圖 1-7 這張圖改編自 OpenAI 的那篇科學論文，其中詳細介紹了整個程序。

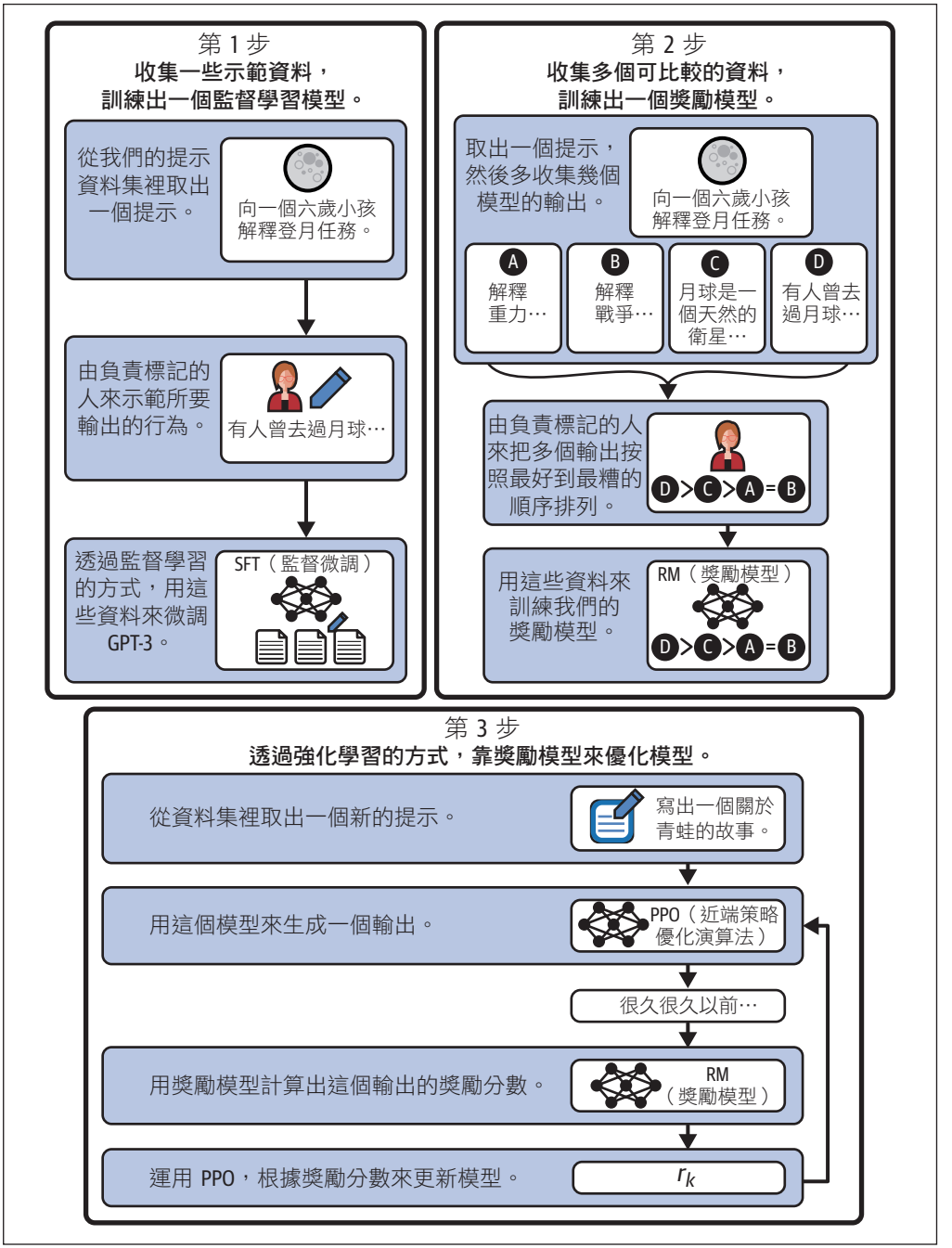

圖 1-7　訓練出 Instruct 模型的幾個步驟（根據 Ouyang 等人的圖片重新繪製）

對於開發者來說，外掛和 GPT 確實可以帶來許多全新的潛在機會。未來每一家公司或許都希望能擁有自家的 LLM 外掛或 GPT。OpenAI 特別開設了一個商店（*https://oreil.ly/LEa6V*），讓 OpenAI 的合作夥伴與社群都可以把自己所開發的好幾千個 GPT 分享出來。OpenAI 也表示，GPT Builder 營收計畫（GPT Builder revenue program）會在 2024 年上半年推出，一開始會先在美國上線。截至撰寫本文的此時，還沒有更多關於此營收計畫的其他資訊。不過可以預期的是，透過外掛或 GPT 添加 AI 功能的應用程式，數量有可能會非常龐大。

總結

LLM 從最簡單的 n-gram 模型開始，一直到 RNN、LSTM 和 Transformer 高階架構，已經走過了很長的一段路。像 LLM 這樣的電腦程式，其實是利用機器學習的技術，去分析大量的文字資料，然後去處理、生成一些很類似人類所慣用的語言。在使用了自注意力和交叉注意力的機制之後，Transformer 更是大幅增強了語言理解的能力。

本書打算探討的就是如何善用 GPT 模型，因為在理解、生成前後文方面，GPT 模型確實可以提供一些非常高階的功能。只要用它來打造應用程式，就可以超越傳統 BERT 或是 LSTM 模型的能力限制，提供一種很類似與真人互動的體驗。

自 2023 年初以來，GPT 模型在 NLP 領域已展現出非凡的能力。以結果來看，它可說是為各個產業 AI 應用的快速發展，做出了相當大的貢獻。從 Be My Eyes 之類的應用，到 Waymark 之類的平台，我們已經可以看到各式各樣不同的使用情境，這些全都可以用來證明，這樣的模型確實有可能徹底改變我們與技術互動的方式。

很重要一定要記住的是，使用這些 LLM 還是有一定的潛在風險。身為一個使用 OpenAI API 的應用程式開發者，你一定要確保你的使用者真正瞭解，模型都會有出錯的風險，而且要懂得去查證 AI 所生成的資訊。

下一章我們會提供一些工具和資訊，讓你可以把 OpenAI 模型當作服務來使用，這樣你就可以真正加入我們，經歷這一切令人難以置信的轉變了。

- API 閘道器（gateway），負責管理使用者從瀏覽器所發出的請求。
- 使用者服務，負責管理使用者。這個服務會去存取資料庫。
- 內容服務，負責執行一些內容生成和處理相關的任務。這個服務會去調用 OpenAI API。

在這樣的做法下，OpenAI API 就會被視為一個外部的服務，然後透過應用程式的後端來進行存取。

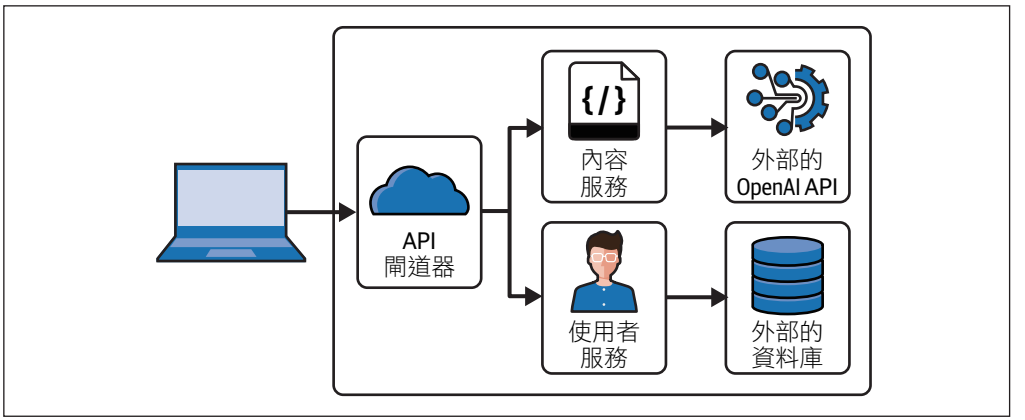

圖 3-1　把 OpenAI API 當成一個外部服務，整合到一個標準的 Web App 架構中

在整個架構中，應該只有你的內容服務，可以透過安全的方式存取到你的 API 金鑰。

針對如何打造出可以運用 OpenAI API 的應用程式，我們現在已經奠定了一定的基礎，接下來可以進入構思階段了。下一節會介紹一下 LLM 的各種能力，協助你把這些能力轉化成潛在的應用。

把 LLM 的各種能力整合到你的專案中

下面有很多不同的解決方案，可以讓你善用 GPT 模型的各種能力，進而從中受益。

對話能力

這種解決方案其實就是讓使用者透過你所定義的介面，直接去存取 GPT 模型，如圖 3-2 所示。使用者與你的系統進行對話的方式，其實就很類似他們在與 ChatGPT 進行對話。為了讓這種解決方案滿足特定的需求，通常都會先給 GPT 模型提供一些特定的提示，或是對模型進行微調。

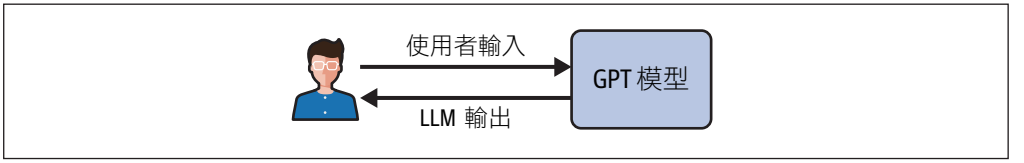

圖 3-2　使用者和 GPT 模型之間的對話流程

舉例來說，你可以針對你所在的城市打造出一個旅遊指南。你的這個對話代理，一開始的初始提示可能是：「*You are a helpful assistant for tourists visiting Lyon, France. Answer the questions as best as you can, in a helpful way, providing information on local attractions, dining options, and transportation.*」（你是一個非常有用的助理，服務對象是遊覽法國里昂的遊客。請盡量以一種很有幫助的方式來回答問題，以提供當地相關的景點、餐飲選擇和交通資訊。）這應該就是你要傳送給 API 的訊息列表其中的第一條訊息。在接下來的訊息中，使用者就可以開始與 LLM 進行後續的對話。

在這樣的系統下，你可以充分發揮模型的對話能力，而且可以根據你的需求進行調整，並針對程式的外觀和整體給人的感覺，進行一些客製化的調整。

這裡有三個面向需要特別留意：

- **提示工程**（*Prompt engineering*），一定要確保聊天機器人沒有忘掉最初的目的。
- **防護機制**（*Guardrails*），目的是控制幻覺和提示注入攻擊。
- **成本**，如果使用的是你所提供的 API 金鑰，一定要確保可以掌控使用的情況。你或許並不希望使用者進行永無休止的對話。

本章和下一章都會再進一步詳細討論這些主題。

語言處理能力

這次的情況，使用者並不知道背後其實有 LLM 的存在。或是你並不直接面對使用者，而是有另一個程式會來使用你的解決方案，如圖 3-3 所示。

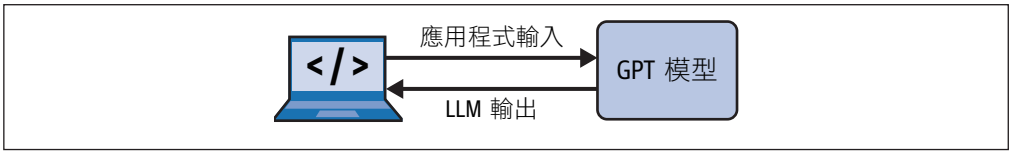

圖 3-3　應用程式與 GPT 模型互動的流程

這裡之所以使用 GPT 模型，並不是為了進行對話，而是為了善用模型的自然語言能力。舉例來說，GPT-4 已被證明能夠妥善處理分類問題，例如可以用下面這樣的提示來進行情緒分析：「*Is the sentiment in the following text positive, negative, or neutral?*」（下面的文字裡所帶有的情緒是正面的、負面的還是中性的？）

LLM 可以用來總結文字、對文件進行排名、執行翻譯任務、提取出語義關係、生成文字等等。這些能力可以讓你用來作為專案的核心能力，如下一節的前兩個範例所示。或者你也可以把它添加到你的系統中，以提供更好的使用者體驗或是作為額外的功能。舉例來說，我們來看一個系統，它可以從一個提供各種服務的平台，收集到各種回饋意見。其中有許多用一般的英文寫下來的回饋意見，就可以交給 GPT 模型去處理，例如提取出關鍵字、給予評分，進行整體性總結，或是做個簡短的說明。我們也可以利用 GPT 模型的輸出來向管理員發出警報，以偵測出特定的問題或是詐騙的情況。

你在前面的章節中，已經接觸過**檢索增強生成**（RAG；retrieval-augmented generation）的概念。這個技術也是一個可以用來展現自然語言處理（NLP）能力的高階應用範例。RAG 的構想如圖 3-4 所示：

1. 針對你的知識庫，建立相應的內嵌。
2. 針對查詢文字或關鍵字，建立相應的內嵌。
3. 用查詢文字相應的內嵌，對知識庫相應的內嵌進行查詢，以檢索出相關的資料。

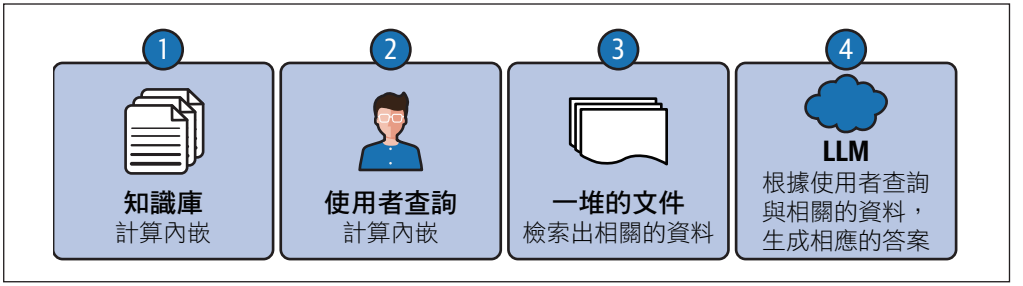

圖 3-4　RAG 的幾個步驟

完成第 3 步，你就已經用語義搜尋的做法完成了資料檢索。到這一步為止，只需要用到內嵌模型，還用不到 LLM。內嵌模型就可以讓你找出**意義上相近**的結果，而不只是找出**字面上相符**的結果。不過，在運用 RAG 的概念時，你還需要多做一個步驟：

4. 運用 LLM 來分析之前語義搜尋所送回來的相關資料，然後再用很有條理的方式做出精確的回答，以作為查詢的回應。

資訊「檢索」的結果，可以「增強」LLM 的「生成」能力，這就是「**檢索增強生成**」這個名稱的由來。在步驟 4 裡調用了 LLM 之後，就可以提供精確又容易理解的回答，而不只是那種還要人工去進行分析與總結的文件提取結果。如果想深入瞭解 RAG 技術，請參見第 4 章的詳細實作設計。

這樣的做法充滿了無限的可能性，而且比起那種直接面對使用者的聊天機器人更容易進行管理。由於任務非常具體而明確，因此產生幻覺的風險也會受到一定的限制。而且因為不會牽涉到任何的對話，因此成本也更容易掌控。

你還是需要留意提示注入攻擊（prompt injection）的問題——如果 GPT 模型所要處理的文字是來自使用者，這樣還是有提示注入攻擊的風險。（關於這個主題更多的資訊，請參見本章隨後第 147 頁的「支援 LLM 的 App 各種可能的漏洞」。）

人機互動能力

滑鼠和 GUI（圖形使用者介面）在 20 世紀 70 年代末期首次商業化，這可說是人機互動的第一次革命。有人說 LLM 是第二次革命，因為它可以讓使用者透過自然語言與電腦系統進行互動。

這次的想法是，利用 LLM 把使用者的輸入處理成另一種格式，讓軟體應用程式的其他部分可以進一步進行解析，如圖 3-5 所示。

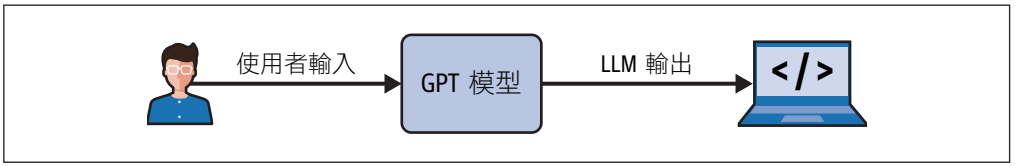

圖 3-5　把 LLM 用來作為人機介面的流程

在這樣的精神下，網頁表單就可以改用一個很大的文字框來取而代之。使用者不需要再一一填寫各個欄位，只要在文字框裡填寫詳細資料就可以了。然後，只要用 GPT 模型去處理這個輸入，就可以把它轉換成符合原始表單的資料。

舉例來說，使用者可以在電子商務網站上直接寫出他所要的查詢 —— *I am looking for blue or red shoes, leather, size 7*（我想找藍色或紅色的鞋子，皮鞋，尺寸是 7 號）——而不用在列表裡用人工方式進行選擇與篩選。OpenAI 模型可以把這樣的查詢，處理轉換成下面的格式：

```
{
   "type": "shoes",
   "material": "leather",
   "size":7,
   "color":[
      "blue",
      "red"
   ]
}
```

這個 JSON 格式的輸出，就可以交給應用程式的其他部分進一步使用了。

這個能力與前一節所描述的 NLP 能力有所重疊，但目標是不同的。前一節所描述的解決方案，LLM 是在背後發揮作用，用來處理一些後端的任務。這裡的 LLM 則是直接面對使用者，在使用者與軟體應用程式之間扮演一個介面的角色。

把各種能力結合起來

前面所說的各種能力，也可以透過不同方式進行組合，進一步強化你的解決方案，或是建立全新的專案，如圖 3-6 所示。

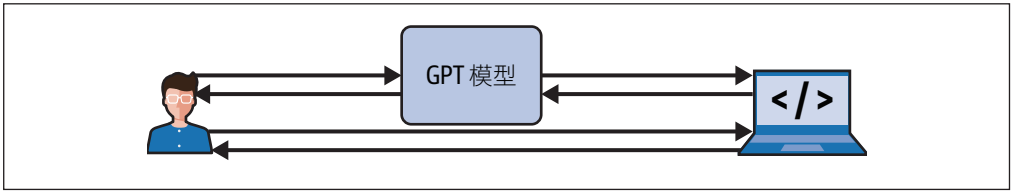

圖 3-6　同時運用 GPT 模型的對話能力、語言處理能力和人機互動能力的流程

舉例來說：

對話能力與自然語言處理能力

RAG 系統並不一定要包含聊天機器人的功能——但如果有這個功能好像也很自然。這樣就可以讓使用者重新微調他們的查詢文字、詢問更多的資訊等等，讓整個系統變得更加強大。隨後的專案 3 描述的就是這樣的一個範例。

對話能力與人機介面能力

這樣就可以創建出一個助理，有能力去執行使用者所請求的各種任務。隨後的專案 4 描述的就是這樣的一個範例。

下一節會提供一些具體的範例專案，其中包含能夠有效實作出這些策略的程式碼範例。由於只是範例，所以我們並不會重複說明 API 金鑰管理和安全性實作相關的細節。如果你想與他人分享你的應用程式，請別忘了我們之前所提過的一些推薦做法。

範例專案

本節的目標就是希望能夠啟發你去打造出一些應用，更充分利用 OpenAI 的各種服務。你應該會發現，這裡所提供的並不是個非常詳盡的範例列表，因為各種可能性可以說是無窮無盡，而且本章的目標只是讓你大概瞭解一下各種可能的應用方式，並深入去研究某些特定的使用情境。

這個專案展示了 LLM 的文字生成能力。正如前面所示，只需要幾行程式碼，你就可以打造出一個簡單但非常有效率的工具。如果想讓這個專案繼續發展，你可以再運用提示工程的技術，來回答更複雜的指示，或是利用微調的方式，來自定義輸出的風格、語氣或格式；隨後你在第 4 章就會看到相應的做法。

如果你想用我們的程式碼親自嘗試一下，可以去找我們的 GitHub 程式碼儲存庫（*https://oreil.ly/DevAppsGPT_GitHub*），然後再隨意調整一下提示的內容，嘗試做出各種不同的要求！

專案 2：YouTube 影片總結——語言處理

事實證明，LLM 很善於進行文字總結。在大多數情況下，它會設法提取出核心的構想，並重新表述原始的輸入，以生成一些感覺上比較清晰而流暢的總結。文字總結的能力，在許多情況下都是很有用的：

媒體監控

快速瞭解概況，而不會被過多的資訊所淹沒。

趨勢觀察

針對科技新聞或團體學術論文，生成相應的摘要，並得出有用的總結。

客戶支援

生成文件概要說明，讓你的客戶不會被一大堆不特別有用的資訊所淹沒。

Email 略讀

針對 email 裡的內容製作出最重要的資訊，並避免太多的 Email 超出你的負荷。

在這個例子裡，我們會針對 YouTube 影片進行總結。你可能會覺得有點驚訝——我們如何把影片輸入到語言模型中呢？

這裡有幾種可能的做法：

1. 直接從 YouTube 提取出影片的字幕，然後用 GPT 模型來對字幕進行總結；這個解法只會去分析影片的聲音部分。

2. 運用 GPT-4o 的視覺能力，以及它很大的前後文視窗，來分析影片的靜態畫面；這個解法只會去分析影片的圖像部分。
3. 結合這兩種做法，來分析影片的聲音與圖像。

我們會先從第一個選項開始著手，因為它比較簡單、比較便宜，只要 `gpt3.5-turbo` 就夠用了。

你應該很輕鬆就可以存取到 YouTube 影片的字幕。在你所選擇觀看的影片下方，就可以找到影片的說明。如圖 3-7 所示，先點擊「...more」（... 更多），然後再選擇「Show transcript」（顯示字幕）。

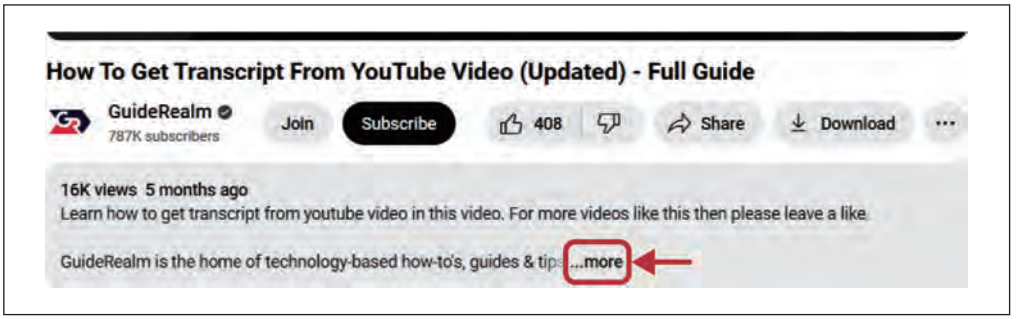

圖 3-7　取得 YouTube 影片裡的字幕

這樣就會出現一個文字框，其中包含影片的字幕，看起來應該如圖 3-8 所示。這個文字框還可以讓你切換時間戳。

如果你只打算對一個影片執行一次這樣的操作，只需要把 YouTube 頁面上所顯示的字幕複製貼上即可。要不然的話，你或許就需要用到更自動化的解決方案，例如 YouTube 所提供的 API（*https://oreil.ly/r-5qw*），它就可以讓你用寫程式的方式與影片進行互動。你可以直接使用這個 API 來取得字幕（`captions`；*https://oreil.ly/DNV3_*），也可以用第三方的函式庫（例如 *youtube-transcript-api*；*https://oreil.ly/rrXGW*）或 Web 工具程式（例如 Captions Grabber 這個字幕擷取工具；*https://oreil.ly/IZzad*）。

一旦取得字幕，你就可以去調用 OpenAI 模型來進行總結。如前所述，我們在這個任務中使用的是 GPT-3.5 Turbo 模型。這個模型非常適合這個簡單的任務，而且在撰寫本文的此時，它就是最便宜的選項。

法就是去測量文件裡出現某些術語的頻率和相關性，來識別出最相關的文件片段。我們會用 RRF 排名倒數融合演算法，把關鍵字搜尋和內嵌搜尋這兩種方法所找出的結果融合起來。它的原理就是把這兩種不同的搜尋策略所檢索出來的文件片段進行排名，然後優先考慮搜尋結果裡排名較高的文件片段。這種混合式搜尋的做法，可以同時利用到精確比對的能力（例如關鍵字搜尋做法）以及前後文理解的能力（例如向量內嵌搜尋），提高搜尋結果的相關性，從而提供更完整的搜尋體驗。

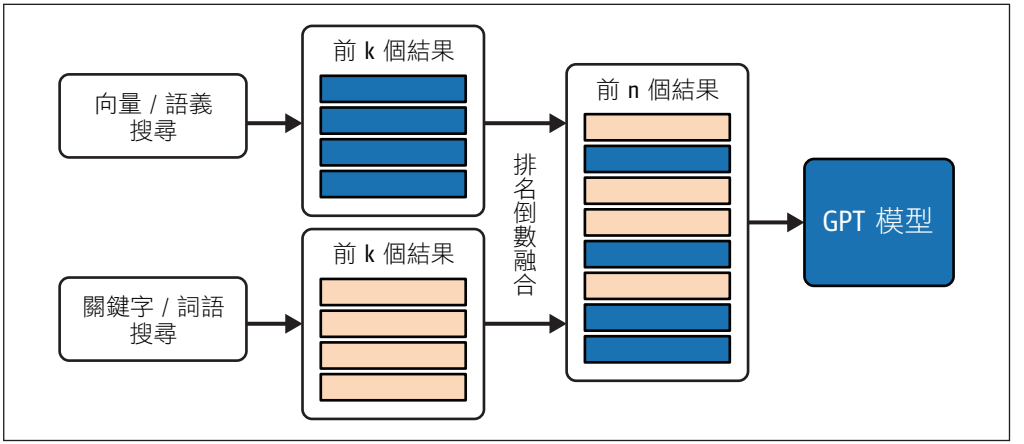

圖 4-13　混合式搜尋的原理

LangChain 和 LlamaIndex 都有提供一些解決方案，可以輕鬆實作出這個做法。

後處理

接下來我們還可以針對結果進行後處理（Postprocessing），例如運用先前所導入的詮釋資料進行篩選或重新排名，或是利用一些方式來進行轉換，以協助你的 LLM 回答當初那個最原始的問題。這裡可以採用的做法，並沒有什麼特別的限制。

所有這些改進的做法，全都總結在圖 4-14 裡了。

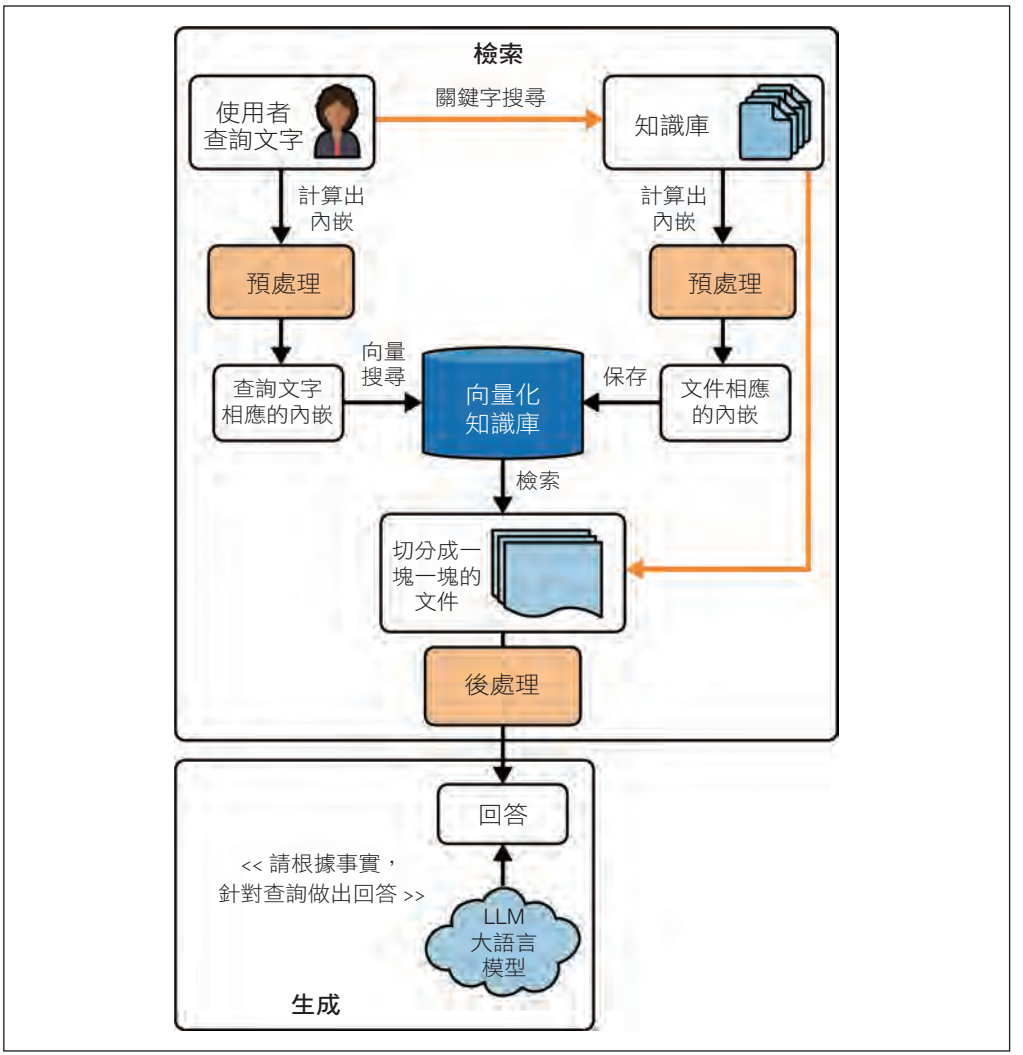

圖 4-14　進階的 RAG 流程

進階 RAG 設計必須嘗試各種不同的做法、不斷分析結果並反覆嘗試，憑藉這些經驗才能獲得成功。關鍵就是要有效衡量出 RAG 設計的表現，才能靠迭代的做法取得真正的改進，而不只是依賴直覺。因此，我們可以根據搜尋的表現本身，以及 LLM 處理文件塊並給出相應回答的能力，來設定不同的評估策略。你可以在第 5 章所介紹的 LangChain 和 LlamaIndex 工具裡找到一些範例，或是使用 Ragas（*https://oreil.ly/yOmH7*）之類其他工具的一些範例，不然你也可以嘗試去實作出自己的評估做法。

這些技術並不互相排斥，事實上，如果可以把這些做法結合起來，或許就可以獲得最佳的結果。

OpenAI 提供了一個最佳化的流程，如圖 4-15 所示。

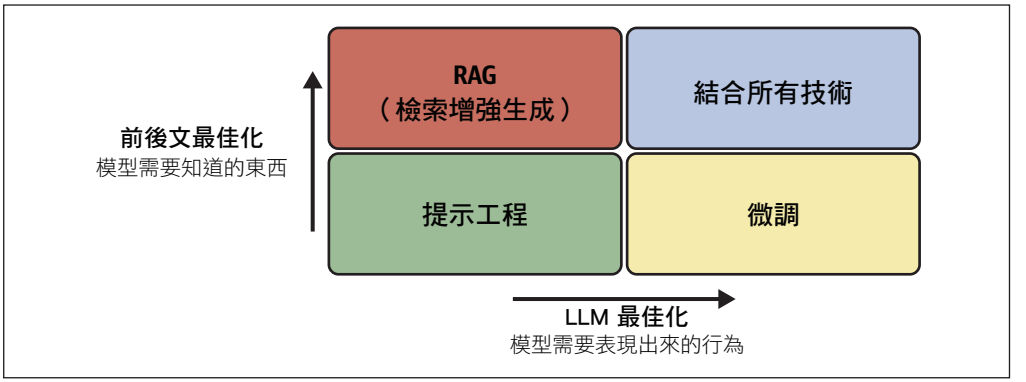

圖 4-15　OpenAI 所提供的最佳化流程

根據經驗法則，一開始總要先嘗試簡單的提示，然後再用少量樣本學習和提示工程技術來做點實驗。因為這樣你就能用比較低的成本，快速取得一些結果，然後很有效率地進行迭代。如果提示工程根本沒什麼效果，那麼 RAG 或微調之類的其他技術，或許也不會有什麼效果。

接下來就要判斷是否需要因為內容、風格或格式上的考量，而需要再進一步改進。如果內容是主要的問題所在，那麼 RAG 就是下一步不錯的選擇。如果是格式上或風格上的問題，就可以考慮採用微調的做法。

最後，把所有的做法結合起來，通常就是針對複雜的問題提供最佳結果的一種解決方式——RAG 可納入更多的知識，提示工程可改進 LLM 的生成結果，微調則可以用來確保所生成的回答更符合所預期的風格與格式。

我們在圖 4-16 中，嘗試把這整個程序描繪出來。

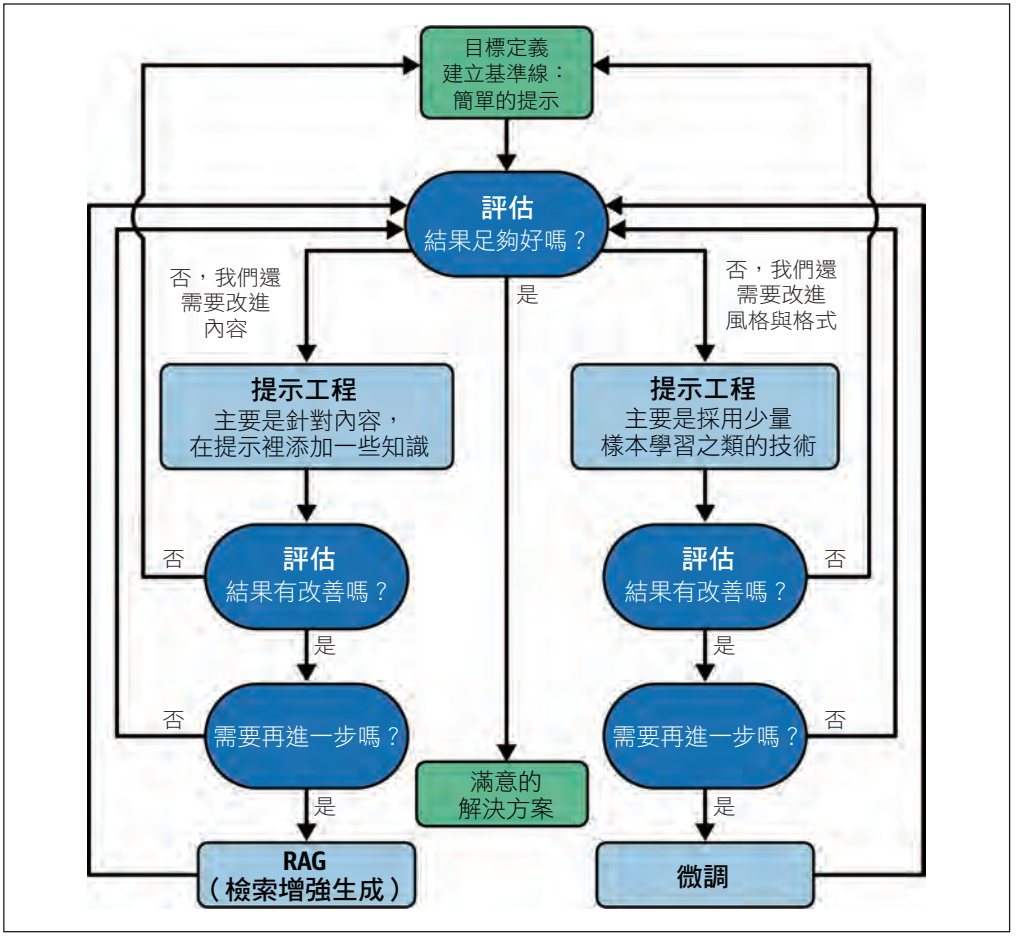

圖 4-16　根據不同情況進行優化的工作流程

這張圖的目的，就是作為整個工作流程的視覺化提醒，其中特別強調的是何時該去使用提示工程、RAG 和微調的做法。正如你所看到的，圖中有很多往回指的箭頭，這就表示我們很強調反覆迭代的重要性。這張圖裡也有很多的評估步驟，下一節就會詳細介紹。

評估

如果想改進你的系統，最關鍵的就是要用比較小的步伐，以反覆迭代的方式進行改進，而不要一下子就跳到比較複雜的解法。因此，對結果進行評估是非常重