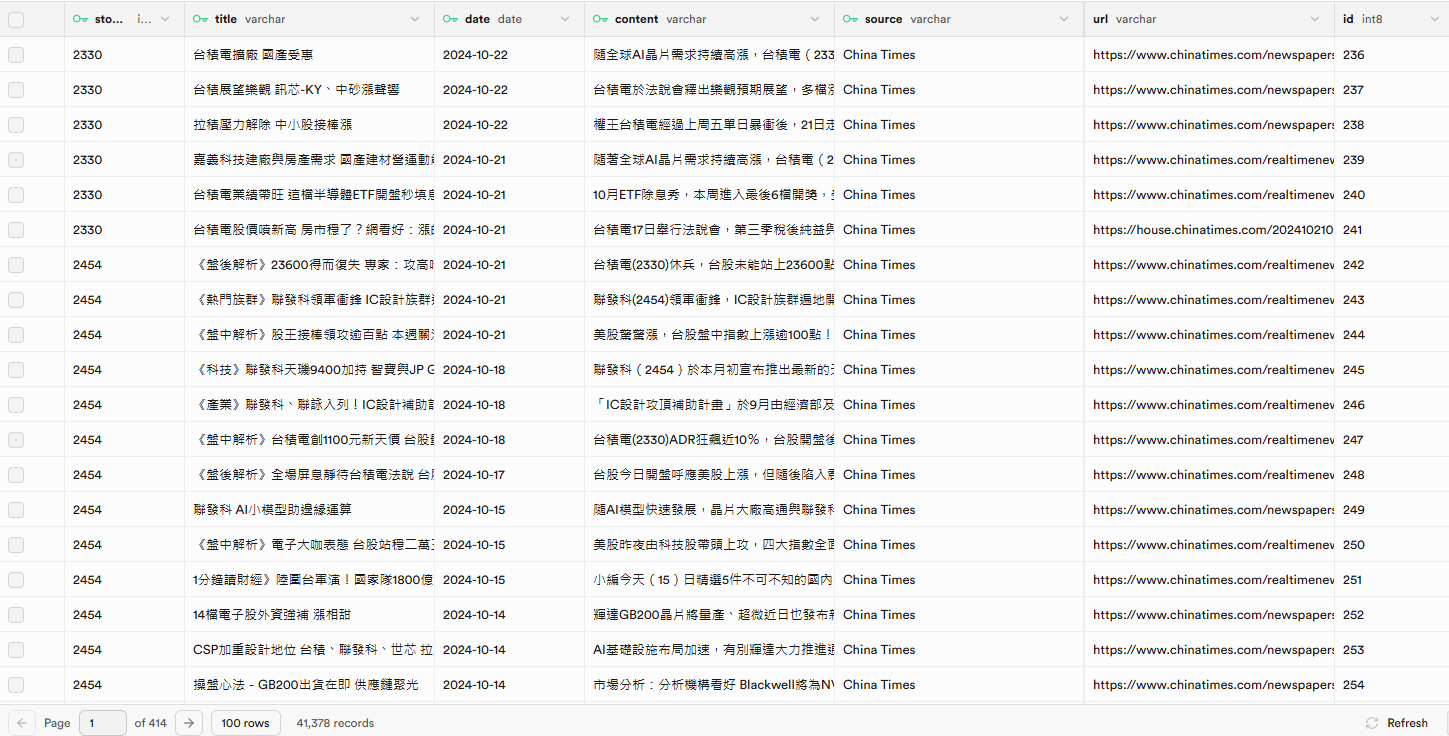
**資料表stock:**

|  |  |
| --- | --- |
| stockID | stock\_name |
| 股票代號 | 股票名稱 |

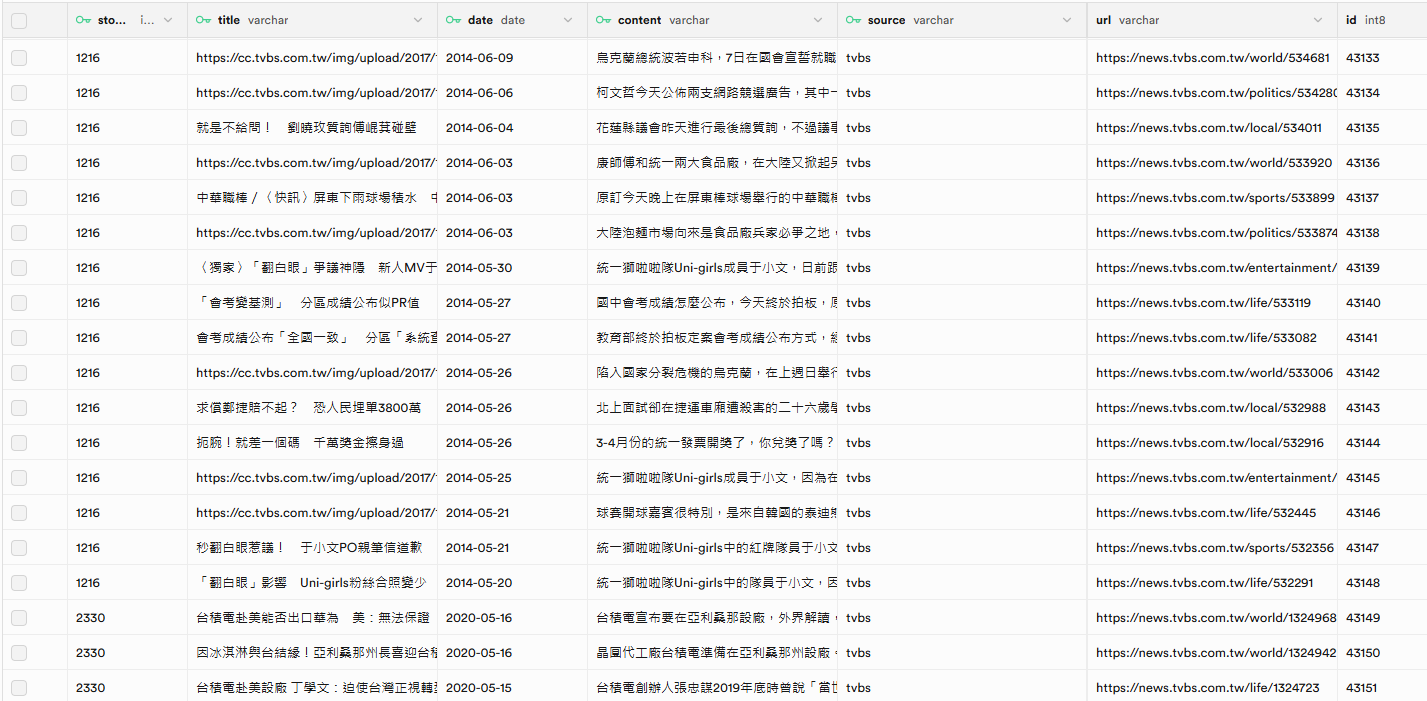


**資料表 stock\_news:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| stockID | title | date | content | source | id | url |
| 股票代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

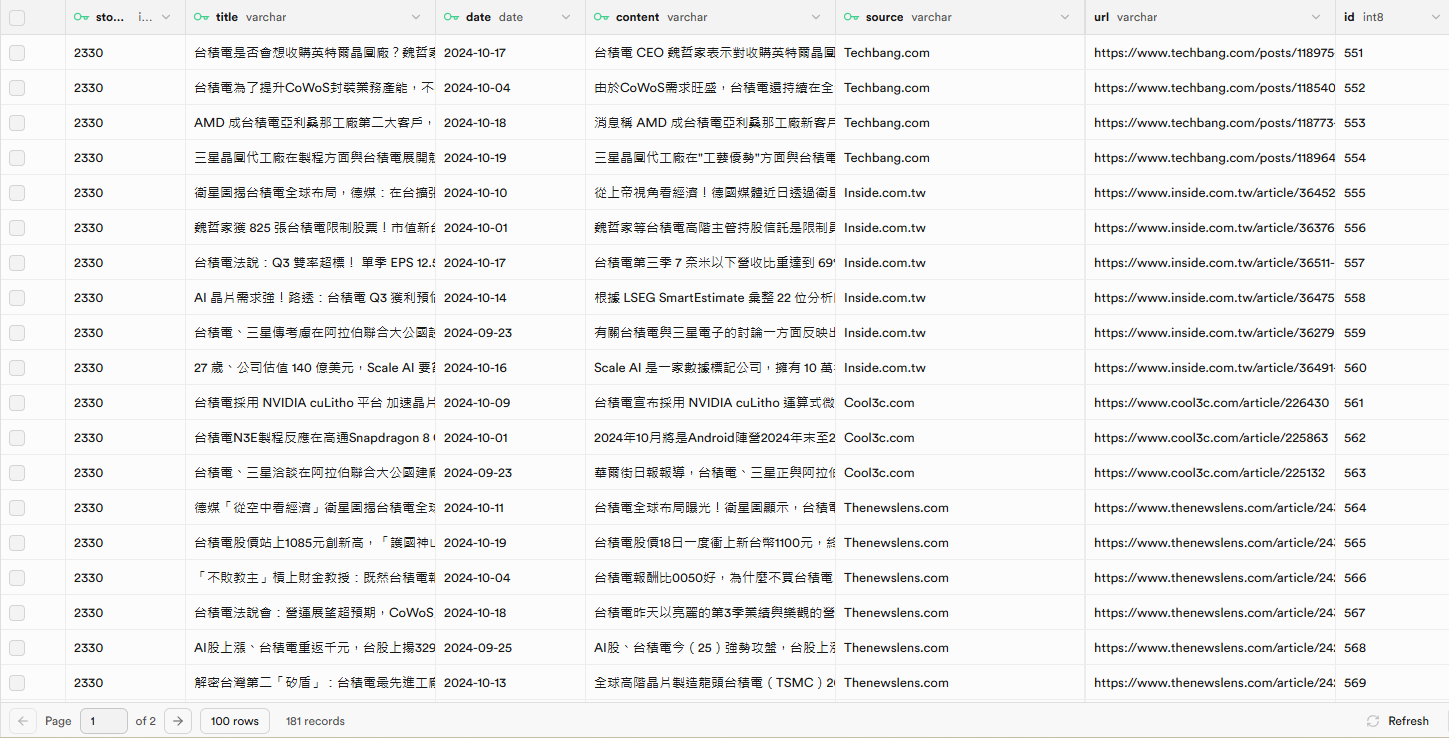


(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)

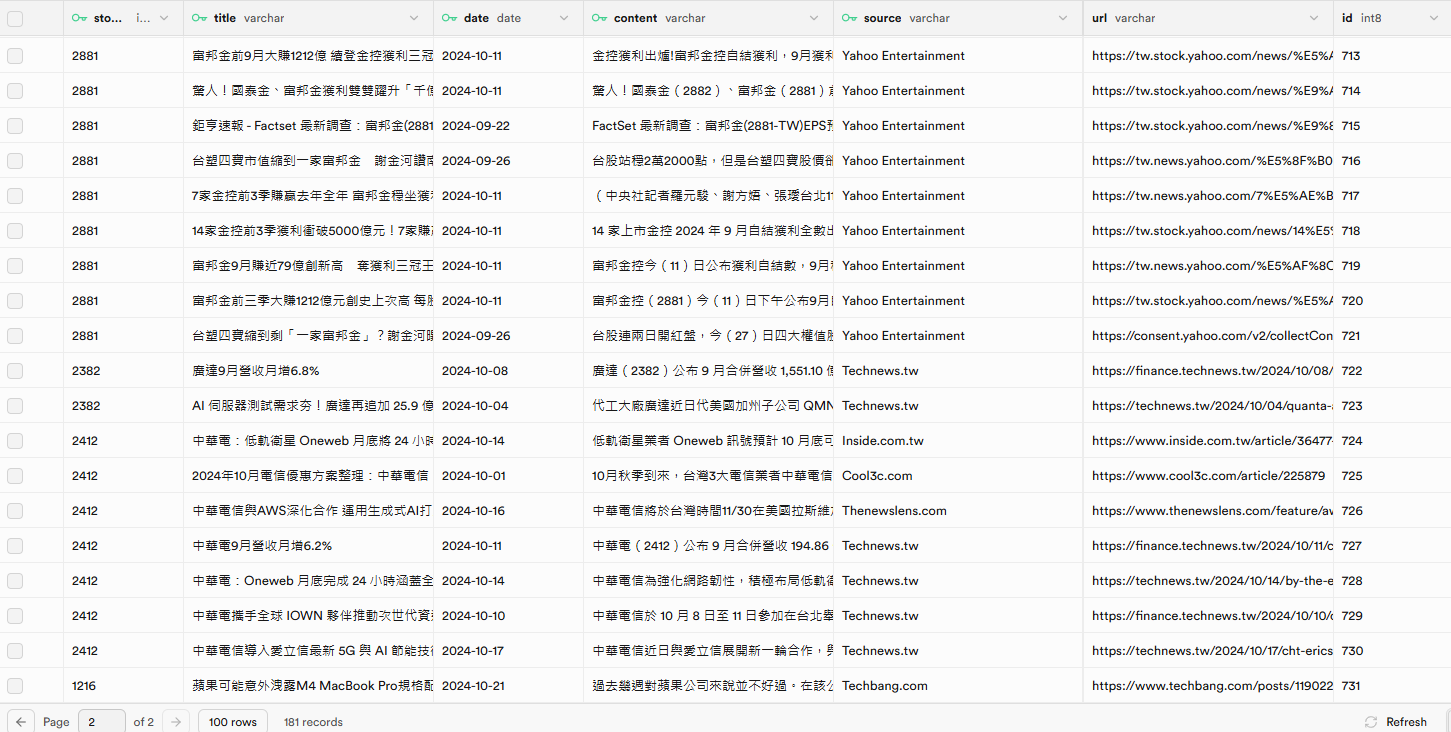
(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

**資料表 stock\_news\_API:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| stockID | title | date | content | source | id | url |
| 股票代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |



(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)



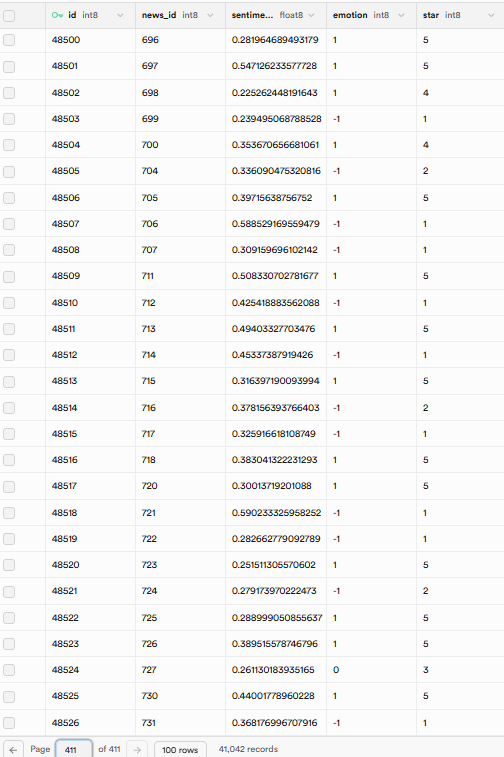
(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

**資料表 stock\_news\_sentiment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | news\_id | sentiment | emotion | star |
| index | 新聞編號 | 情緒分數 | 情緒類型 | 星等 |



(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)



(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

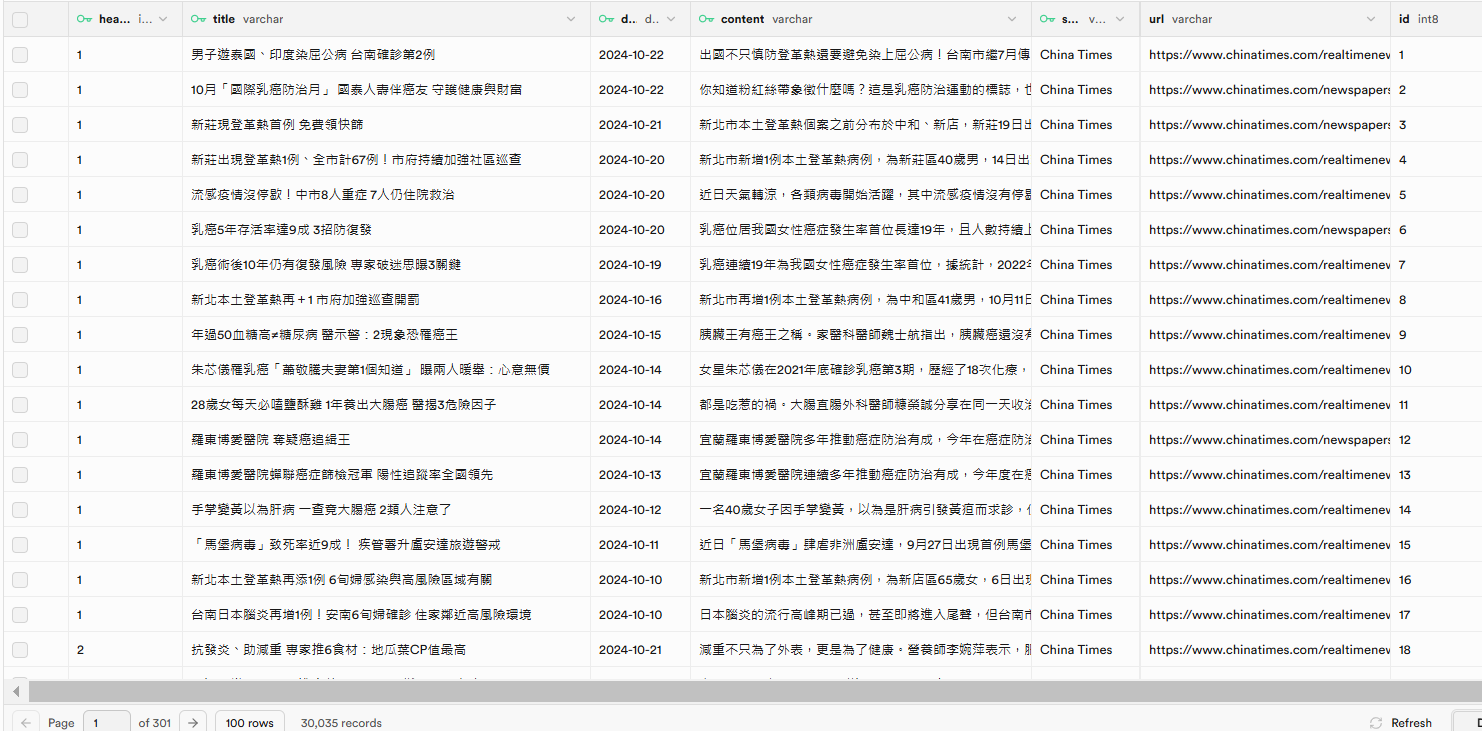
**資料表health:**

|  |  |
| --- | --- |
| healthID | health\_name |
| keyword代號 | keyword名稱 |



**資料表 health\_news:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| healthID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |

****

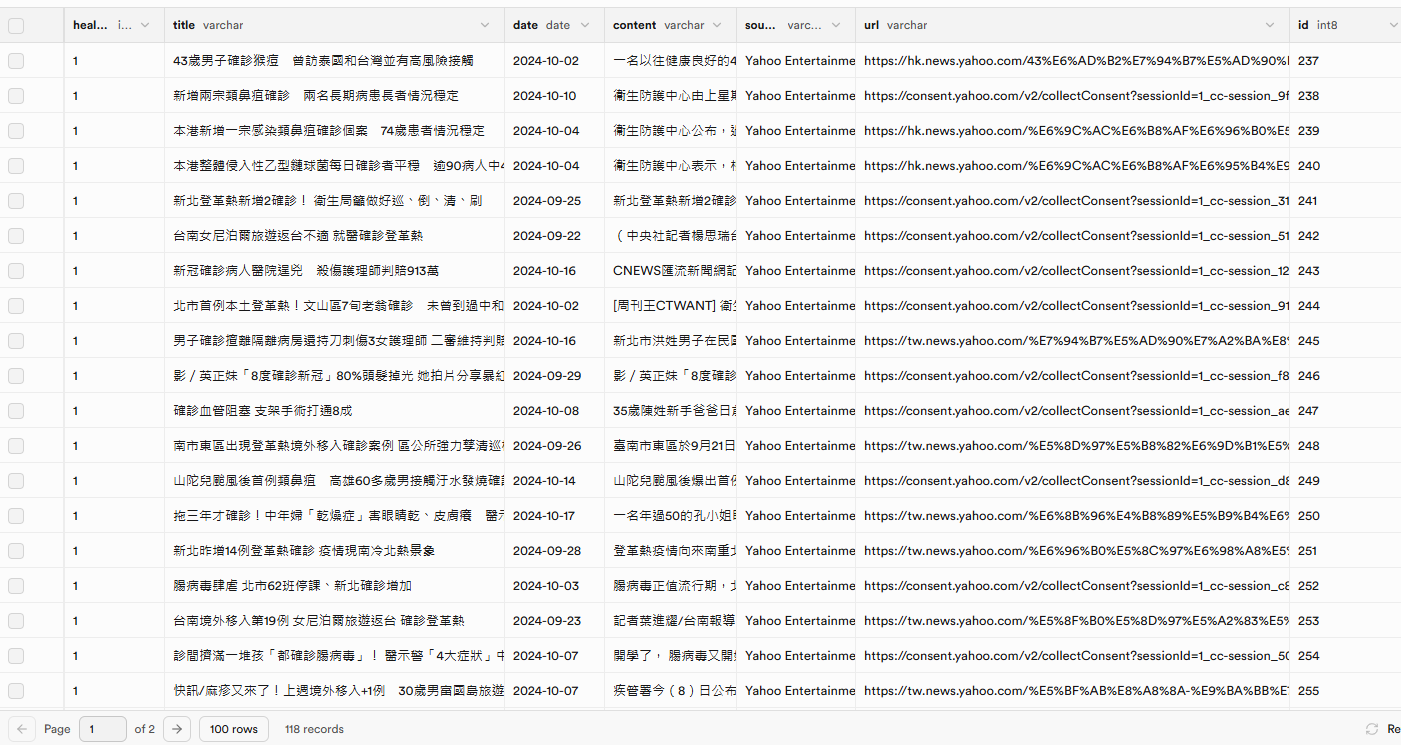
(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)

****

(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

**資料表 health\_news\_API:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| healthID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |



(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)



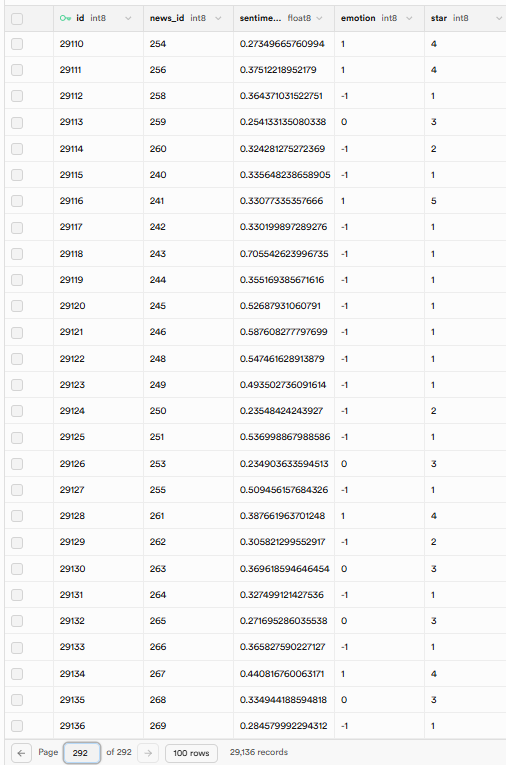
(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

**資料表 health\_news\_sentiment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | news\_id | sentiment | emotion | star |
| index | 新聞編號 | 情緒分數 | 情緒類型 | 星等 |



(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)



(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

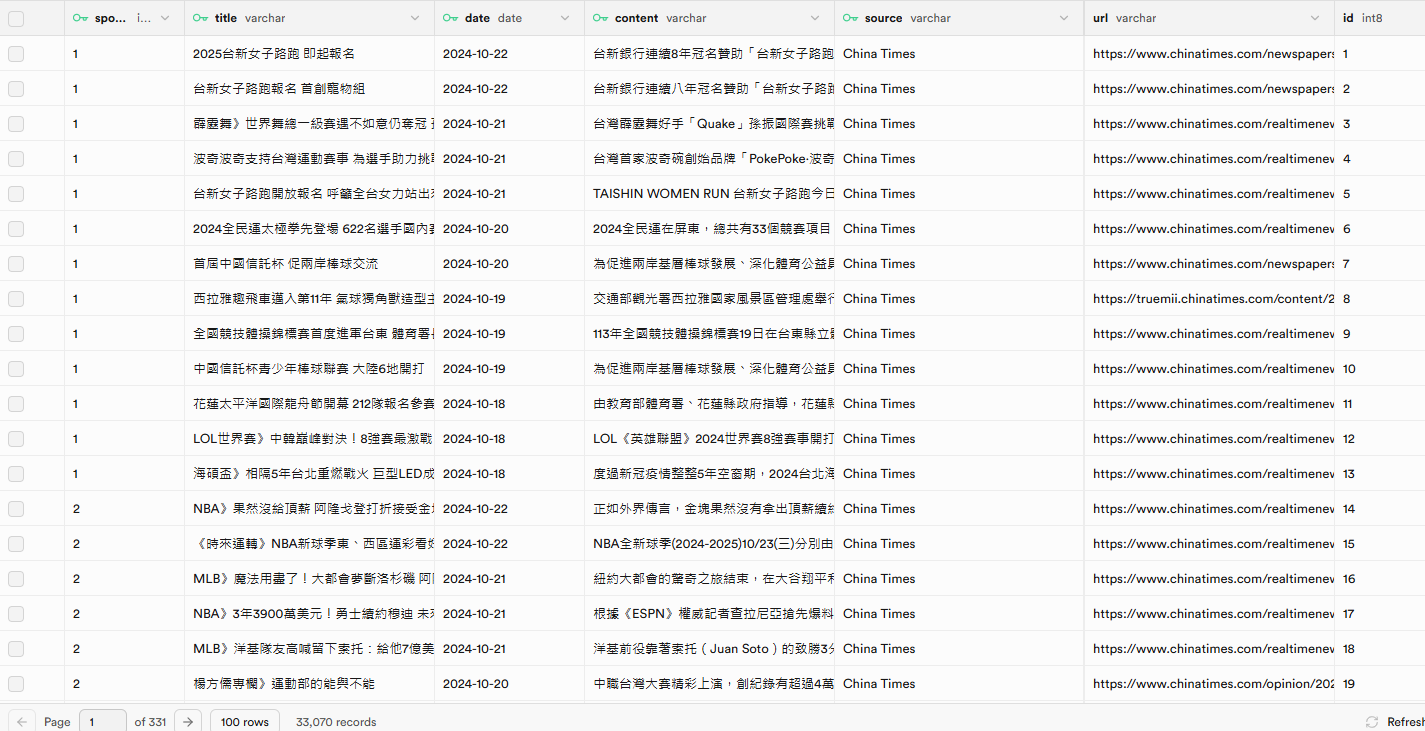
**資料表sport:**

|  |  |
| --- | --- |
| sportID | sport\_name |
| keyword代號 | keyword名稱 |

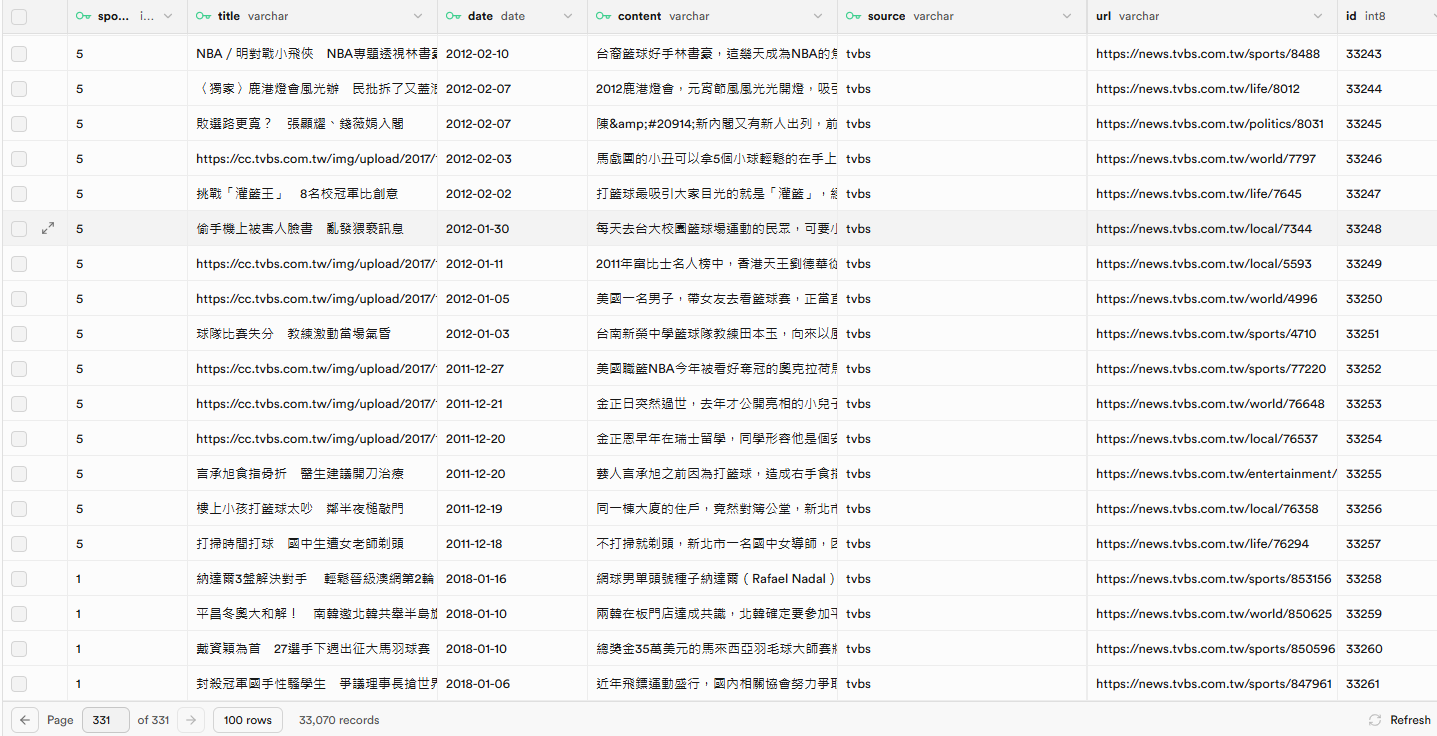


**資料表 sport\_news:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| sportID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |



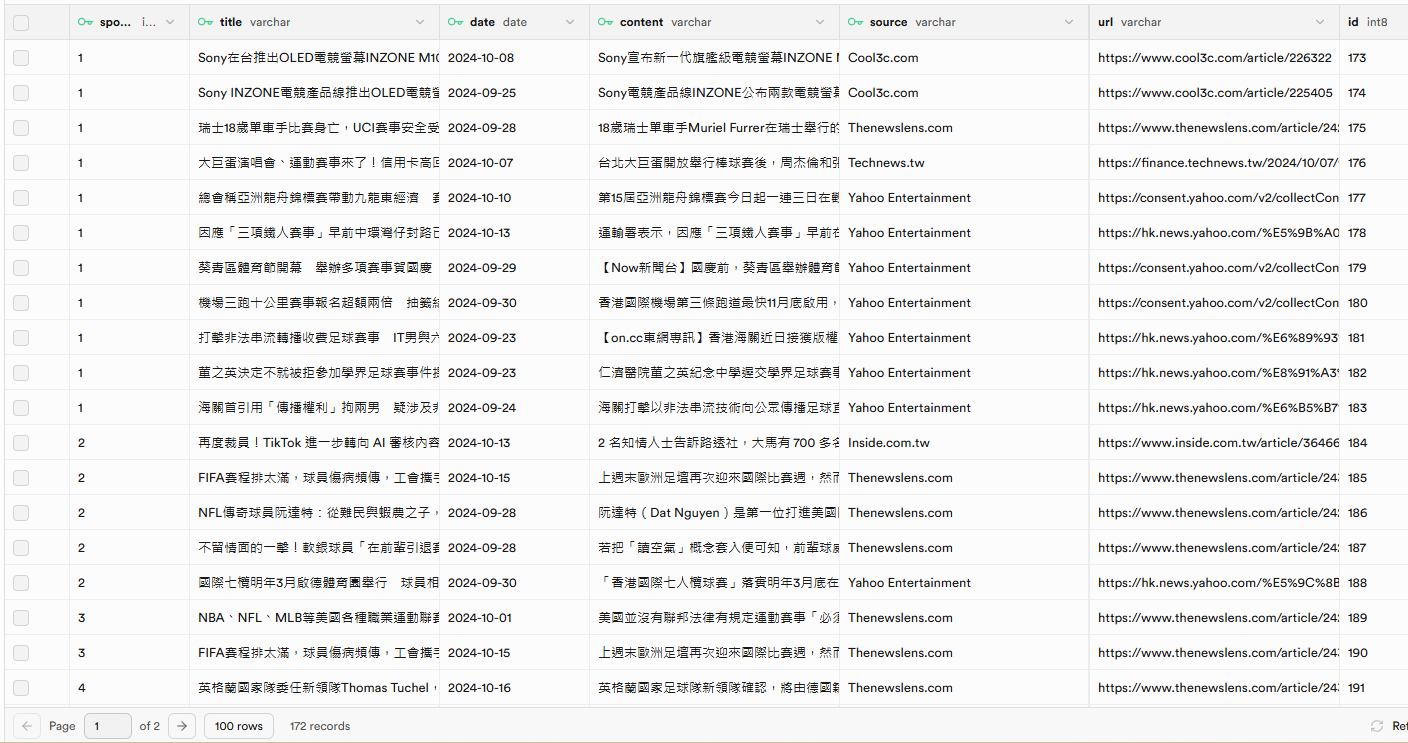
(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)



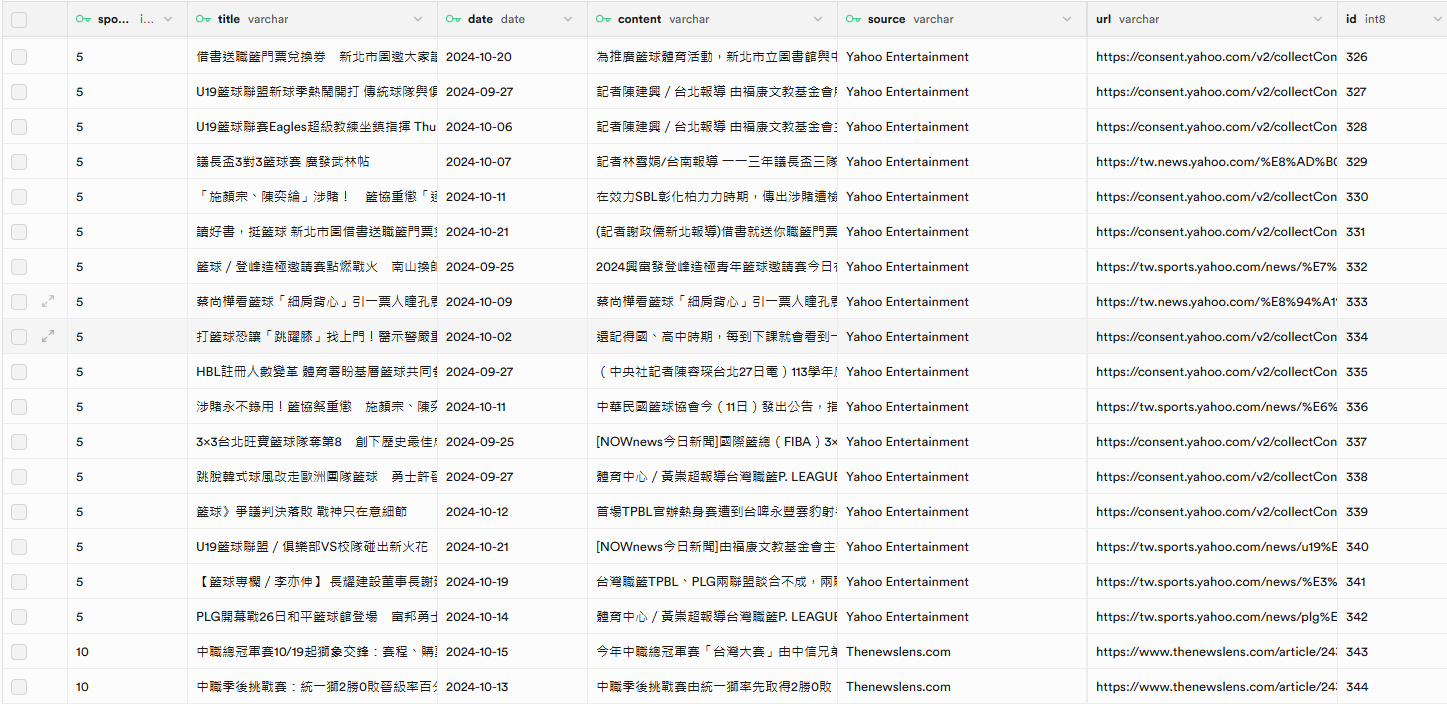
(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

**資料表 sport\_news\_API:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| sportID | title | date | content | source | id | url |
| keyword代號 | 標題 | 日期 | 內容 | 來源 | index | 連結 |



(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)



(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

**資料表 sport\_news\_sentiment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | news\_id | sentiment | emotion | star |
| index | 新聞編號 | 情緒分數 | 情緒類型 | 星等 |



(“IR\_GenAI\_Team\_Proj1”以id排序 第1頁)



(“IR\_GenAI\_Team\_Proj1”以id排序 最後一頁)

展示databse 瀏覽、搜尋功能(以health\_news為示範):

1. 資料以date排序

SELECT \*

FROM health\_news

ORDER BY date ASC;



1. 找出資料表health\_news和health\_news\_API中的 source

SELECT DISTINCT source

FROM health\_news

UNION

SELECT DISTINCT source

FROM "health\_news\_API";



1. 資料以id 做升序排序

SELECT \*

FROM health\_news

ORDER BY id ASC;



1. Visualization

在視覺化的部分，我們主要針對情緒分析與網路爬蟲皆有做出使用者介面，以下是網路爬蟲的使用者介面，可以透過選取欲爬的新聞網(中國時報、TVBS、LTN、API)與選擇要爬取的主題(Stock、Helth、Sport)與遇爬取的頁數(此頁數會從最新新聞往後爬取)，最後動態更新到資料庫!



在情緒分析視覺化的部分，我們在前端使用三個圖表呈現。使用者在網頁上選擇主題 (Health、Sport、Stock)、新聞來源及時間區間後，按下確認即可即時生成以下三張圖表：圖一為情緒分布圖，顯示正面、中立和負面情緒數量(positive、neutral、negative)；圖二為star數；圖三為時間序列圖，展示情緒隨時間的變化。

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 文字, 圖表, 繪圖 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 行, 繪圖 的圖片

自動產生的描述

以下針對三個測試範例做說明：

一張含有 文字, 螢幕擷取畫面, 繪圖, 行 的圖片

自動產生的描述

這是health 2018~2020，由此顯示人民對健康的相關議題愈來愈有信心，因為技術的上升，其中2019年是分界點，因為在疫情下的台灣，展現出極高的防禦心理，造成新聞多為正面報導。

一張含有 文字, 螢幕擷取畫面, 行, 繪圖 的圖片

自動產生的描述

由此圖可以顯示，由於近2024年奧運，因此有關於運動的相關議題，皆於2024開始情緒上漲，並於2024 8月屬於高峰值群。

一張含有 文字, 字型, 行, 螢幕擷取畫面 的圖片

自動產生的描述

股市方面，可以發現是屬於動盪較大的情緒指標，這會與當時刻的股市大盤走向有關，而最近因為戰爭、天災等等...因素，因此情緒指標走跌。

而以下是網頁的其他網頁(包含首頁，與介紹如何使用):

首頁:



關於:



心得:

A1105505 林彧頎

在本次的專題中，我主要負責分配工作、整合、web、word、ppt與爬蟲主要的操作，我在本次的專題中，學習到很多，最主要於爬蟲的部份我們要學習如何可以不間斷的一直爬蟲，並使其可以自適應與動態爬蟲，在與老師對談的過程中，我了解到，動態爬蟲並非只單一指selenium，而是有許多操作，我們在撰寫程式碼時，除了思考到要如何更加動態，也有去觀察不同網頁本身是因為怎麼的原因容易被擋，在過程中，我認為我最深刻的是，生成式AI的極限性，在過程中，有許多組向我詢問一些想法，因為我們這組是最早開始動工的，我認為如果設計一個生成式AI可以動態偵測網路架構並修改，是十分不切實際的，生成式AI僅能在小部分幫助很大，但當達到整個網頁架構時，他就會受限於字數限制、判斷能力，而無法給出最好的方法，我認為最好的做法，是在程式中，適時加入GenAI的判斷，像是針對非相關的新聞判斷，簡易判斷網頁結構的改變，與改善爬蟲時，向GenAI詢問等等…，這些我認為才是最好與GenAI溝通的方法，另外再寫網頁的部分，因為我希望可以達成可以長時間爬蟲、動態爬蟲、並且資料量大，因此我不追求時間最長(因為過長其實沒有用，以近期的資料愈好)，因此最長可以爬到2003年，此程式碼，我讓它架設在server上跑整整2星期，由此可見此程式是如此不被網站偵測為機器人，另外我們使用動態設定頁數的方式給使用者，也是為了讓使用者可以動態爬取尚未爬到的資料，最後在資料庫的部分，我們為了讓組員皆可以使用此database因此我使用supabase的雲端資料庫，但此資料庫不太方便公開給他人(**僅可以加入特定使用者，因此我有用.sql的方式提供整個資料庫，如助教或老師需要觀看此資料庫，可以提供給我github帳號，我將提供整個資料庫給老師或助教看**)。  
以下將簡述一些爬蟲學習到的事:

在進行新聞資料爬取的過程中，我們先準備了一組常用的 User-Agent 列表，讓每次 request 都能隨機選用其中一個 User-Agent，藉此減少因使用固定 User-Agent 而被網站偵測為爬蟲的風險。此外，我們還利用了 Chrome WebDriver 的 execute\_cdp\_cmd 方法，把 navigator.webdriver 屬性設為 undefined，以隱藏 WebDriver 的使用。還有使用生成式AI與css selector等作適應性變化。甚至我們為了讓爬蟲更加多元，可以來源更多，因此我們還利用API的方法，還有許多沒敘述的方法請詳細參閱前面章節。

A1105524 吳雨宣

在爬取中時的新聞時，我們定義了一組常見的 User-Agent 列表，每次request時都隨機選一個 User-Agent ，避免因為固定的 User-Agent 而被偵測為爬蟲。除此之外，還使用 Chrome WebDriver 中的 execute\_cdp\_cmd 方法，將 navigator.webdriver 屬性設定為 undefined ，這可以隱藏 WebDriver 的使用。本來想說既然中時用這些機制去解決被偵測為爬蟲的問題，同樣身為新聞網的自由時報應該也可以用這種方式吧，沒想到中時的爬蟲架構竟然不適用於自由時報。因為中時的部分，我們是直接在搜尋結果出來的那頁就直接抓取我們所需的新聞資料，發現自由時報不是用這種方式後，我們就改成在搜尋結果那頁先將所有的新聞連結都存起來，再透過新聞連結用 BeautifulSoup 的方式去抓取內文、日期等。而 TVBS 的部分雖然也是在搜尋結果那個頁面直接抓取需要的資料，但我是用 requests 和 BeautifulSoup 來進行網頁請求和解析。爬了三間新聞網後，發現爬蟲真的不是一件簡單的事，因為每個網頁的架構都不一樣，雖然生程式AI可以給我們大概的爬蟲架構，但很多細節都要人類自己去觀察修正，像是欲爬取資料的位置、網頁的架構等。這次的專題在爬蟲方面，我們真的花了很多心力，但也在這一次次的爬蟲中，對網頁架構和反爬蟲機制有更清楚的認識。

A1105507 蘇柏諺  
我覺得這次專題實作讓我學到最多的是如何用transformer精準的判斷sentiment，一開始我們使用Gemini等模型測試，但精準度、處理時間和對中文的理解都有很大的差異，特別是這次我們要處理幾十萬筆的新聞分析，所以實作時也要把龐大的資料量考慮進去，最後我們使用"nlptown/bert-base-multilingual-uncased-sentiment"模型，很順利的處理完所有新聞的情緒分析，讓我感到很有成就感，另外為了儲存資料並方便小組開發，我們學會使用supabase這個功能強大且好用的共用資料庫，也讓我覺得這次的實作受益良多。

A1105523 巫柔筠

這次的專題我負責的是，情緒分析的部分，我們一開始試了很多不同的方式跟模型，最後選擇的模型是"nlptown/bert-base-multilingual-uncased-sentiment"，這個模型會給星等，我們再利用五顆星等，去分成負面、中立、正面，我們也有嘗試做一些微調，但結果並沒有原本模型來的好，所以沒有更改太多參數。我主要是處理前端，可以用日期的區間、主題和來源去查詢統計的數據資料，一開始沒有注意到，因為我們的資料量龐大，所以用了兩個資料庫，這兩個資料庫的id會有重複，跑出來的結果會有誤，之後做了修正。這次的專題對於情緒分析又有更深入的了解，在前端跟資料的處理也學到了更多的技巧，這些經驗對於未來的專題都有很大的幫助。

A1105521 黎子崴

在期中專題我主要參與了情緒分析與視覺化的部分，目標是通過情緒分析來理解新聞文章的情感傾向。這個專題實作讓我有機會深入認識NLP技術，使用工具來解析不同主題的情緒變化。我們選用了 transformers 庫中的預訓練模型BERT進行文本情緒分類。一開始，我們在幾個不同版本的BERT預訓練模型來選擇，最後選擇nlptown/bert-base-multilingual-uncased-sentiment。原因是此版本已經基於龐大的資料集進行了預訓練，能更好的捕捉語言中的情緒表達特徵，且無須多做微調便能符合新聞情緒分析的情緒分類需求。視覺化的部分我們可以基於三種不同的資料組合生成圖表，多方呈現情緒變化和趨勢。

另外，此專題需要處理大量的新聞數據並進行情緒分析，我們選擇了 Supabase 來管理和儲存情緒分析結果。Supabase 是開源的BaaS ，它提供了方便的資料庫管理功能，讓我們能夠輕鬆儲存並查詢分析結果，同時可以透過 API 請求將情緒分析結果和前端的應用融合。

在前端方面，我們使用了 Flask 框架來構建 API 和管理資料流，使前端可以通過簡單的 API 請求即時獲取情緒分析結果。使用者能透過前端即時查看不同時間範圍、新聞來源的情緒趨勢，更直觀地理解情緒分析結果。