給 RAG 安裝上 Agent 頭腦

在現代的訊息處理系統中，如何從大量數據中快速且準確地提取出有用訊息是一個關鍵的挑戰！！Retrieval Augmented Generation（RAG）框架已經證明了其在多個領域的有效性，尤其是在問答系統中。然而，隨著應用需求的不斷深化和擴展，RAG 的一些固有限制也逐漸浮現。之前小編有跟大家介紹何謂 RAG，今天將深入帶大家探討 RAG 的一些限制並介紹如何通過引入「Agent 大腦」，即「 Genentic RAG」，來克服這些挑戰，進一步增強系統的靈活性和效率！

RAG 框架的核心限制

RAG 結合了檢索和生成兩大技術，目的是利用已有的文檔數據庫來增強語言模型的回答能力。儘管這一方法提升了回答的質量，但在實際應用中，尤其是面對複雜或多維度的查詢時，存在以下幾個核心限制：

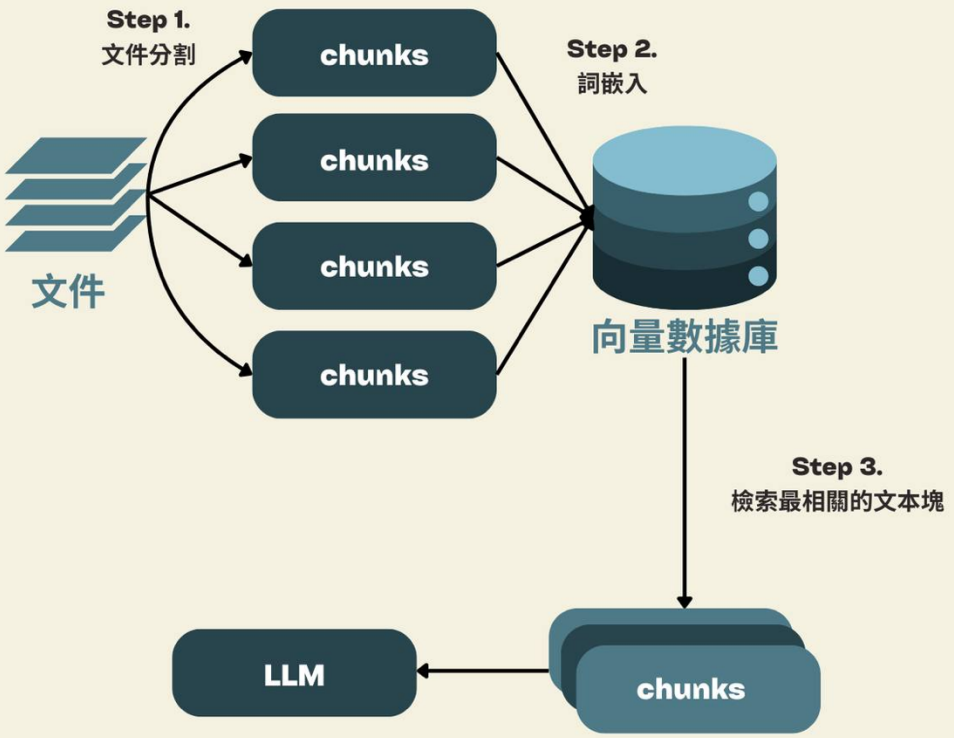
1. 訊息的片段化檢索：RAG 通常只從文檔中提取最相關的片段，這可能導致訊息的不完整，影響結果的全面性。

2. 查詢的覆蓋範圍限制：在處理涉及多個文檔或需要廣泛文獻回顧的查詢時，RAG 的檢索範圍受限於頂部 K 個文檔，可能忽略其他重要訊息。

3. 結構化數據查詢的不足：RAG 在處理結構化數據查詢時的表現不如專用的數據庫查詢工具，因為其檢索機制未必能準確映射到結構化數據的特定需求。

RAG Framework

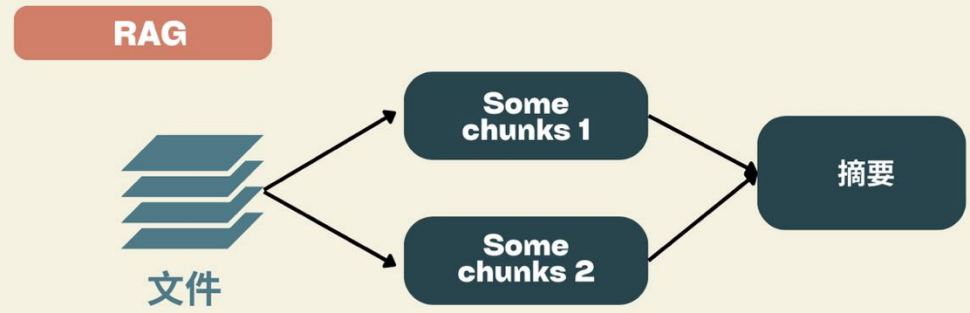
RAG是一種融合檢索技術與生成模型的AI架構，用於增強語言模型在生成文本時的相關性和準確性！

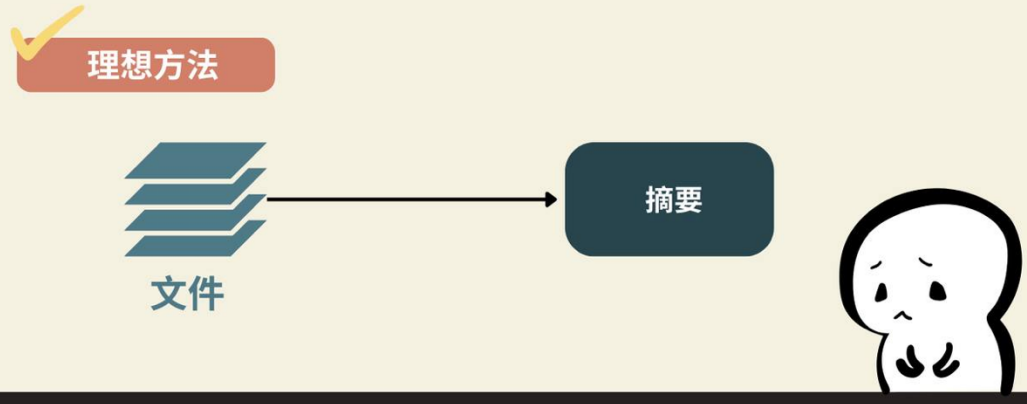


- 文件分割（Document ⟶Chunks）：首先，將一個完整的文件分割成多個小塊，稱為Chunks，使數據更容易被處理和分析
- 向量數據庫（Chunks ⟶ Vector DB）：接著，這些小塊被轉換為向量形式並存儲在向量數據庫中。此過程涉及對文本的特徵提取，使其能被機器學習模型理解和處理
- 塊檢索（Vector DB ⟶ Chunk Retrieval）：根據用戶的查詢，系統會從向量數據庫中檢索最相關的「top K」數據塊。此步驟是通過計算查詢向量和存儲的向量之間的相似度來完成的
- 語言模型處理（LLM）：檢索到的數據塊隨後被送入大型語言模型（LLM），並基於這些數據塊生成解答。

Challenge 1. Summarization

範例：總結文件 Summarize the document）





現有RAG方法

- 傳統的RAG方法從向量數據庫中檢索出最相關的「top K」數據塊，並將這些塊進行摘要

問題

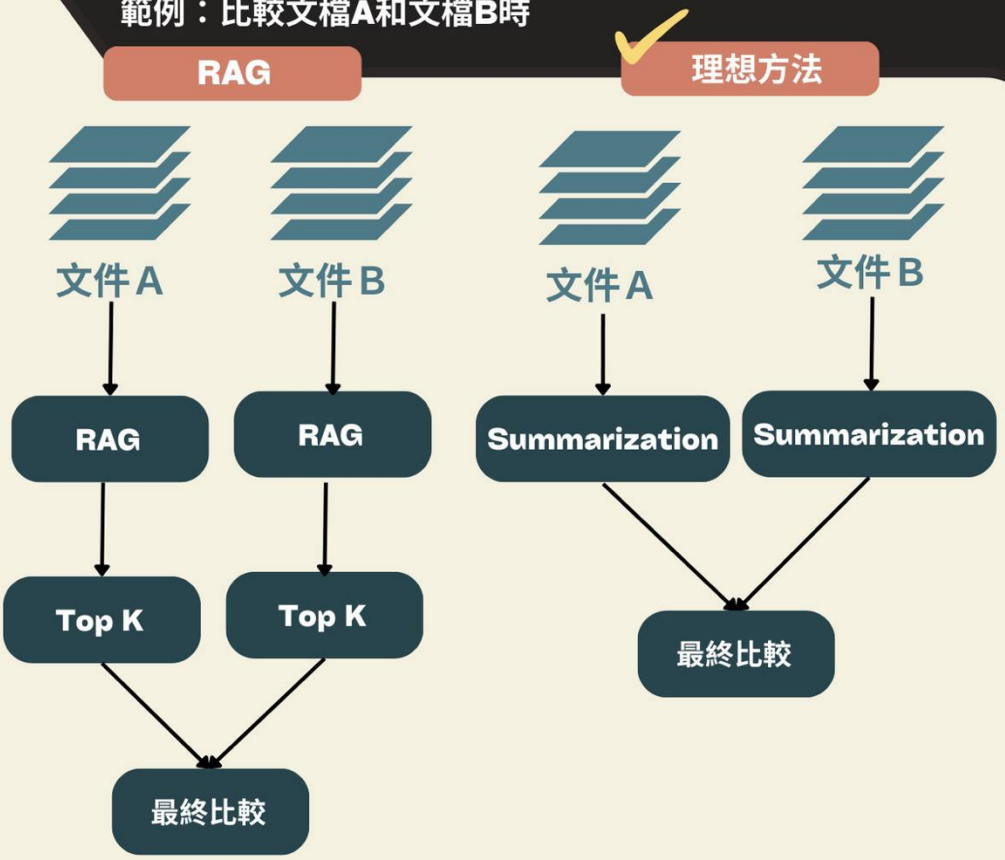
- 僅僅使用「top K」數據塊可能會導致摘要不夠全面，因為這些chunks只代表了文件的一部分，可能忽略了其他重要信息。

理想方法

- 檢索文件的所有數據塊，然後對整個文件進行摘要

Challenge 2. Comparing Docs

範例：比較文檔A和文檔B時



現有RAG方法

- 對文檔A進行RAG處理，檢索出「top K」數據塊。
- 對文檔B進行RAG處理，檢索出「top K」數據塊。
- 比較檢索到的這些數據塊。

理想方法

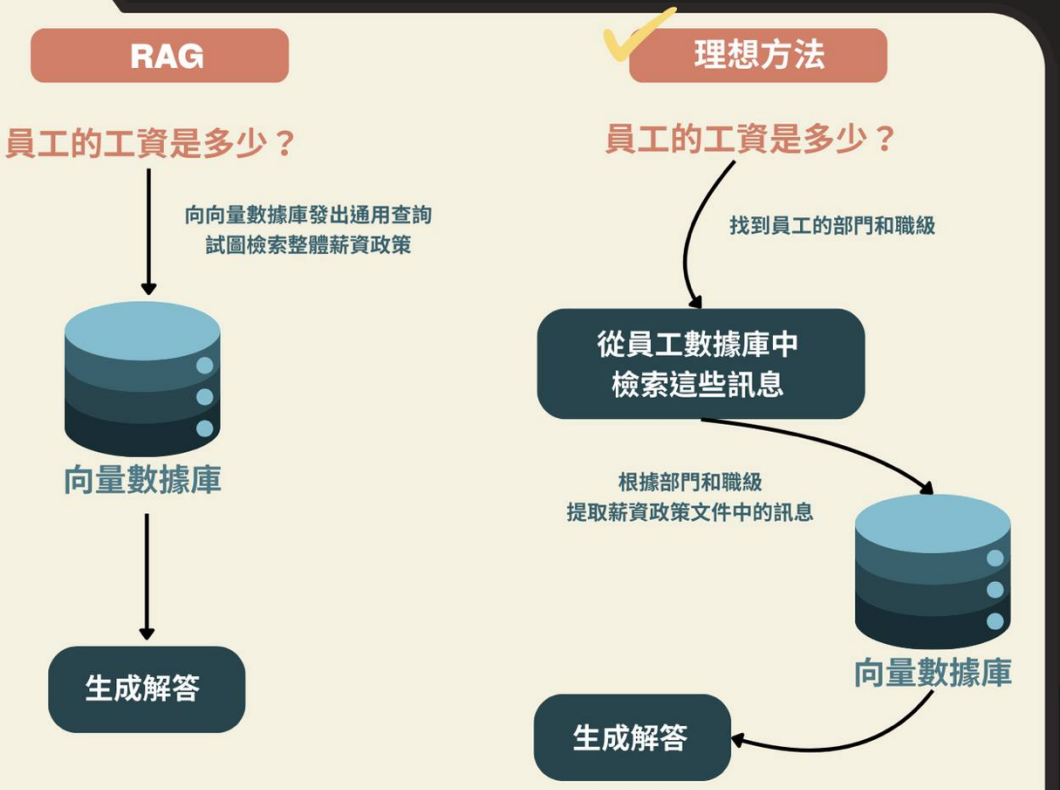
- 對文檔A進行摘要，生成一個綜合總結。
- 對文檔B進行摘要，生成一個綜合總結。
- 比較兩個摘要。

問題

- 無法全面比較兩個文檔，因為它只比較了各自的一部分，可能導致比較結果不準確。

Challenge 3. Structured Data Analysis

範例：員工的工資是多少？



現有RAG方法

- 系統向向量數據庫發出一個通用查詢，試圖從整體薪資政策文檔中找到答案。
- 基於這個查詢，系統檢索數據並生成最終答案。

問題

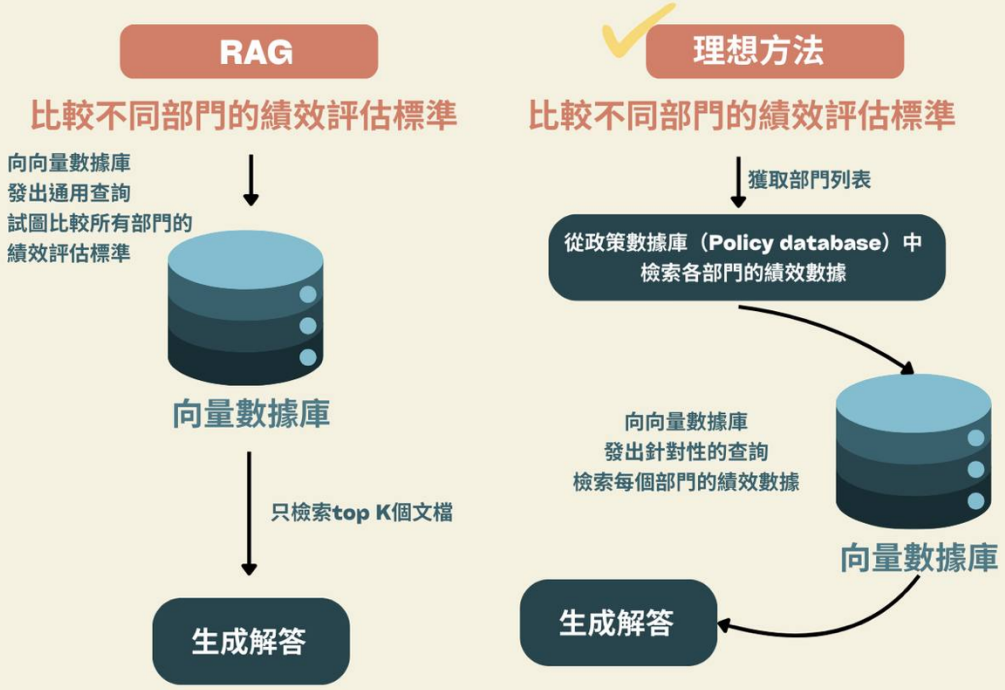
- 這種方法不夠直接，因為它沒有考慮具體的員工屬性（如部門、職級），導致結果可能不準確。

理想方法

- 找到員工的部門和職級：先從結構化數據表中檢索出員工的部門和職級訊息。
- 根據部門和職級提取薪資訊息生成最終答案

Challenge 4. The Multi-part Question

範例：比較不同部門的績效評估標準



現有RAG方法

- 系統向向量數據庫發出一個通用查詢，試圖比較所有部門的績效評估標準。
- 基於這個查詢，系統檢索數據並生成最終答案。

問題

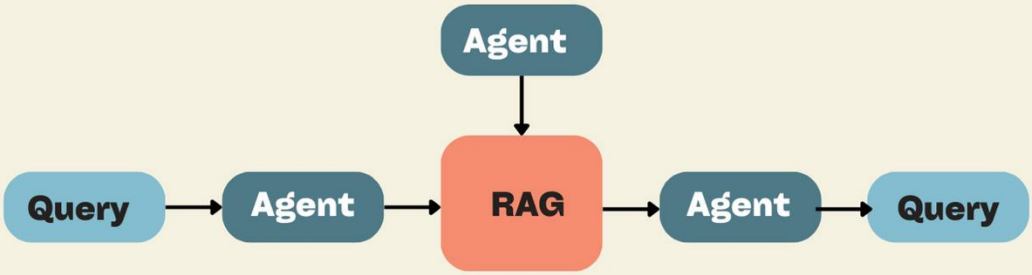
- 由於系統只能傳遞top K個上下文，因此可以比較的部門數量最多限制為K，這可能無法涵蓋所有部門。

理想方法

- 首先從政策數據庫中檢索出所有部門的列表。
- 根據部門向量數據庫中提取每個部門的績效評估標準數據。
- 生成最終答案：結合這些數據生成涵蓋所有部門的績效評估標準比較結果。

Agentic RAG

如果引入 Agents 就能解決RAG框架中遇到的四個問題拉！



核心概念

- Agentic RAG 框架通過引入代理、自動化推理和靈活的工具選擇，顯著提升了數據檢索與生成系統的效能。
- Agentic 不僅能夠處理複雜的查詢，還能夠根據需求選擇最合適的工具和方法，從而提供更準確和全面的答案。

運作流程

- 查詢（Query）：用戶發出查詢。
- 代理處理（Agent）：自定義代理接收查詢，並決定如何處理查詢。
- RAG檢索（RAG）：代理可以選擇使用RAG來進行檢索，獲取相關數據塊。
- 代理再處理（Agent）：代理根據檢索結果進行進一步的推理和數據處理。
- 最終回答（Final Answer）：代理生成最終回答並返回給用戶。

Genentic RAG 的創新解決方案

Genentic RAG 通過在傳統 RAG 框架中整合「Agent」概念，提出了一種更為動態和靈活的處理策略。這些 Agents 能夠在必要時調用不同的工具和技術，更加精確和全面地回答複雜查詢。具體改進如下：

1. 多 Agent 協同：不同的 Agents 專門處理不同類型的查詢或數據源，如路由 Agent 負責將查詢分配到最合適的處理工具，而查詢規劃 Agent 則專門解構複雜查詢為更易管理的子查詢。

2. 動態工具選擇與應用：Agents 根據查詢的具體需求動態選擇最適合的工具，例如在需要時使用向量數據庫或直接數據庫查詢，這樣可以更精確地定位和提取所需訊息。

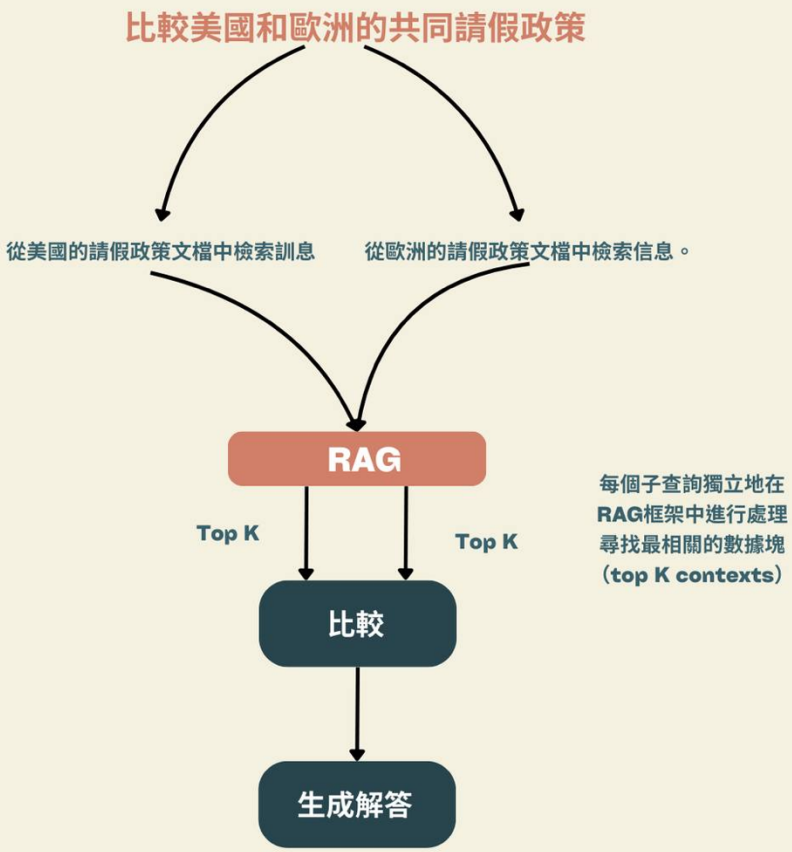
3. 增強的查詢處理能力：通過將查詢分解為可管理的部分並平行處理，Genentic RAG 能夠更有效地處理大規模或多維度查詢，從而提高整體的處理速度和準確性。

透過整合「Agent」概念，Genentic RAG 解決傳統的 RAG 在處理複雜和多維度查詢時所遇到的限制。這項創新不僅提高了問答系統的效率和準確性，還拓寬了其應用範圍。

Query Planning Agent

查詢規劃代理
專門處理復雜的查詢，將它們拆分成多個子查詢，每個子查詢都可以獨立執行，這樣提高了查詢的整體效率和準確性。

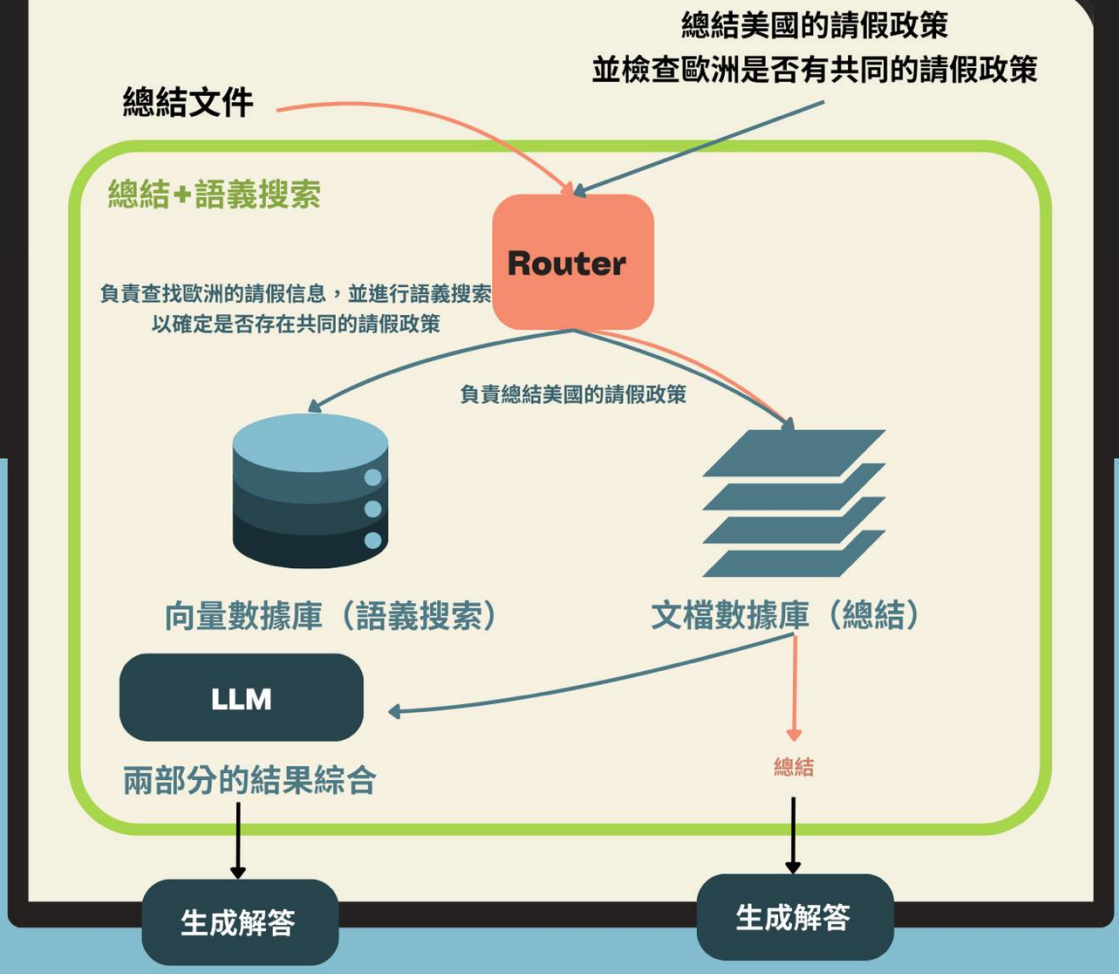
子查詢執行：每個子查詢都根據其需求被送到適合的處理流程中，這可能涉及RAG框架或其他數據處理工具。



- 查詢規劃代理透過智能分配查詢到合適的處理流程，有效處理多部分或複雜的查詢。
- 這種方法提高了查詢處理的效率和精確性，適用於需要從大量分散的資料源中擷取並比較訊息的情況。

Routing Agent

路由代理：
負責將查詢路由到一個或多個工具，以獲取所需的答案

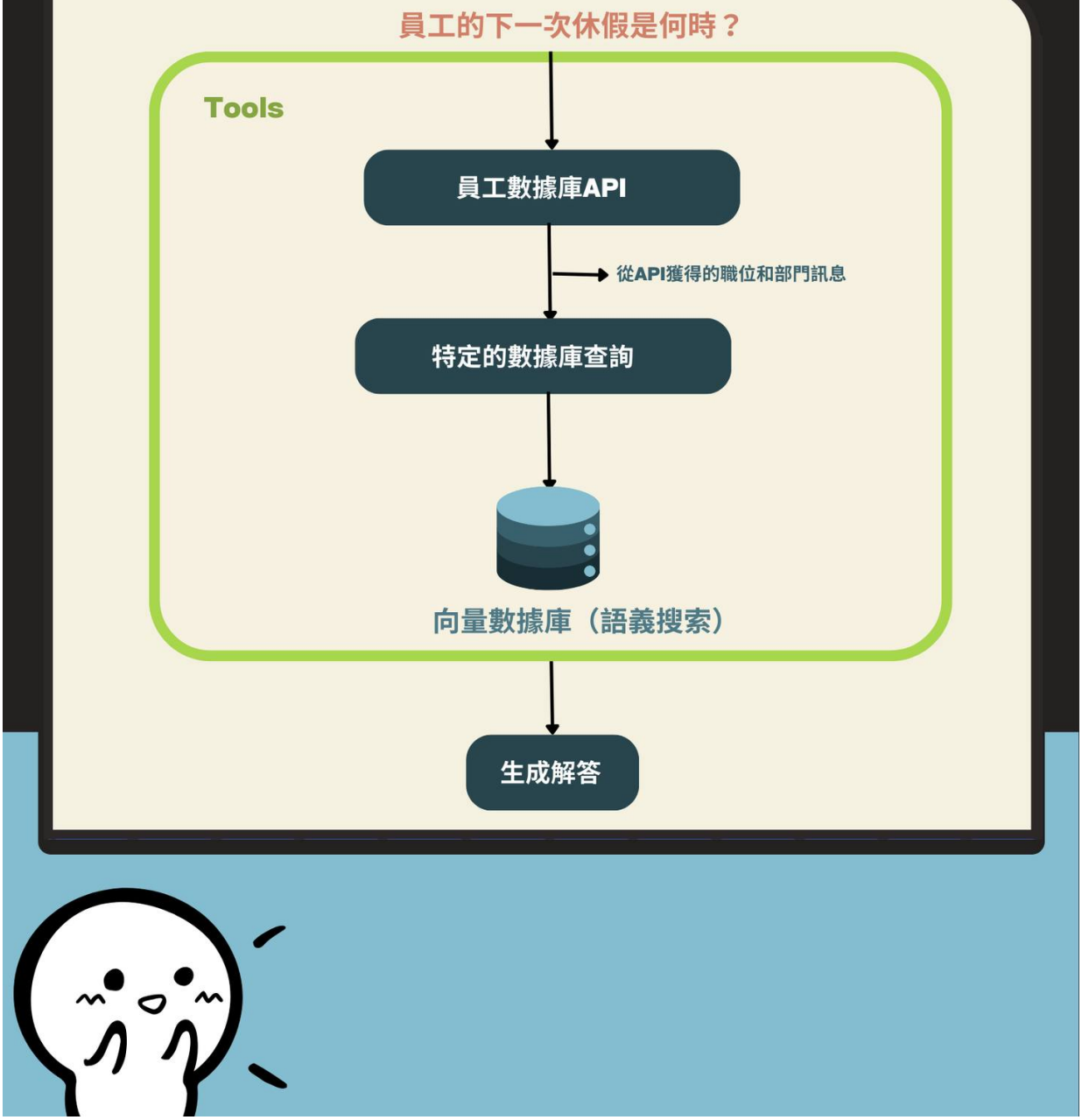


核心概念

- 路由代理：可以將查詢分配到一個或多個工具中，根據查詢的需求來決定使用哪種工具來處理。
- 多工具協作：一個查詢可以被路由代理分配到多個工具中進行處理，以獲得綜合的答案。

Tools For Agents

大型語言模型和其他智能代理可以使用多種工具來處理查詢，
包括API調用和數據庫查詢，這使得處理更加靈活和高效



参考資料

https://www.reddit.com/r/ChatGPT/comments/1d17eoa/how_agentic_rag_solves_problem_with_current_rag/?rdt=47727