中華民國第 65 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

第三名

052513

結合 LSH 及知識圖譜改善 RAG

學校名稱： 新北市立新莊高級中學

作者：	指導老師：
高二 宋宣錕 高二 林于竣 高二 郭品序	李文乾

關鍵詞： RAG、LSH、知識圖譜

摘要

本研究透過結合局部敏感雜湊（Locality-Sensitive Hashing, LSH）與知識圖譜（Knowledge Graph, KG）以改善檢索增強生成技術（Retrieval-Augmented Generation, RAG）在檢索的時候難以在精確度與效率之間取得平衡的問題。透過使用 LSH 將資料分桶，接著利用知識圖譜進一步篩選資料，以提高檢索的精確度與效率。

實驗結果顯示，結合知識圖譜與 LSH 後的系統，在精確度（precision）上可達到 91%，相較於 VectorRAG 的 84%提升約 8.33%，與 GraphRAG 的 96%則僅有 5.21%的差距。此外，本系統在檢索時間上較 GraphRAG 降低了 95.38%。由此可以證實，透過結合 LSH 及知識圖譜能在保持高精確度的同時，顯著提高檢索效率。

壹、 前言

一、研究動機

檢索增強生成（Retrieval-Augmented Generation, RAG）是一種結合檢索系統與生成式 AI 的技術架構，能夠在生成回應的同時，使用資料庫中的相關資料進行回應，從而降低 AI 在生成回答的時候可能出現的「幻覺（hallucination）」，即產生不正確、虛構或與事實不符的內容。

現在，許多企業也開始將 AI 技術引入到內部流程中，例如文獻查詢、員工訓練、技術支援與輔助等。但此時企業也會面臨一些挑戰，像是企業內部往往擁有大量涉及商業機密或技術資料的敏感資訊，若將這些內容交由像是 ChatGPT 或 Gemini 等的雲端模型處理，則會有資料外洩的風險。

此外，雲端模型在正常情況下也沒辦法獲取企業內部最新的技術資料，因此也難以給出具體且有依據的答案。因此，部分企業已開始部署本地 AI，透過整合私有資料庫作為 RAG 的知識來源，從而打造專屬於企業的 AI 助手。這種方式不僅能保護資料不會外洩，也能讓 AI 根據員工所上傳的資料來進行回覆，使 AI 能夠獲取公司最新的技術文件，進而提高員工的工作效率。

但是，傳統的 RAG 在實作上仍然存在著許多的挑戰，像是向量的檢索雖然快速，但在處理語意模糊、需要推理的問題時，準確率的表現常常並不穩定。為了解決這個問題，已有研究開始使用知識圖譜（Knowledge Graph, KG）以加強其在語意、關係結構與邏輯推理上的能力，但這卻也讓系統變得更加複雜，進而造成查詢效率下降，使其難以應付需要快速回應的使用場景。

二、研究目的

本研究的目的主要有三個部分，首先設計一個能夠從文章中提取出關鍵字的系統，並將其存入資料庫內以構建知識圖譜。再來透過使用 LSH，使系統能快速將相似資料分類完畢，並在查詢的時候根據每個桶與問題的相關性進行排序。最後，將本系統與現有系統，如 VectorRAG、GraphRAG 等系統進行比較，以評估本研究方法的優勢與可行性。

(一) 提取關鍵字與知識圖譜構建

為提升知識圖譜構建的語意準確度，本研究將透過使用大語言模型（Large language model, LLM）來提取文章中的關鍵字，並透過使用提示詞（prompt），使其獲取的結果能夠聚焦於專有名詞，如人名、地名、技術名詞，排除無意義或過於通用的詞語，以確保每個關鍵字都具備實際的意義。而後會將這些關鍵字在文章中的關聯梳理出來，並用於構建知識圖譜。此外我們也將對不同的 LLM 進行比較，以找到能夠兼顧生成品質與硬體需求的模型。

(二) LSH 表建立與排序

本研究使用局部敏感雜湊（Locality-Sensitive Hashing, LSH）技術，以提升資訊檢索效率。透過 LSH 將資料映射到不同雜湊桶內，在查詢時，便可以根據各桶之間與問題的相似度進行排序，使系統能夠縮小搜尋範圍，降低檢索時間，提升整體的效率。

(三) 不同 RAG 之比較

為了驗證本研究所提出的方法可行性，本研究將與現有系統進行比較，包括僅使用向量檢索的 VectorRAG，以及以知識圖譜為主的 GraphRAG。比較項目包含查詢速度、預處理時間與資訊準確性等方面，並依此評估本研究方法的優勢。

三、文獻回顧

本章將針對本研究所涉及的關鍵技術進行深入分析與討論，包含 RAG、LSH、知識圖譜與 BERT。這些技術分別在知識檢索、語意理解、資料關聯與生成式 AI 中扮演重要的角色，同時也是本研究系統設計的基礎。

(一) RAG 架構概述與核心原理

傳統的生成式模型（如 GPT 或 BART）雖然能產生流暢的文句，但其知識來源只來自訓練資料，缺乏動態查詢外部知識的能力，因此容易出現幻覺的問題。因此 Patrick Lewis 等人（2020）[3]提出了 RAG（Retrieval-Augmented Generation）模型與架構，透過結合預訓練生成模型與非參數記憶（如檢索模組），讓模型在生成過程中能夠動態查詢外部知識，從而提升其在需要大量專業知識的自然語言處理任務的準確度。RAG 模型架構如圖 1 所示。

圖 1、RAG 模型架構，取自黃適文（2024）之文章《什麼是 RAG？初學者也看得懂的檢索增強生成（RAG）基礎指南！》

(二) 基於語意向量的 RAG 的應用與挑戰

嵌入向量檢索（Embedding-based Retrieval）為 RAG 架構中最基礎，且被廣泛使用的技術之一。透過 BERT 等語言模型，文本內容可被轉換為嵌入向量（embedding vectors），再經由相似度計算（如餘弦相似度）進行檢索，從而快速找出與查詢相關的內容。該方法具有操作簡單、查詢快速等優點，因此被廣泛應用於一般性問答系統中。

然而，相關研究指出，嵌入向量檢索在處理複雜或專業領域問題時，存在語意理解不足與推理能力缺乏的問題，使其檢索階段可能引入與查詢問題無關的資料，形成所謂的「雜訊 （noise）」。這些資料若未經過妥善的篩選，可能導致生成內容產生語意漂移（semanticdrift），進而影響輸出的品質。徐金廷（2024）[6]與林祐宸（2024）[7]均指出，單純依賴向量相似度可能導致查詢結果與使用者需求不符，尤其在處理具有明確邏輯結構或需跨領域語意連結的查詢任務時，其表現相對受限。因此，雖然語意向量技術具備一定的基礎能力，但仍需要以其他的技術手段，如重排序（reranking）、知識圖譜或多階段檢索策略，以提升其在深層語意理解與跨主題內容生成上的能力。

(三) 局部敏感雜湊於高效檢索中的應用

局部敏感雜湊（Locality-Sensitive Hashing, LSH）是高維空間中進行近似最近鄰（Approximate Nearest Neighbor, ANN）搜索的技術。其原理是透過雜湊函數，將高維資料映射到低維空間，使得相似的資料可以被映射到同一個雜湊桶中，藉此快速的將相似的資料分類到一起。

根據 Jafari 等人（2021）[2]的研究可知，LSH 在處理大規模資料時，不僅具有優秀的檢索效率，也廣泛應用於高維度數據的場景，如語意向量、圖像特徵與醫學資料等。

因此，我們推測，若將 LSH 引入包含知識圖譜的 RAG 系統中，將可在檢索階段就對資料進行初步的篩選，從而避免逐一比對所有資料，使其能夠降低檢索所需的時間。

(四) **知識圖譜導入 RAG 架構對效能影響的評估**

知識圖譜（Knowledge Graph, KG）是透過圖狀結構表達實體（Entity）與實體之間的語意關聯。而「實體」是指知識中具備獨立意義的事物，例如人名、地名、事件、組織或抽象的概念。

在圖譜中，每個實體都是一個節點，這些節點之間透過「關係（Relation）」彼此連結。而「關係」代表的是實體間的語意關聯。這種由實體與關係所組成的基本結構，稱為「三元組（Triple）」。

三元組由三個部分組成，分別是主詞、謂詞與賓詞。主詞是某個實體，謂詞表示的是這個實體與另一個實體之間的關係，而賓詞則是與主詞相關的另一個實體。例如，在「愛因斯坦是物理學家」這個三元組中，「愛因斯坦」是主詞、「是」是謂詞、「物理學家」是賓詞。

透過大量的三元組組合，知識圖譜能夠清楚呈現實體之間的語意關係，進而被用於語意理解、資訊整合與智慧推理等方面。圖 2 為知識圖譜示意圖。

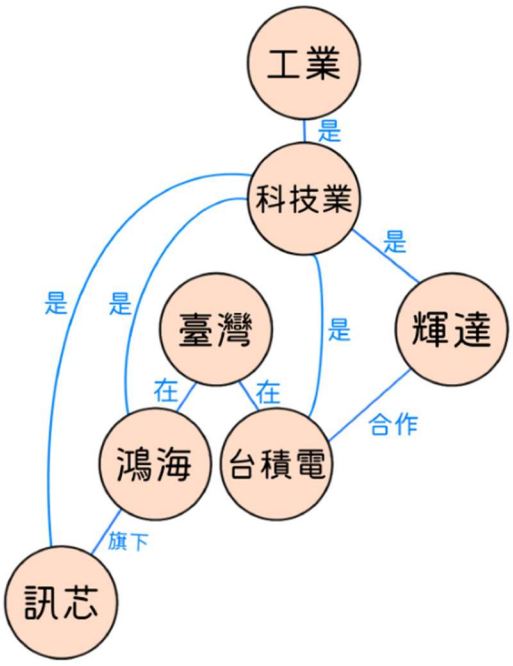

圖 2、知識圖譜示意圖（來源：自行製作）

根據 Edge 等人（2024）[1]的研究可以發現，結合知識圖譜的 GraphRAG 相較使用向量的 RAG，能更有效的捕捉文本之間的關聯性，進而使 LLM 能提供更全面且準確的回答。然而，此系統對硬體或雲端資源的需求也相對較高。以 Sindy_he（2025）[15]的測試結果為例，在使用 NVIDIA RTX 4090 部署 Deepseek-r1 32b 模型的情境下，處理 2 萬字的資料需花費約 3 小時的時間。而根據 fan84sunny（2024）[16]的測試也可以知道，在同樣的資料大小與設備情況下，RAG 只需要 30 秒就能回答出答案，但 GraphRAG 可能需要到 1 分半才能回答答案。Bowen Chiu（2024）[20]也表示：「雖然這種方法允許更細緻和具上下文感知的檢索，可以使大型語言模型產生更準確和全面的回應。」但他也表示：「雖然結果很有趣，但 GraphRAG 需要幾乎 10 倍的時間和 10 倍的 token 來生成」。此外，根據 huangyihe（2024）[17]的測試，在使用 GPT 4 turbo preview 的情況下，處理約三萬字的《AChristmasCarol》並進行一次回覆需花費 11 美元。由此可知，將知識圖譜引入 RAG 雖然可以提高精確度，但在預處理、檢索與生成階段皆需投入更高的計算與金錢成本。

(五) BERT 模型於語意向量檢索的貢獻與限制

在向量檢索技術中，向量化模型的準確性會直接影響檢索與內容生成的品質。Devlin 等人（2018）[5]所提出的 BERT（Bidirectional Encoder Representations from Transformers）模型。BERT 透過雙向注意力機制整合上下文資訊，顯著提升模型在語意理解上的精確度。

BERT 採用「遮蔽語言模型（Masked Language Model, MLM）」與「下一句預測（Next Sentence Prediction, NSP）」作為預訓練目標。透過 MLM，模型可學習單字在上下文中可能出現的位置與意義；而 NSP 則強化了模型在理解句子之間語意關聯的能力。這兩項設計使 BERT 能夠生成具上下文意義的高維語意向量。

貳、 研究設備器材

一、硬體：筆記型電腦 CPU : R9 9955HX、GPU : RTX 5070 Ti laptop、RAM : 48GB。

二、軟體

(一) Rust 3.18.0

以高效能與記憶體安全為特點的程式語言，本研究用於設計構建雜湊表的部分。

(二) Python 3.12.9

具有簡潔語法與豐富套件的程式語言，被廣泛應用於資料分析、機器學習等領域。本研究主要的系統架構皆使用 Python 進行編寫。

(三) Python 套件

1. Numba 0.61.2

   加速數值運算的即時編譯器，可將 Python 函數轉換為機器碼，並支援調用 CUDA 進行平行計算。

2. Yaspin 3.1.0

   終端指令列動畫提示器，用於顯示運行狀態。

3. Colorama 0.4.6

   使終端輸出支援彩色與格式化文字，提升可讀性。

4. Python-dotenv 1.1.0

   用於載入環境變數，以提高項目安全性。

5. Rich 14.0.0

   進階的終端格式化工具，支援色彩、表格、Markdown 與除錯輸出。

6. Google-genai 1.11.0

   Google 生成式 AI 的 Python SDK，用於調用 Gemini。

7. Openai 1.76.0

   OpenAI 的 Python SDK，支援 GPT 等模型的呼叫與管理。

8. Sentence-transformers 4.1.0

   基於 Transformer 模型的句子嵌入工具，用於語意相似度與向量檢索。

9. Neo4j 5.28.1

   Neo4j 圖形資料庫的 Python 驅動程式，用於構建知識圖譜與節點與關係查詢。

10. Chromadb 1.0.7

    輕量級嵌入向量資料庫，支援相似度搜尋與語意查詢。

11. Flask 3.0.2

    輕量級 Web 框架，用於建構 API 與網頁服務。

12. Flask-cors 5.0.1

    Flask 擴充套件，用於處理跨來源資源共享（CORS）設定。

13. Flask-socketio 5.3.6

    使 Flask 支援 WebSocket 通訊，用於即時互動應用。

14. python-socketio 5.11.1

    Socket.IO 的 Python 客戶端與伺服器實作。

15. eventlet 0.35.2

    非同步網路程式庫，用於與 Flask-SocketIO 搭配實現高併發伺服器。

16. Maturin 1.8.3

    Rust 與 Python 的橋接工具，用於打包並發佈基於 Rust 編寫的 Python 擴充模組。

(四) LM Studio 0.3.16

用於部署本地端大型語言模型，支援模型下載、API 伺服與自定義提示，適合進行離線語言模型應用開發與測試。

參、 研究過程與方法

為了讓整體流程更清楚並方便後續處理，本研究將系統分為「預處理」與「生成」兩個階段。

一、預處理階段

在預處理階段，本系統會依照以下方式進行資料處理，以利後續語意分析與高效檢索（圖 3 為關鍵字與關聯性獲取示意圖，圖 4 為預處理階段流程圖）。

(一) 關鍵字與摘要生成

首先將輸入的資料交給 LLM 進行分析，提取出具有實際意義的關鍵字與進行內容摘要。

(二) 關鍵字語意關聯分析

將資料與獲取的關鍵字交給 LLM 進行分析，找出關鍵字在原文之間的關聯為何，並構建出對應的三元組。圖 3 為關鍵字與關聯性獲取示意圖

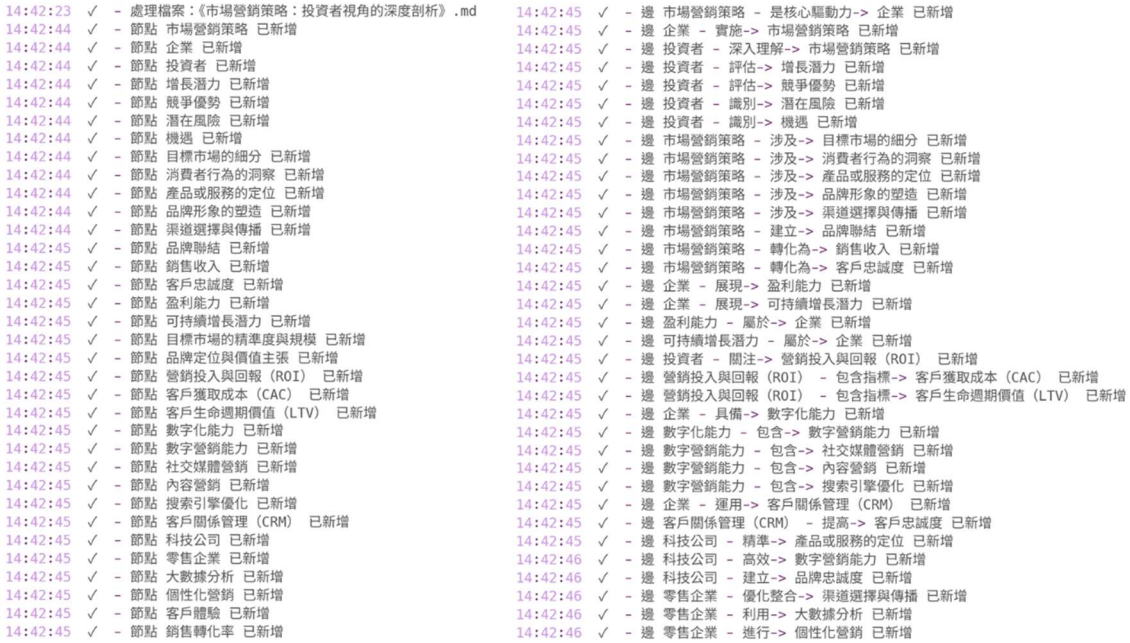

圖 3、關鍵字與關聯性獲取示意圖（來源：自行製作）

(三) 關鍵字翻譯

為了讓系統在處理多語言資料時保持高精確度，我們會將非專有名詞統一翻譯成英文，而專有名詞，如人名、地名、公司名稱，則保持不變。

(四) 資料向量化

將資料與獲取的關鍵字進行向量化處理，並儲存至向量資料庫。此外，為確保模型在處理中文資料時仍能維持高準確度與穩定性，本研究選用由北京智源研究院（2023）[21]開發的 BAAI/bge-large-zh 進行測試。

(五) 其他

將檔案的完整路徑、檔名與判斷後的關鍵字存入資料庫中，以利後續查詢使用。

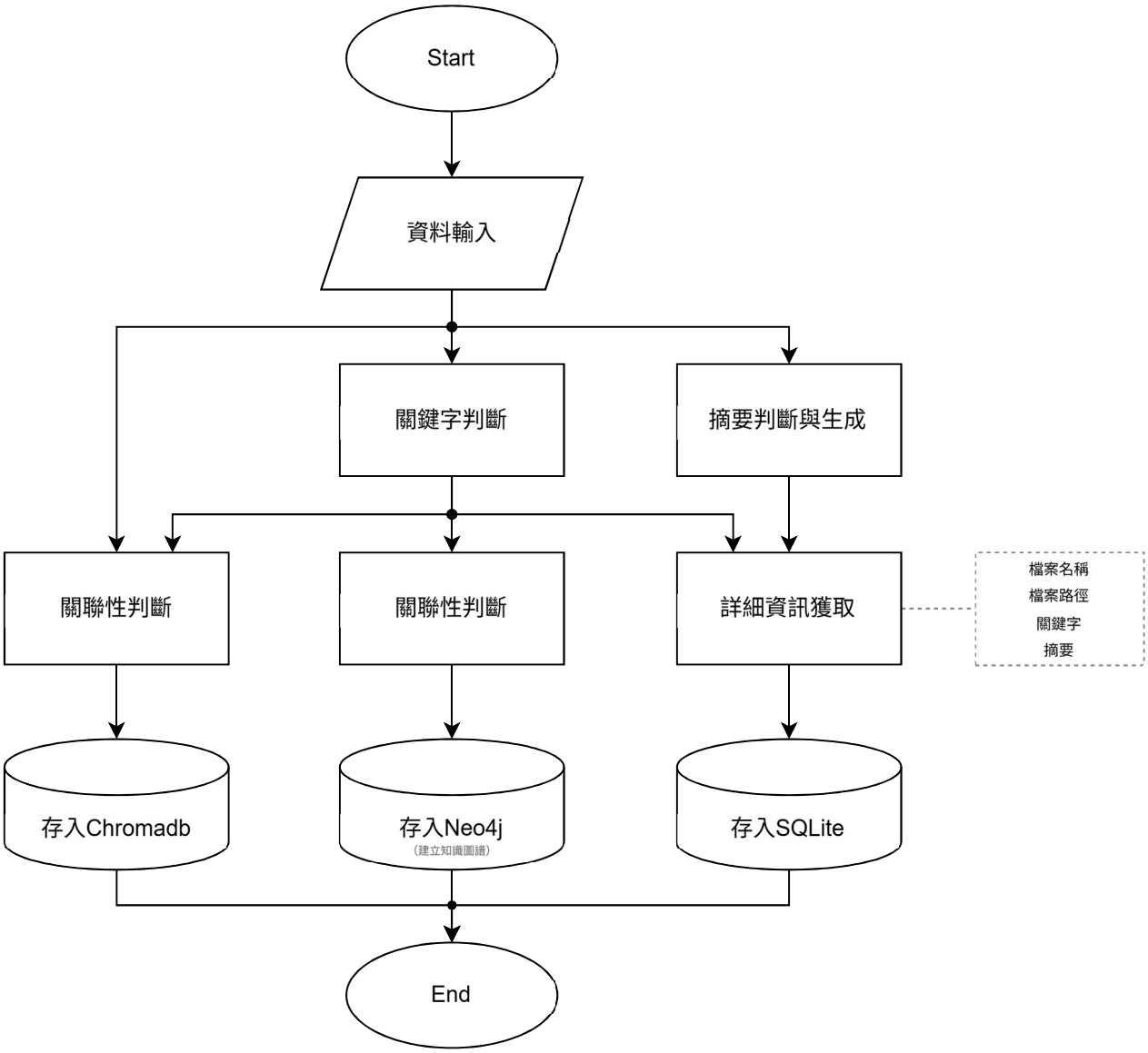

圖 4、預處理流程圖（來源：自行製作）

二、生成階段

在生成階段中，本研究進一步將流程細分為「檢索」與「生成」兩個子階段，而在檢索又可以分為關鍵字判斷、雜湊表處理、資料篩選與重排序四部分。這樣我們將能更清楚的掌握每個階段所耗費的時間，並針對特定環節進行調整與改進。圖 6 為生成階段流程圖。

(一) 檢索

1. 關鍵字判斷

在關鍵字判斷我們分為三部分，首先會透過 LLM 判斷使用者所詢問的問題需要什麼關鍵字，例如輸入「介紹一下 Docker 容器化部署」，此時就可以輸出「Docker 容器化」。這樣在處理複雜問題的時候可以避免僅使用向量尋找相關關鍵字仍可能出現噪音的情況。而後，會根據 LLM 所判斷出來的關鍵字使用向量搜索找到與其最接近的三個關鍵字。例如資料庫中有「Docker」這個關鍵字的時候，此階段就能夠獲取到此關鍵字與另外兩個相似度最高的關鍵字。最後會將找到的關鍵字使用知識圖譜向下三層找到與其相關的 5 個關鍵字。

2. 雜湊表建立與排序

在此部分我們會先取得全部資料的向量，並透過一組降維矩陣對其進行降維。系統會根據文檔數量（N）和向量維度（D）自動計算適當的雜湊表數量，其中 N 代表文檔總數，D 代表向量維度，tables 代表雜湊表數量，hashes_per_table 代表每個雜湊表中的雜湊函數數量。

$$tables = max(2, log_2(N))$$

$$hashes\_per\_table = max(2, log_2(D))$$

接著，對每個文檔向量（v），使用隨機投影矩陣 proj_matrix 進行降維映射，其中 proj_matrix 是從標準正態分佈生成的隨機投影矩陣，sign 函數將結果轉換為二進制。這樣就可以得到一個已初步根據相似將文檔分類到一起的雜湊表。

$$h(v) = sign(proj\_matrix \cdot v)$$

此外，在建立雜湊桶的同時，也會計算每個桶內平均向量（即質心），以方便後續能根據問題與質心的相似度進行快速排序。

而在這部分，為了在建構雜湊表能夠有更快的速度，本研究使用 Rust 來實現此部分，而質心計算的部分則使用 CUDA 進行計算，以有效處理大量向量資料的平均計算任務。再來當問題輸入後，將其向量化，並將根據其與每個桶的相關性進行排序，將可得到一組依相似度排序的雜湊桶。

3. 資料篩選

在資料篩選階段，本研究採用了動態調整的篩選機制。首先，設定一個初始的篩選條件（0.7），作為文檔篩選的基準值。當系統處理每個 LSH 桶中的文檔時，會分析文檔中的關鍵字，並計算這些關鍵字與搜索時找到的關鍵字有多少相同，如果有交集的關鍵字超過 2 個，這個文檔就會被加入到候選文檔集合中。系統會根據處理結果動態調整篩選條件：如果當前桶中符合條件的文檔比例達到 70%，且已經收集到超過 50 個候選文檔，系統就會停止搜索；如果沒有達到這個標準，系統會逐步降低篩選條件，繼續搜索更多資料。

4. 重排序

最後，我們會使用重排序模型，將篩選出來的資料依照與問題的相關性進行排序，讓相關性最高的內容排在最前面。這樣不僅能幫助 LLM 在生成回應時，可以先得到最重要的資訊，也能讓使用者在閱讀時，按照資料的重要程度依序瀏覽，以提升整體的實用性。此外，為確保模型在處理中文資料時仍能維持高準確度與穩定性，本研究選用由北京智源研究院（2023）[22]開發的 BAAI/bge-reranker-large 進行測試。

(二) 生成

為了讓使用者可以快速了解檢索到的資訊，在檢索完後，我們會將檢索到的資料的摘要交給 LLM 生成回應。然而，若沒有對其進行限制，LLM 會傾向依賴這些資料進行回答，從而缺乏對資料正確性與一致性的判斷能力，導致模型會將錯誤的資訊直接加入回應中。因此，我們在提供參考內容前，需要透過調整 prompt，以引導模型在回答之前能夠進行初步的判斷，避免傳達錯誤的資訊。圖 5 為生成階段提示詞。

```
你是一位專業的 AI 助手。請根據提供的上下文信息回答用戶的問題，並自動補充相關的專業知識和背景資訊。回答應該詳細、完整，
並使用 Markdown 格式來美化排版。

回答要求：

1. 內容要求：
   * 優先使用提供的上下文信息作為回答的主要依據
   * 回答必須詳細，控制在 500-800 字之間
   * 自動補充相關的專業知識和背景資訊
   * 保持專業且客觀的立場
   * 內容應該深入且全面

2. 知識補充要求：
   * 自動補充相關的專業術語解釋
   * 補充相關的技術背景和發展歷程
   * 提供實際應用案例或最佳實踐
   * 補充相關的技術趨勢或最新發展
   * 確保補充內容的準確性和專業性

3. 格式要求：
   * 使用 Markdown 格式進行排版
   * 使用標題、小標題等基本格式
   * 使用列表或項目符號來呈現關鍵點
   * 確保排版簡潔清晰
   * 適當使用粗體或斜體強調重要內容

4. 結構要求：
   * 開頭直接回答核心問題
   * 使用多個小標題分隔主要部分
   * 每個部分都應該詳細闡述
   * 使用簡短的列表呈現要點
   * 結尾應該有總結性的內容

5. 補充原則：
   * 補充的內容必須與問題高度相關
   * 確保補充的知識準確且有用
   * 補充內容應該豐富且深入
   * 保持內容的邏輯性和連貫性
   * 適當引用權威來源或最佳實踐
```

圖 5、生成階段 prompt 設計（來源：自行製作）

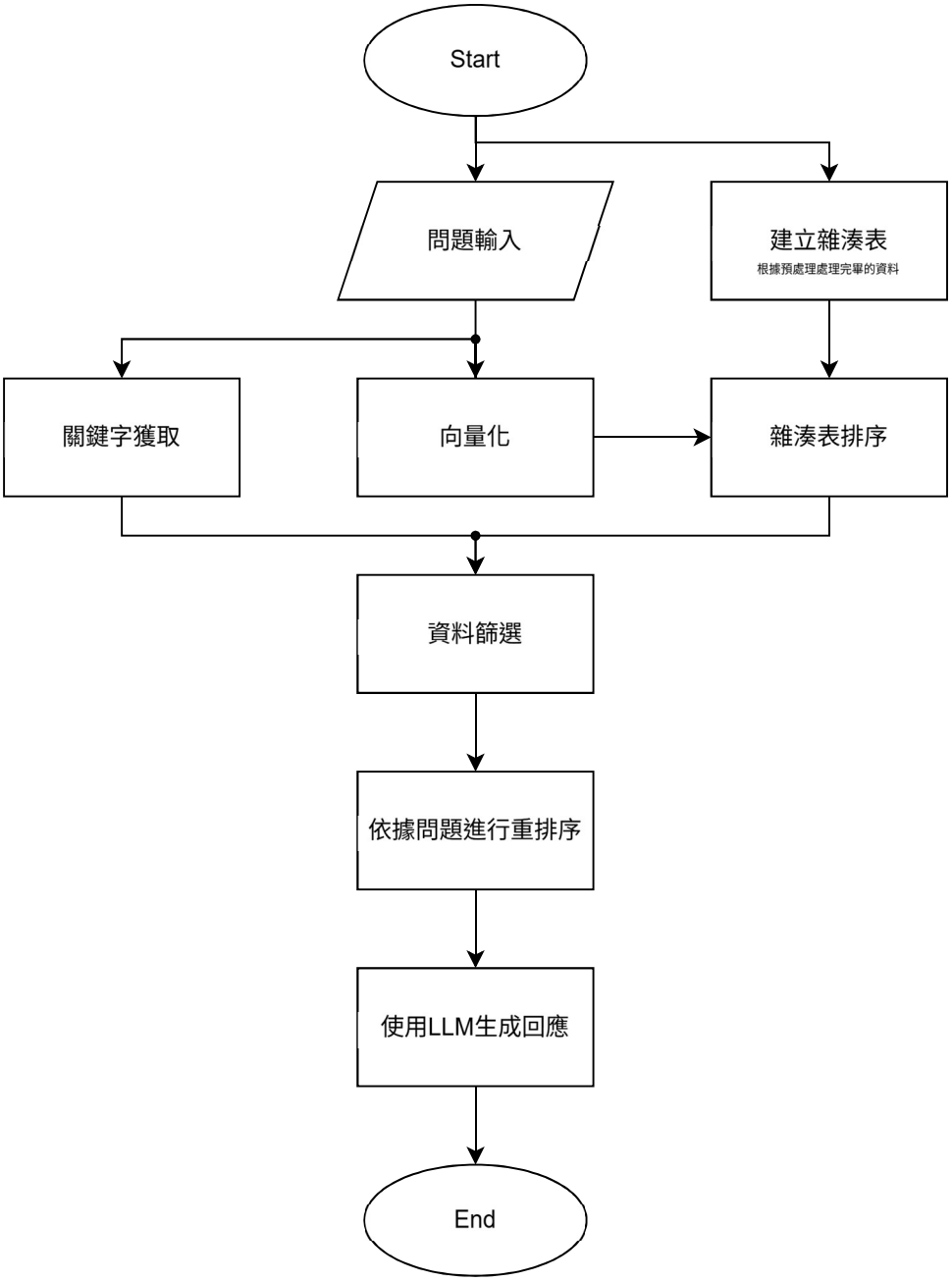

圖 6、生成階段流程圖（來源：自行製作）

肆、 研究結果

設計完系統後，我們透過使用 LLM 分別對不同 LLM 在關鍵字與關聯性提取上、不同 RAG 系統的 precision 與 recall、預處理方面及檢索時間的評估，具體評估方式如下。

一、關鍵字與關聯性評估

在此部分，我們會隨機抽取 10 筆資料進行測試，並分別對以下五點進行評估，最後將其進行平均，取得最終的平均分數。

(一) 關鍵字完整性（completeness）

(二) 關鍵字準確性（accuracy）

(三) 關鍵字分類準確性（classification accuracy）

(四) 關係分析準確性（relationship accuracy）

(五) 翻譯準確性（translation accuracy）

在此部分，我們將對五個模型進行詳細比較，分別為：Gemini 2.5 Flash thinking = 0、Gemini 2.5 Flash thinking = 4096、Deepseek R1 0528 Qwen3 8b，以及 Gemma3 的 4b QAT 與 12b QAT 版本（如表 1、表 2）。

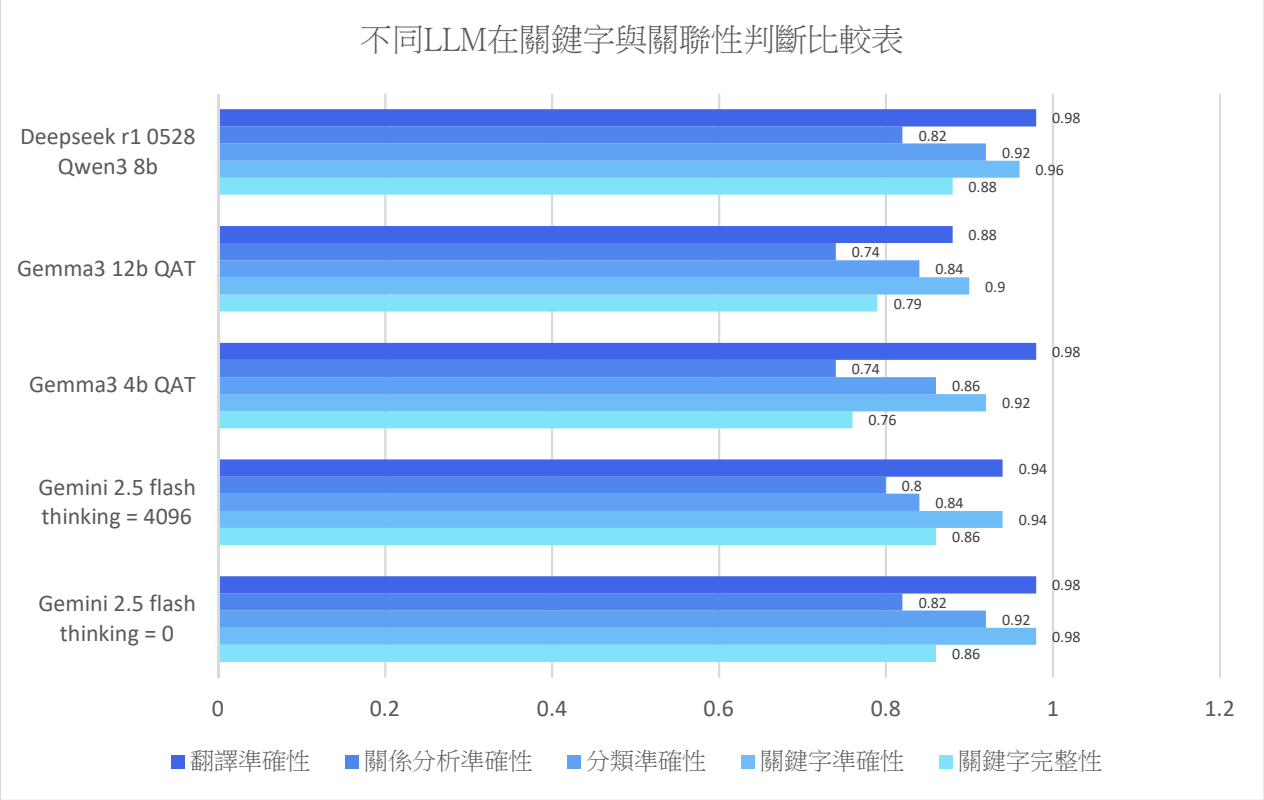

表 1、不同 LLM 在關鍵字與關聯性判斷比較表（來源：自行製作）

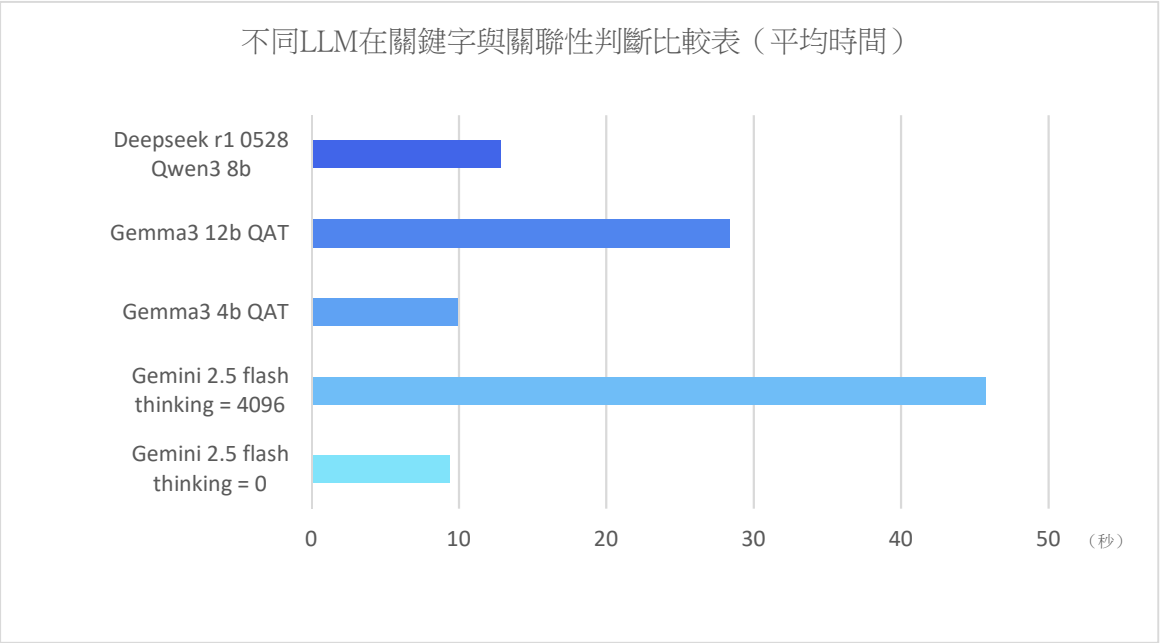

表 2、不同 LLM 在關鍵字與關聯性判斷比較表（平均時間）（來源：自行製作）

根據實驗數據可發現，表現最佳的模型為未開啟 thinking 模式的 Gemini 2.5 flash 以及 Deepseek r1 0528 Qwen3 8b。其中，Gemini 2.5 flash 在未啟用 thinking 模式時，其平均分數可高達 91%，且執行時間僅有 9.39 秒。然而，當開啟 Gemini 2.5 flash 的 thinking 模式後，平均分數反而下降至 88%，而執行時間則大幅增加至 45.71 秒，約為原來的 5 倍。我們推測可能是因為在 thinking 模式時，模型會因為 prompt 的設定而進行更多的推理與判斷，反而會導致過度思考，進而影響到整體的分數。

而在本地模型的部分，我們發現表現最好的是 Deepseek r1 0528 Qwen3 8b，而非參數量最大的 Gemma3 12b QAT。這也顯示出模型的參數量並非影響模型表現的唯一因素。儘管 Gemma3 12b QAT 擁有較大的模型規模，但在實驗中的平均分數僅有 82%，且執行時間高達 28.39 秒。在平均分數上與 Deepseek r1 0528 Qwen3 8b 有著 10%的差距，而執行時間上卻提高了約 2.2 倍。

因此我們可以總結，若需要本地部署，且硬體資源允許的情況下，使用 Deepseek r1 0528 Qwen3 8b 是相對最佳的選擇，而若顯存較為有限，則可考慮使用 Gemma3 4b QAT，這樣將能在效能與資源之間取得平衡。

二、precision 與 recall 評估

(一) precision

在計算 precision 的部分，我們將資料分為三種類型，分別為：

1. 完全相關：資料直接回答了問題的核心內容
2. 部分相關：資料提供了一些相關信息，但不足以完全回答問題
3. 完全不相關：資料與問題完全無關

最後計算 $\frac{\text{完全相關+部分相關}}{\text{總文檔數}}$ 便可得到 precision。

(二) recall

在計算 recall 的部分，我們先將問題設定為與「Docker 容器化部署」相關的，並將評估分為四個維度，每個維度都有不同的權重和評估問題：

1. 基礎概念（權重：0.3）
   (1) 評估文檔是否包含容器相關概念
   (2) 評估是否包含 Docker 相關概念
2. 核心組件（權重：0.3）
   (1) 評估文檔是否包含 Docker 基本組件
   (2) 評估文檔是否包含 Docker 配置文件
3. 部署相關（權重：0.2）
   (1) 評估文檔是否包含環境相關內容
   (2) 評估文檔是否包含安全相關內容
   (3) 評估文檔是否包含運維相關內容
4. 工具相關（權重：0.2）
   (1) 評估文檔是否包含容器編排內容
   (2) 評估文檔是否包含開發工具內容
   (3) 評估文檔是否包含其他相關內容

最後根據此公式 $\text{recall} = \sum_{i=1}^{4} \text{維度分數}_i \times \text{權重}_i$ 即可計算出 recall 分數。

根據我們的測試結果，本系統在處理總字數高達 30 萬字的資料時，precision 可高達 91%，而 recall 則為 82%。

根據表 3（precision 比較表）及表 4（recall 比較表）可以發現，本系統（NRAG）相較 GraphRAG 可以保持接近 95%的 precision，而與 VectorRAG 相比，則可以有 8.3%的提升。而 recall 的部分是因為本研究在處理文件時，並無對其進行更細部的拆分，因此在評分上相較之下會較為弱勢，但仍然能保持 82%的 recall。

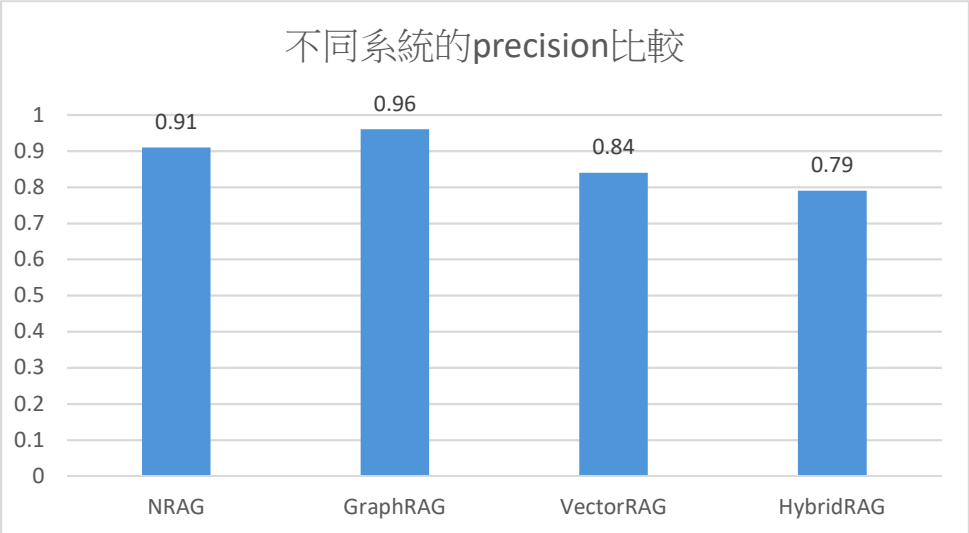

表 3、不同系統的 precision 比較（來源：自行製作）

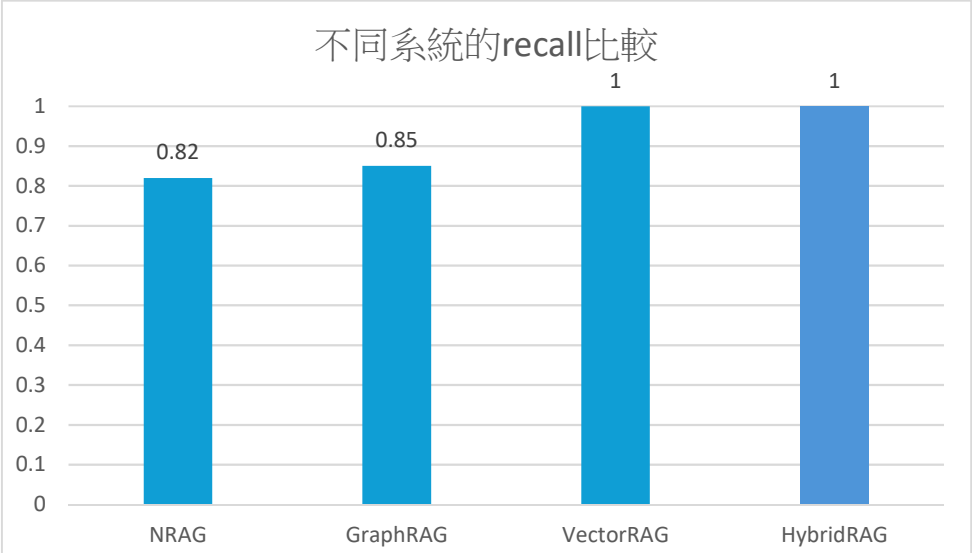

表 4、不同系統的 recall 比較（來源：自行製作）

三、預處理評估

經過我們的測試，本系統平均每筆資料僅需 8.43 秒便可處理完畢，而平均處理每一千字需要 8.64 秒。因此若與 Sindy_he（2025）[15]對 GraphRAG 的測試結果進行比較，本研究將可以省下 76.5%的預處理時間。

此外，我們亦在不同硬體與部署環境下進行測試。若於本研究設備上使用 Ollama 部署本地的 Gemma3 4b 模型，雖可執行，但是預處理所需時間明顯偏長，且由於筆記型電腦並不適合進行如此長時間的高負載運算，因此我們最終決定放棄此測試。而實際測試中，耗時約兩小時僅完成第一階段 6%的進度，並且資源使用率也時常處於閒置的狀態。

而若是改以 LM Studio 部署 Gemma3 4b QAT，並關閉同步處理功能後，雖可將處理十萬字資料的時間壓縮至 20 分鐘左右，但最終階段卻會因格式錯誤而中斷處理。

而在另一台配備 NVIDIA RTX 3080 10GB 的桌上型電腦上則成功在使用 Ollama 部署 Gemma3 4b 的情況下，經過測試，預處理時間約為 13 小時。這也顯示其在資源調度上存在明顯限制。

四、檢索時間評估

在檢索時間上，本系統在處理約 10 萬字的資料時，僅需要 12 秒，而在 GraphRAG 上，我們透過 ollama 在本地部署 Gemma3 4b 進行測試時，他的檢索時間則需要約 4 分 20 秒的執行時間，相比本系統有著接近 21.67 倍的的時間增長。

根據以上實驗結果可知，透過結合 LSH 及知識圖譜，與改良預處理流程，我們將可以在保持高精確度的同時，顯著降低執行與預處理的時間。此結果不僅佐證本系統在效能優化上的可行性，也為後續低延遲、高效率的 RAG 系統開發提供參考。圖 7 為本執行示意圖。

圖 7、執行示意圖（來源：自行製作）

伍、 結論

為解決 RAG 難以兼顧查詢效率與精確度的問題，本研究提出透過結合 LSH 與知識圖譜的系統。透過使用 LSH 快速篩選相關度高的資料，再使用知識圖譜進行更精細的篩選，以達到加速檢索與提高精確度的目的。

根據實驗結果顯示，本系統不僅在精確度上能保持 GraphRAG 的 94.79%，並且能夠減少高達 76.5%的預處理時間與 95.38%的檢索時間，使效率與準確度兼具。結果證明透過結合 LSH 及知識圖譜，與改良預處理流程可以提供更高效率的解決方案。

未來，本系統將可應用於更多場景中，包括但不限於企業內部的知識管理、醫療資訊、法律文件分析與科技研發等高專業性的領域，協助使用者能夠快速從龐大的資料中獲取關鍵的資訊。同時，也能根據企業需求調整提示詞與模型配置，以因應不同領域的需求。

此外，若是將文檔分割的技術引入本系統，將文章進行切割，將可使生成的時候可以更精確的載入資料，並節省生成時所需的 token 量。

而若是引入多模態資料處理（如圖像、語音與結構化資料），本系統將有望成為更全面的 AI 輔助系統，進一步拓展其對複雜任務的理解與處理能力，提升跨領域應用的靈活性與精確性，並融入使用者日常生活中，以協助人們在學習與工作中能夠更加高效與便利

最後，我們期望能為 RAG 系統在應用上的可行性、彈性與安全性提供貢獻，並且打造一個快速又可信的智慧知識輔助系統，推動 AI 在實務場域中的應用與普及。

陸、 參考文獻資料

一、英文文獻

[1] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2024). *From local to global: A Graph RAG approach to query-focused summarization* [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2404.16130

[2] Jafari, O., Maurya, P., Nagarkar, P., Islam, K. M., & Crushev, C. (2021). *A survey on locality sensitive hashing algorithms and their applications* [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2102.08942

[3] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). *Retrieval-augmented generation for knowledge-intensive NLP tasks* [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2005.11401

[4] Sarmah, B., Hall, B., Rao, R., Patel, S., Pasquali, S., & Mehta, D. (2024). *HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction* [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2408.04948

[5] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). *BERT: Pre-training of deep bidirectional transformers for language understanding*. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171 – 4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

二、中文文獻

[6] 徐金廷（2024）。《利用檢索增強生成於線上問答系統之研究與實作》（碩士論文，明新科技大學資訊管理系）。臺灣博碩士論文知識加值系統。https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/ccd=eSLaM7/record?r1=1&h1=1

[7] 林祐宸（2024）。《基於微調語言模型及檢索增強生成建構繁體中文法律文件問答系統》（碩士論文，東吳大學資訊工程學系）。臺灣博碩士論文知識加值系統。https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/ccd=eSLaM7/record?r1=1&h1=2

[8] 黃翊嘉（2024）。《以知識圖譜強化 RAG 技術之 QA 大語言模型》（碩士論文，淡江大學資訊工程學系）。臺灣博碩士論文知識加值系統。https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/ccd=eSLaM7/record?r1=1&h1=3

三、網路資源

[9] Sorathiya, A. (2021, January 29). Introduction to Neo4j. *TechPanel*. https://medium.com/techpanel/introduction-to-neo4j-84bd1ccfb37e

[10] OOSGA. (2024)。*知識圖譜（Knowledge Graph）的定義為何？* OOSGA 知識庫. https://zh.oosga.com/docs/knowledge-graph/

[11] Cloudflare. (2024)。什麼是 AI 幻覺？Cloudflare 學習中心. https://www.cloudflare.com/zh-tw/learning/ai/what-are-ai-hallucinations/

[12] 陳妍姍 (2024)。模擬語言模型的習性：LLM 會偏袒什麼樣的文章？探討 RAG 架構的潛在攻擊危機。臺灣大學資訊工程學系。SITCON 2024。https://sitcon.org/2024/poster/摸透語言模型的習性_LLM_會偏袒什麼樣的文章.pdf

[13] 楊鈞宜（2024）。RAG 技術終極入門：基礎架構與工作原理詳解。idataagent。https://idataagent.com/2024/05/12/the-ultimate-introduction-to-rag-technology-detailed-explanation-of-infrastructure-and-working-principles/

[14] Ralph Tech. (n.d.). 資料結構學習筆記：雜湊表 (Hash Table). Medium. Retrieved December 7, 2024, from https://medium.com/@ralph-tech/資料結構學習筆記-雜湊表-hash-table-15f490f8ede6

[15] Sindy_he. (2025, May 14). 2025 最新版微软 GraphRAG 2.0.0 本地部署教程：基于 Ollama 快速构建知识图谱. CSDN 博客. https://blog.csdn.net/m0_54356251/article/details/146074188

[16] fan84sunny（2024, Sep）【AI 筆記】30 天從論文入門到 Pytorch 實戰：GraphRAG 論文閱讀 Day 27. iT 邦幫忙. https://ithelp.ithome.com.tw/m/articles/10345984

[17] huangyihe（2024, Jul, 13）GraphRAG：很好，但很贵！Youtube. https://youtu.be/n8CIlK_mO2g?si=-89MZPDIHbvvZJU6

[18] 黃適文. (2024, Oct, 08)。什麼是 RAG（Retrieval-Augmented Generation）？Solwen AI. https://solwen.ai/posts/what-is-rag

[19] zhihaoshi1729（2023, Jun）如何理解 Locality Sensitive Hashing 中的 LSH Family？Medium. https://medium.com/@zhihaoshi1729/如何理解 locality-sensitive-hashing 之中的 lsh-family-2c31f4aba84c

[20] Chiu, B.（2024, Sep 22）。何時不應該用知識圖譜 GraphRAG？Medium。https://medium.com/@bohachu/不要使用知識圖譜-graphrag-的時機-7354d101b53b

四、模型資源

[21] 北京智源人工智慧研究院（BAAI）. (n.d.). *BAAI/bge-large-zh*. Hugging Face. https://huggingface.co/BAAI/bge-large-zh

[22] 北京智源人工智慧研究院（BAAI）. (2023, September 12). *BAAI/bge-reranker-large*. Hugging Face. https://huggingface.co/BAAI/bge-reranker-large

[23] Google. (2025). *Gemma 3 4B Instruction-Tuned QAT Q4_0 GGUF*. Hugging Face. https://huggingface.co/google/gemma-3-4b-it-qat-q4_0-gguf

[24] Google. (2025). *Gemma 3 12B Instruction-Tuned QAT Q4_0 GGUF*. Hugging Face. https://huggingface.co/google/gemma-3-12b-it-qat-q4_0-gguf

[25] DeepSeek AI. (2025). *DeepSeek-R1-0528-Qwen3-8B*. Hugging Face. https://huggingface.co/deepseek-ai/DeepSeek-R1-0528-Qwen3-8B

【評語】052513

本作品結合局部敏感雜湊（Locality-Sensitive Hashing, LSH）與知識圖譜（Knowledge Graph, KG）以改善檢索增強生成技術（Retrieval-Augmented Generation, RAG）在檢索的時候難以在精確度與效率之間取得平衡的問題。此為學術上重要的研究議題，研究題目具學術性與發展性。

文獻上雖有類似的相關研究，雖非採用 LSH+KG，建議仍可進行比較分析，以凸顯本研究的價值。

作品海報

結合LSH及知識圖譜改善RAG

研究動機

為了提高員工的工作效率，許多企業已經開始將 AI 技術導入內部的流程中。然而，若是單純使用ChatGPT或Gemini等雲端模型，便可能面臨公司機密資料外洩的風險。此外，除非企業自行訓練或微調AI模型，否則這些模型並無法獲得公司的內部資料。

為了解決此問題，部分企業會使用RAG，讓AI在回應之前，能夠先查找與問題相關的資料或技術文件，藉此提升回應的準確性。然而，此方式在實作上仍存在許多挑戰，例如基於向量檢索的RAG雖然速度快，卻在面對語意模糊或需要推理的問題時，表現並不穩定。

為強化AI的語意理解與邏輯推理能力，近年已有研究嘗試將知識圖譜結合進RAG中，透過實體與關係的結構化資訊進行輔助。然而，這樣的設計也使系統架構更加複雜，導致查詢效率下降，因此也難以應對需要即時回應的應用場景。

研究目的

1. 透過知識圖譜提升系統在語意理解與關鍵詞關聯能力
2. 構建LSH系統，以快速分類相似資料
3. 比較不同RAG系統，驗證本系統在準確率與效率上的優勢

	知識圖譜構建	LSH	不同系統的比較
運作原理	以三元組方式建立實體間的語意關係	使相似向量映射至同一雜湊桶	與相似系統進行評估
系統設計	使用LLM擷取關鍵字，並構建圖譜	使用隨機投影矩陣進行降維映射	使用LLM評估不同系統的Precision與Recall
使用目的	提高檢索準確性	提升檢索效率	驗證本方法在效能與成本間的優勢

表1、研究目的統整表（來源：自行製作）

文獻探討

技術	生成式AI	VectorRAG		GraphRAG	
困境	在生成回應時，模型可能會出現幻覺	語意推理能力不足	易納入無關資料	處理效率緩慢	系統資源需求過高
解決方式	使用RAG，讓AI在回應前先查詢相關資料	將知識圖譜引入RAG系統內使其在檢索資料時能夠有更高精確性		使用LSH快速篩選資料以提升檢索效率	流程模組化以降低系統資源耗用

RAG 架構概述與核心原理

ChatGPT或Gemini等生成式AI，雖然能生成流暢的語句，但其知識來源僅來自訓練時的資料，若無加上聯網或是檢索機制的話，容易使AI在回應的時候產生幻覺(hallucination)。為了解決此限制，Patrick Lewis等人（2020）提出RAG（Retrieval-Augmented Generation）架構，透過結合預訓練生成模型與非參數記憶（如檢索模組），使模型能在生成過程中動態查詢外部知識，提升其在需要專業知識的任務中的準確度，RAG架構圖如圖1所示。

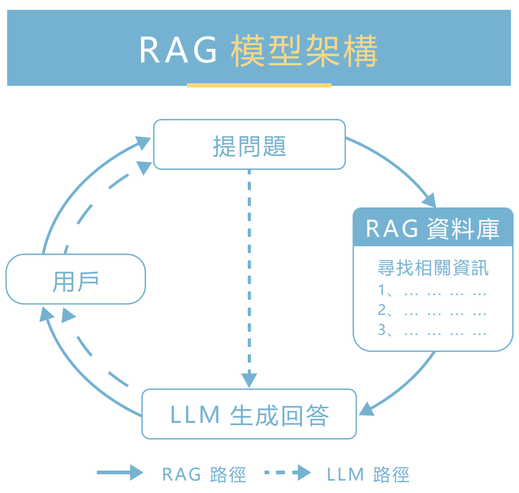

圖1、RAG模型架構，取自黃適文（2024）之文章《什麼是 RAG？初學者也看得懂的檢索 增強生成（RAG）基礎指南！》

基於語意向量的RAG的應用與挑戰

嵌入向量檢索（Embedding-based Retrieval）為RAG架構中最基礎，且被廣泛使用的技術之一。透過BERT等語言模型，文本內容可被轉換為嵌入向量（Embedding vectors），再經由相似度計算進行檢索，從而快速找出與查詢相關的內容。該方法具有操作簡單、查詢快速等優點，因此被廣泛應用於一般性問答系統中。然而，相關研究指出，嵌入向量檢索在處理複雜或專業領域問題時，存在語意理解不足與推理能力缺乏的問題，使其檢索階段可能引入與查詢問題無關的資料，形成所謂的雜訊。這些資料若未經過妥善的篩選，可能導致生成內容產生語意漂移（Semanticdrift），進而影響輸出的品質。徐金廷（2024）與林祐宸（2024）均指出，單純依賴向量相似度可能導致查詢結果與使用者需求不符，尤其是在處理具有明確邏輯結構或需跨領域語意連結的查詢任務時，其表現相對受限。因此，雖然語意向量技術具備一定的基礎能力，但仍需要以其他的技術手段來提升其在深層語意理解與跨主題內容生成上的能力。

局部敏感雜湊於高效檢索中的應用

局部敏感雜湊（Locality-Sensitive Hashing, LSH）是高維空間中進行近似最近鄰（Approximate Nearest Neighbor, ANN）搜索的技術。其原理是透過雜湊函數，將高維資料映射到低維空間，使相似的資料可以被映射到同一個雜湊桶中，藉此快速的將相似的資料分類到一起。 根據Jafari等人（2021）的研究可知，LSH在處理大規模資料時，不僅具有優秀的檢索效率，也廣泛應用於高維度數據，如語意向量、圖像特徵與醫學資料等場景。LSH架構圖如圖2。

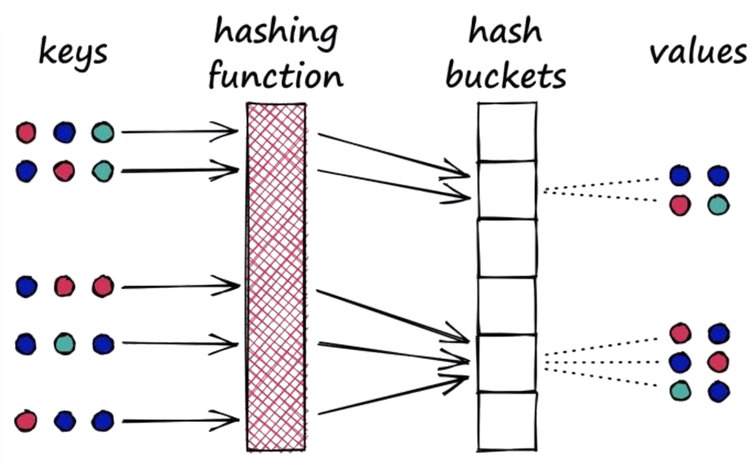

圖2、LSH架構圖，取自科波拉（2022）之文章《局部敏感哈希算法及其SparkScala實現》

評估知識圖譜導入RAG的影響

知識圖譜（Knowledge Graph）是透過圖狀結構表達實體與實體之間的語意關聯。透過將實體與關係組成「三元組（Triple）」，知識圖譜能夠清楚呈現實體之間的語意關係，進而被用於語意理解、資訊整合與智慧推理等方面。圖3為知識圖譜示意圖。

根據Edge等人（2024）的研究可以發現，結合知識圖譜的GraphRAG相較使用向量的RAG，能更有效的捕捉文本之間的關聯性，進而使LLM能提供更全面且準確的回答。然而，此系統對硬體或雲端資源的需求也相對較高。根據Sindy_he、Bowen Chiu與huangyihe等人的測試可以發現，雖然此方式能提升精確度，但在預處理、檢索與生成階段，計算與金錢成本也相對更高。

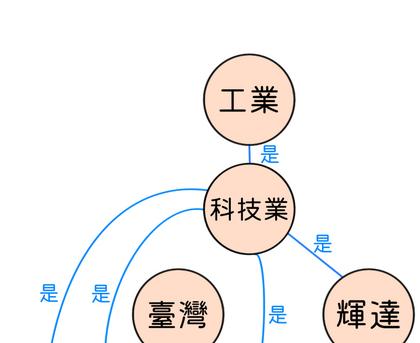

圖3、知識圖譜示意圖（來源：自行製作）

研究過程與方法

預處理階段

1. 關鍵字判斷與摘要生成
使用LLM提取出具有實際意義的關鍵字與進行內容摘要

2. 關鍵字語意關聯分析
將資料與獲取的關鍵字交給LLM進行分析，找出關鍵字在原文之間的關聯為何，並構建出對應的三元組

3. 關鍵字翻譯
將非專有名詞統一翻譯成英文，而專有名詞，如人名、地名等，則保持不變

4. 資料向量化
將資料與獲取的關鍵字進行向量化處理，並儲存至向量資料庫

> ⓘ 模型介紹：向量化模型
>
> 本研究使用由北京智源研究院於2023年推出的文本向量化模型 BAAI/bge-large。該模型支援多語言語意理解與檢索任務，能有效提取文本語意特徵，提升關鍵字擷取與語意比對的準確性。

5. 其他內容
將檔案路徑、檔名與判斷後的關鍵字等資訊添加到資料庫中

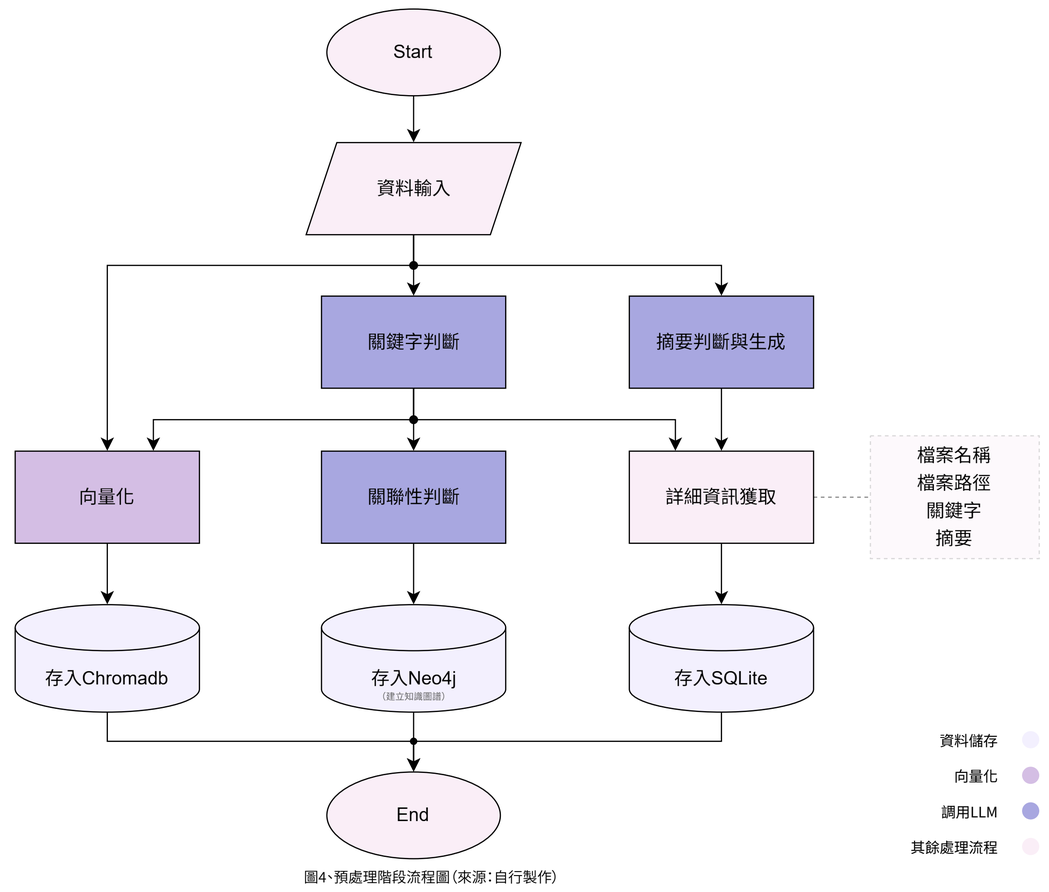

圖4、預處理階段流程圖（來源：自行製作）

生成階段

1. 關鍵字獲取
- 透過LLM判斷使用者所詢問的問題需要什麼關鍵字
- 根據判斷出來的關鍵字使用向量搜索找到最接近的三個關鍵字
- 將找到的關鍵字使用知識圖譜向下三層找到與其相關的五個關鍵字

2. 雜湊表建立與排序
- 根據文檔數量和向量維度自動計算適當的雜湊表數量
- 使用隨機投影矩陣進行降維映射

3. 資料篩選
- 初始篩選條件設定：設定初始篩選條件（70%），當文檔中的關鍵字與先前獲取的關鍵字有超過2個交集，即納入候選文檔
- 候選文檔累積與條件調整：若當前桶中符合條件的文檔比例已達到篩選條件，且已經收集超過50個候選文檔，系統會停止搜索，反之，系統則會逐步降低篩選條件，繼續搜索更多資料

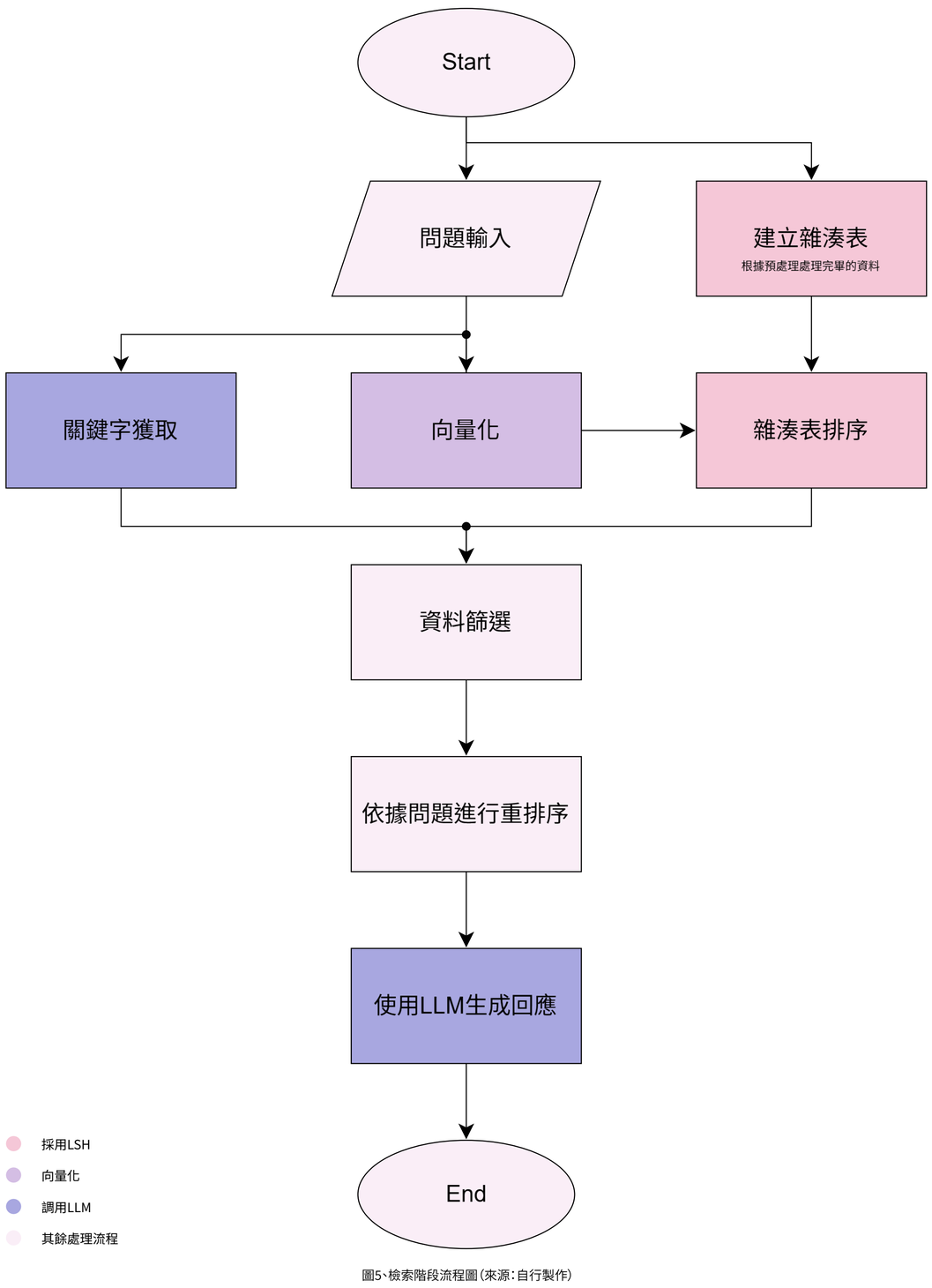

圖5、檢索階段流程圖（來源：自行製作）

4. 重排序
透過重排序模型，將篩選出的資料依照與查詢的相關性重新排列，讓高相關內容優先呈現

> ⓘ 模型介紹：重排序模型
>
> 本研究使用由北京智源研究院於 2023 年推出的語意重排序模型BAAI/bge-reranker-large。該模型專為語意檢索中的結果排序優化設計，能根據查詢與文本間的語意關聯性進行精細排序，有效提升語意比對與資訊檢索的準確性。

5. 生成回應
將檢索階段篩選出的相關資料作為上下文，提供給LLM進行回應

> ⓘ 補充：生成階段提示詞設計
>
> 為避免LLM傾向直接依賴檢索到的參考資料進行作答，而忽略對資料正確性與一致性的判斷，導致錯誤資訊被納入回應內容，本研究特別設計了生成階段的prompt，引導模型在回答前先進行初步判斷與過濾，以降低誤導性內容的產生，提升回應的可信度與準確性。

> ⓘ 生成階段提示詞
>
> 你是一位專業的 AI 助手。請根據提供的上下文信息回答用戶的問題，並自動補充相關的專業知識和背景資訊。回答應該詳細、完整，並使用 Markdown 格式來美化排版。
>
> 回答要求：
>
> 1. 內容要求：
>   * 優先使用提供的上下文信息作為回答的主要依據
>   * 回答必須詳細，控制在 500-800 字之間
>   * 自動補充相關的專業知識和背景資訊
>   * 保持專業且客觀的立場
>   * 內容應該深入且全面
> 2. 知識補充要求：
>   * 自動補充相關的專業術語解釋
>   * 補充相關的技術背景和發展歷程
>   * 提供實際應用案例或最佳實踐
>   * 補充相關的技術趨勢或最新發展
>   * 確保補充內容的準確性和專業性
> 3. 格式要求：
>   * 使用 Markdown 格式進行排版
>   * 使用標題、小標題等基本格式
>   * 使用列表或項目符號來呈現關鍵點
>   * 確保排版簡潔清晰
>   * 適當使用粗體或斜體強調重要內容
> 4. 結構要求：
>   * 開頭直接回答核心問題
>   * 使用多個小標題分隔主要部分
>   * 每個部分都應該詳細闡述
>   * 使用簡短的列表呈現要點
>   * 結尾應該有總結性的內容
> 5. 補充原則：
>   * 補充的內容必須與問題高度相關
>   * 確保補充的知識準確且有用
>   * 補充內容應該豐富且深入
>   * 保持內容的邏輯性和連貫性
>   * 適當引用權威來源或最佳實踐

研究結果

關鍵字與關聯性評估

此部分我們會隨機抽取10筆資料進行測試，並分別對關鍵字的完整性、準確性，以及關鍵字分類、關係分析與翻譯等方面的準確性進行評估。

根據實驗數據可以知道，在需要本地部署，且硬體資源充足的情況下，Deepseek r1 0528 Qwen3 8b是相對最佳的選擇。而若顯存資源有限，則可考慮Gemma3 4b QAT，這樣將能在效能與資源之間取得較好的平衡。而在雲端資源方面，Gemini 2.5 flash（未開啟 thinking模式）可以提供穩定且高品質的輸出表現。

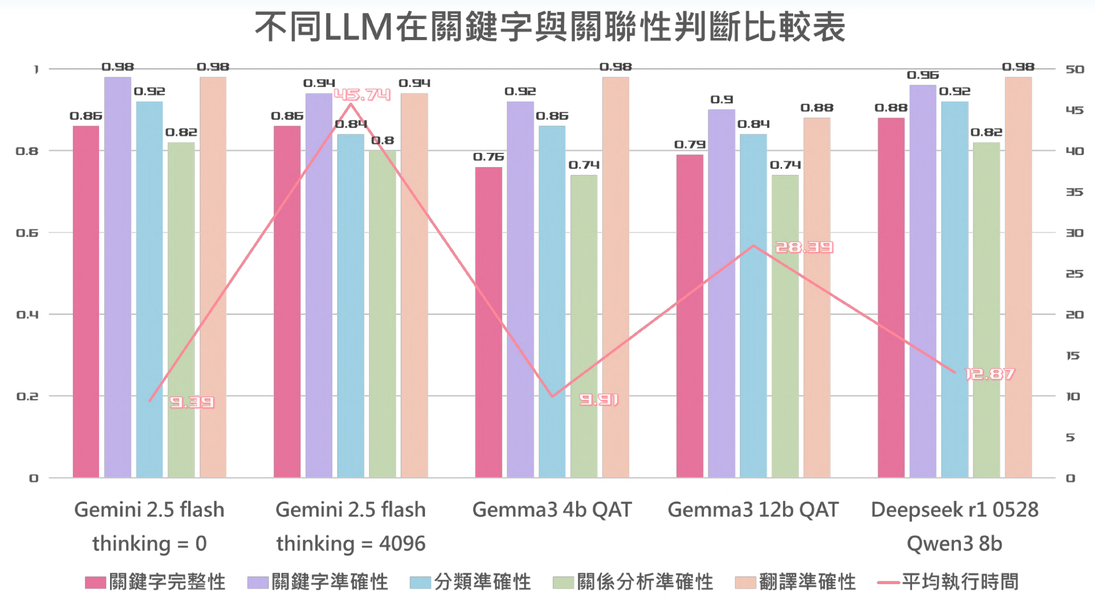

表2、不同LLM在關鍵字與關聯性判斷比較表（來源：自行製作）

Precision與Recall評估

在計算Precision的部分，我們將資料分為完全相關、部分相關與完全不相關三種，再來計算完全相關與部分相關占全部檢索資料的比例。

而在Recall的部分，我們採用的則是先設定一個問題，並將其再細分為不同的維度，且每個維度都有不同的權重和評估的問題。最後，根據各維度加權得分計算出最終的Recall分數。最後，根據檢索到的資料進行評估，以取得最終的Recall分數。

根據測試結果，本系統（NRAG）相比GraphRAG可以保持94.79%的Precision，而與VectorRAG相比，則可以有8.3%的提升。Recall的部分是因為本研究在處理文件時，並無對其進行更細部的拆分，因此在評分上相較之下會較為弱勢，但仍然能保持82%的Recall。

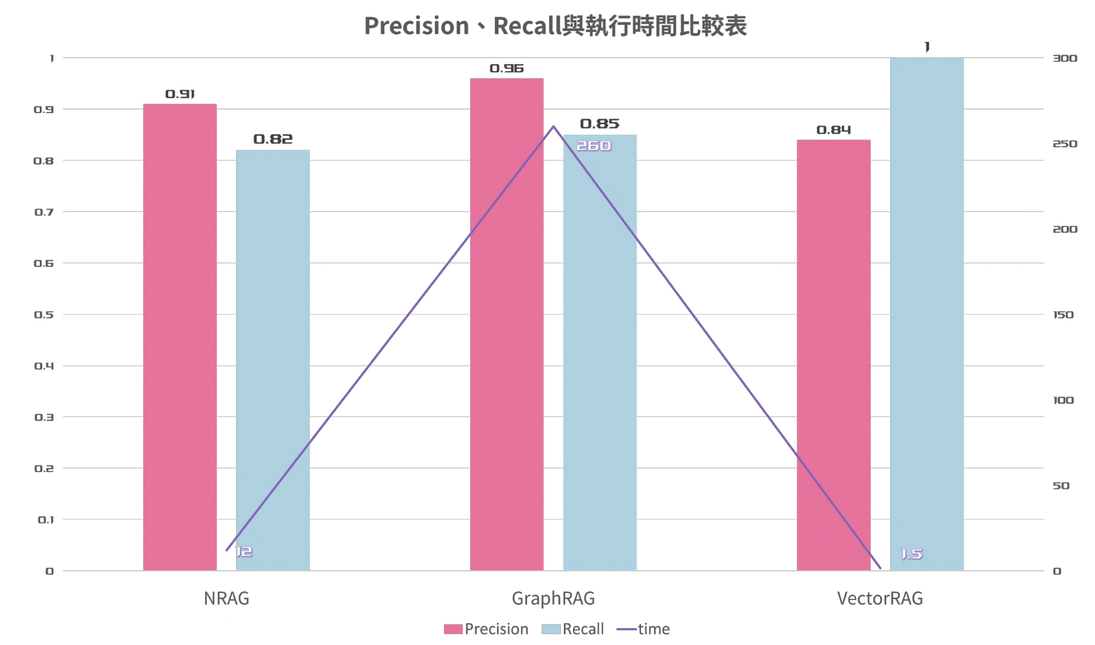

表3、Precision與Recall比較表（來源：自行製作）

本研究中GraphRAG與VectorRAG的Precision與Recall數據引用自文獻[4]。因設計差異與整合成本考量，並且部分系統原代碼並無法獲取，因此本研究未進行統一測試，僅作為檢索效能之趨勢性比較參考。

預處理與檢索時間評估

經過我們的測試，本系統平均處理每一千字僅需要8.64秒。因此若與Sindy_he（2025）對GraphRAG的測試結果進行比較，本研究將可以省下76.5%的預處理時間。

此外，我們在另一台配備 NVIDIA RTX 3080的電腦上，使用Ollama部署Gemma 3 4b進行約10萬字資料的測試，所得到的預處理時間約為13小時。

而在檢索時間上，本系統在處理約10萬字的資料時，僅需要12秒，而在GraphRAG上，我們透過Ollama在本地部署Gemma3 4b進行測試時，他的檢索時間則需要約4分20秒，相比本系統增加21.67倍的處理時間。

最後為了分析是否是因為硬體的差異，導致我們測試的檢索時間異常緩慢，因此我們分別在RTX 3080與RTX 5070 Ti laptop上進行測試。而在觀察工作管理員的資源使用情況後我們發現，GraphRAG在本地部署的情況下，即使硬體資源充足，硬體調用的使用率仍然偏低，這也顯示其在資源調用上存在明顯缺陷。

此外，根據 huangyihe（2024）使用ChatGPT API進行的測試，處理約三萬字文本的預處理時間僅數分鐘，而檢索時間僅需15秒左右。此結果亦佐證了 GraphRAG 在本地調用時存在資源調用效率不足的問題。

研究結論

為解決RAG難以兼顧查詢效率與精確度的問題，本研究提出透過結合LSH及知識圖譜以改善RAG。透過使用LSH快速篩選相關度高的資料，再使用知識圖譜進行更精細的篩選，使其能夠在提高檢索效率的同時保持高精確度。

根據實驗結果顯示，本系統相較於VectorRAG，精確度有8.33%的提升。而與GraphRAG相比，則能維持94.79%的精確度，同時減少76.5%的預處理時間與96.38%的檢索時間。此結果證明，此方式能夠在提供更好的檢索效率的同時，保持高度的精確度。

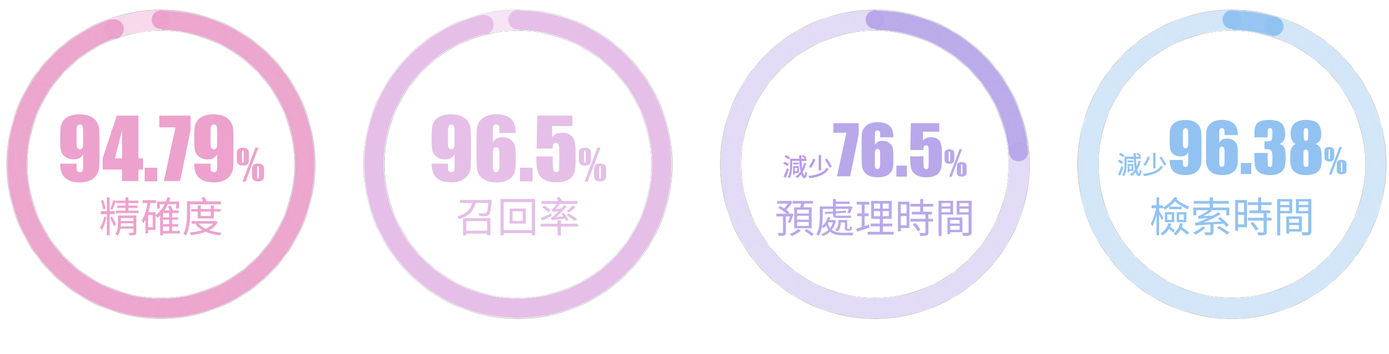

圖6、本研究與GraphRAG的精確度與處理效率比較圖（來源：自行製作）

未來展望

未來，本系統能用於更多場景中，包括企業內部的知識管理、醫療資訊、法律文件分析與科技研發等高專業性的領域，協助使用者能夠快速從龐大的資料中獲取關鍵的資訊。也能根據企業需求調整提示詞與模型配置，因應不同領域的需求。此外若是將文檔分割的技術引入本系統，將文章進行切割，將可使生成的時候可以更精確的載入資料，從而節省生成時所需的token量。最後我們期望能為RAG系統在應用上的可行性、彈性與安全性提供貢獻，並且打造一個快速又可信的智慧知識輔助系統，推動AI在實務場域中的應用與普及。

參考文獻

[1] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2024). From local to global: A Graph RAG approach to query-focused summarization [Preprint]. arXiv.
[2] Jafari, O., Maurya, P., Nagarkar, P., Islam, K. M., & Crushev, C. (2021). A survey on locality sensitive hashing algorithms and their applications [Preprint]. arXiv.
[3] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). Retrieval-augmented generation for knowledge-intensive NLP tasks [Preprint]. arXiv.
[4] Sarmah, B., Hall, B., Rao, R., Patel, S., Pasquali, S., & Mehta, D. (2024). HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction [Preprint]. arXiv.
[6] 徐金廷（2024）。《利用檢索增強生成於線上問答系統之研究與實作》（碩士論文，明新科技大學資訊管理系）。臺灣博碩士論文知識加值系統。
[7] 林祐宸（2024）。《基於微調語言模型及檢索增強生成建構繁體中文法律文件問答系統》（碩士論文，東吳大學資訊工程學系）。臺灣博碩士論文知識加值系統。
[8] 黃翊嘉（2024）。《以知識圖譜強化RAG技術之QA大語言模型》（碩士論文，淡江大學資訊工程學系）。臺灣博碩士論文知識加值系統。
[9] Ralph Tech. (2024, Dec 7). 資料結構學習筆記：雜湊表 (Hash Table). Medium.
[10] Sindy_he. (2025, May 14). 2025最新版微软GraphRAG 2.0.0本地部署教程：基于Ollama快速构建知识图谱. CSDN博客.
[11] huangyihe（2024, Jul, 13）GraphRAG：很好，但很贵！Youtube.
[12] Solwen AI. (2024)。什麼是 RAG（Retrieval-Augmented Generation）？Solwen AI 博客.
[14] Chiu, B.（2024 年 9 月 22 日）。何時不應該用知識圖譜 GraphRAG？Medium。
[15] 科波拉. (2022). 局部敏感哈希算法及其Spark與Scala實現. 百度貼吧.