前言

使用機器學習驅動應用的目標

過去十年來，機器學習（ML）被越來越廣泛地用於驅動各種產品，例如：自動化的客服系統、翻譯服務、推薦引擎、詐欺偵測模型，以及更多更多。

令人驚訝的是，在教導工程師和科學家們如何建立這類產品的可用資源卻很少。很多書籍和課程都會教導如何訓練 ML 模型或是如何建立軟體專案，但很少同時整合這兩個世界，去教如何建立由 ML 驅動的實際應用。

部署 ML 作為應用的一部分，需要混合創意、強大的工程實踐，以及一個分析性的思維模式。打造 ML 產品的挑戰性是聲名狼藉的，因為它們不僅僅是簡單地在一個資料集上訓練模型，還需要更多方法，像是為給定特徵選擇正確的 ML 方法、分析模型錯誤和資料品質問題，以及驗證模型結果以確保產品的品質，這些都是 ML 建立過程的核心中具挑戰性的問題。

本書涵蓋了此過程的每個步驟，透過分享方法的組合、範例程式，以及從我和其他經驗豐富從業者的建議，來幫助您完成每個步驟。我們將介紹設計、建立和部署 ML 驅動應用所需的實際技能。本書的目的是幫助您在 ML 過程中的每個部分獲得成功。

使用 ML 打造實際應用

如果您經常閱讀 ML 論文和公司的工程部落格，您也許會對線性代數方程式的組合和工程術語感到不知所措。這個領域的混合特性使得許多能貢獻他們各種專業的工程師和科學家們對 ML 領域感到卻步。同樣地，創業者和產品領導者通常困在把他們的商業構想和當今 ML 所能實現的（和未來的可能性）結合在一起。

這本書介紹我在多家企業資料團隊中的工作，以及透過我在 Insight Data Science 領導人工智慧學程的工作時，幫助了數百位資料科學家、軟體工程師和產品經理建立應用 ML 的專案中，所學到的經驗與教訓。

本書的目標是分享逐步的實際引導以建立 ML 驅動的應用。這是實際而且著重在具體的指導和方法以幫助您建立原型、疊代和部署模型。因為它橫跨了廣泛的主題，只有在每個步驟中有需要的情況我們才會更仔細的探討，如果您對這些主題有興趣，我會盡可能提供資源來幫助您更深入了解介紹的主題。

重要的概念會透過實際的範例進行說明，包含在本書最後會從概念到部署模型的一個案例研究。大部分的範例都會附有圖示說明，而且許多會包含程式碼。在本書中所有使用到的程式碼都可以在本書的 GitHub 儲存庫（*https://oreil.ly/ml-powered-applicataions*）中找到。

由於本書著重在描述 ML 的過程，所以每章都是建立在前一章的概念之上。因此我建議您依序閱讀本書，以便您能了解每個成功的步驟是如何置於整個過程當中。如果您正在尋找要探索 ML 過程中的一小部分，搭配一本較專業的書可能會更好。如果是這樣，我會分享一些建議。

額外資源

- 如果您想充分了解 ML 以從頭開始撰寫自己的演算法，我推薦 Joel Grus 撰寫的《*Data Science from Scratch*》。如果您在追求深度學習理論，由 Ian Goodfellow、Yoshua Bengio、Aaron Courville 所撰寫的《*Deep Learning*》（MIT Press），是一個綜合性的資源。
- 如果您想知道如何在特定資料集上有效率又準確地訓練模型，Kaggle（*https://www.kaggle.com/*）和 fast.ai（*https://fast.ai*）是很棒的地方可以去看看。
- 如果您想學習如何建立需要處理大量資料的可擴展應用，我推薦您去看 Martin Kleppmann 撰寫的《*Designing Data-Intensive Applications*》（O'Reilly）。

如果您已經有程式撰寫經驗和一些基本的 ML 知識，而且想要打造 ML 驅動的產品，這本書將會引導您走過從產品概念到可交付原型的整個過程。如果您已經是資料科學家或 ML 工程師，這本書將增加您 ML 開發工具的新技術。如果您不知道如何撰寫程式但是要與資料科學家合作，只要您願意跳過一些深入的程式範例，那麼這本書就可以幫助您了解 ML 的過程。

首先，我們將深入了解 ML 的實際含意。

實際的 ML

這段介紹的目的是將 ML 視為一個運用資料模式的過程，以自動化地調校演算法。這是個一般的定義，因此當您聽到很多應用、工具和服務正開始導入 ML 作為它們運作的核心時，您將不會感到驚訝。

其中一些任務是面向使用者的，例如：搜尋引擎、社群平台上的推薦、翻譯服務、自動偵測照片中的熟悉臉孔、遵循語音命令的指示，或是試圖在電子郵件中提供有用的建議以完成句子。

有些工作以較不顯而易見的方式，悄悄地過濾垃圾郵件和詐欺帳戶、投放廣告、預測未來的使用模式以有效率地分配資源，或實驗每位使用者的個人化網站體驗。

目前有許多產品利用 ML，甚至有更多也可以這樣做。實際的 ML 指的是識別可以從 ML 中受益的實際問題，並為這些問題提供成功解決方案的任務。從高階的產品目標到 ML 驅動的結果是一項具有挑戰性的任務，在本書中會試著幫助您去完成它。

一些 ML 課程會藉由提供資料集並訓練模型來教導學生關於 ML 的方法，但是在資料集上訓練出一個演算法只是 ML 過程的一小部分。引人注目的 ML 驅動產品仰賴的不僅僅是一個彙整的準確度分數而已，而是一個長期過程的結果。本書將從構想開始接續到生產的整個過程，以一個示範應用說明每個步驟。我們將分享每天和部署這類系統的應用團隊共事中，所學到的工具、最佳做法和常見陷阱。

本書介紹的內容

為了介紹打造 ML 驅動應用的主題，本書的焦點是具體且實際的。特別的是，本書的目的在說明整個打造 ML 驅動應用的過程。

為此，我會先描述過程中每個步驟的處理方法。接著，我會以一個專案的例子作為個案研究來說明這些方法。本書還包含很多企業中實際的 ML 例子，以及特別訪談與已經建立並維護生產環境中 ML 模型的專家。

ML 的完整過程

為了成功地向使用者提供 ML 產品，您需要做的不僅僅是簡單地訓練出一個模型，您需要考慮周全地將您的展品需求**轉換**為 ML 問題、**收集**足夠的資料、有效率地在模型之間進行**疊代**、**驗證**您的結果並以穩固的方式**部署**它們。

建立模型通常只是 ML 專案總工作量的十分之一，掌握整個 ML 管線對於成功建立專案、在 ML 面試中成功，以及在 ML 團隊中成為傑出貢獻者至關重要。

技術性且實際的案例研究

雖然我們不會用 C 語言從頭實作演算法，但我們將透過使用更高級抽象的函式庫和工具來保持實際和技術性。我們將在走過本書內容時建立一個 ML 應用範例，從最初的構想到已部署的產品。

我將會適時使用程式碼片段來說明關鍵概念，以及利用圖片描述我們的應用。學習 ML 的最好方法是實踐它，因此，我鼓勵您閱讀本書時，複製範例並改寫它們，以打造您自己的 ML 應用。

真實的商業應用

在本書中，我會收錄來自 ML 領導人的對談和建議，這些領導人曾在科技公司如：StitchFix、Jawbone 和 FigureEight 的資料團隊中工作。這些討論將涵蓋與數百萬使用者建立 ML 應用後獲得的實際建議，這會導正一些關於是什麼使資料科學家和資料科學團隊成功的普遍誤解。

先備知識

本書假設您已對程式設計有些熟悉度。我將主要使用 Python 作為技術性範例並假設讀者已熟悉其語法。如果您想更新您的 Python 知識，我推薦由 Kenneth Reitz 和 Tanya Schlusser 撰寫的《*The Hitchhiker's Guide to Python*》（O'Reilly）。

此外，雖然我會定義本書中大部分提到的 ML 概念，但不會介紹所有 ML 演算法使用的內部運作方式。大多數演算法是標準的 ML 方法，會在如第 x 頁「額外資源」中所提到的導論式 ML 資源中被介紹到。

我們的案例研究：ML 輔助寫作

為了更具體說明這個構想，閱讀本書時我們將一起建立一個 ML 應用。

作為案例研究，我選擇了一個應用可以準確地說明疊代和部署 ML 模型的複雜性。我還想介紹一個可以產生價值的產品，因此我們將會實作一個由 *ML 驅動的寫作助手*。

我們的目標是建立一個可以幫助使用者寫得更好的系統。特別的是，我們會致力於幫助人們寫出更好的問題，這似乎是一個非常模糊的目標，但因為一些關鍵因素，它是一個很好的範例。當我們觀察此專案時，我將對它做更清晰的定義。

文字資料是無所不在的

文字資料可被大量地使用於您可以想到的大多數使用案例中，而且是許多實際 ML 應用的核心。我們是否試著更好地了解我們的產品評論、更精準地分類前來的幫助請求，或是量身製作我們潛在受眾的促銷訊息，這些我們都將會用到並產生文字資料。

寫作助手是有用的

從 Gmail 的文本預測功能到 Grammarly 的智慧拼字檢查器，ML 驅動的寫作輔助編輯器已經證明能以多種方式傳遞價值給使用者，這使得我們特別有興趣去探索如何從頭開始建構它們。

ML 輔助寫作是獨立的

很多 ML 應用只有在緊密整合進廣泛的生態系統中時才能運作，例如：叫車公司的預計到達時間（ETA）預測、線上零售業的搜尋和推薦系統，以及廣告出價模型。儘管文字編輯器可以從整合進文件編輯生態系統中受益，但是它可以透過自己證明價值，而且可以透過簡單的網站公開。

在這整本書中，這個專案將會讓我們突顯出打造 ML 應用的挑戰，以及我們建議的相關解決方案。

ML 過程

從構想到部署 ML 應用是條漫長又曲折的道路。在看過許多公司和個人建立了這樣的專案後，我已經確認了四個關鍵的連續階段，每個階段都將會在本書中的一個部分介紹。

1. **辨認正確的 ML 方法**：ML 的領域廣泛，而且通常會針對給定的產品目標提出多種解決方案。特定應用問題的最佳方法取決於許多因素，例如：成功的評估標準、資料可用性，以及任務的複雜性。此階段的目標是設定正確的成功標準，並確認合適的初始資料集和模型選擇。
2. **建立初始原型**：在模型上進行工作之前，先從建立一個端對端的原型開始。此原型應該旨在解決不涉及 ML 的產品目標，並讓您能夠確定如何最好地應用 ML。當原型被建立好，您就應該對是否需要 ML 有所想法了，並且應該能夠開始收集資料集來訓練模型。
3. **對模型進行疊代**：現在您有了資料集，您可以訓練模型並評估其缺陷。此階段的目標是在錯誤分析和實作中交替重複。提升此疊代循環的速度是提升 ML 開發速度的最佳方法。
4. **部署和監視**：當模型顯示出良好的效能，您應該挑選一個適當的部署方案。當部署好後，模型通常會以無法預期的方式故障。本書的最後兩章將介紹減少和監視模型錯誤的方法。

有很多基礎內容要介紹，所以讓我們進入並開始學習第 1 章吧！

本書編排慣例

本書使用以下的編排慣例：

斜體（*Italic*）

代表新的術語、URLs、email 地址、檔名和副檔名。中文以楷體標示。

定寬體（`Constant width`）

用來列出程式，以及在內文引用的程式元素，例如：變數或函式名稱、資料庫、資料型態、環境變數、陳述式及關鍵字。

定寬粗體（**`Constant width bold`**）

代表應該由使用者依字面輸入指令或其他文字。

等寬斜體（*`Constant width italic`*）

代表應該替換成使用者提供的數值，或由上下文決定的數值。

第一部分

發現正確的 ML 方法

絕大多數的人或公司都可以掌握到他們想處理的問題，比如說：預測哪些顧客即將離開線上平台，或者是如何建造一架無人駕駛機來跟隨使用者滑雪下山。同樣地，大部分的人也都能快速地學習如何訓練一個模型去分類顧客，或是在特定資料集上以合理的準確度來偵測物體。

可是，很少人或公司能做到理解問題、評估最佳解決方案、創造用 ML 解決問題的計畫，並自信地執行該計畫。這些技能通常需要在經驗中學習，同時經歷過好幾個野心過高、趕不上截止日期的專案計畫。

對一個特定產品來說，可能有許多種 ML 解決方案。在圖 I-1 中，您可以在左側看到一個寫作助手工具的樣品，包含寫作建議和讓使用者提供回饋的機會；在圖的右側則是能提供這種建議的可能 ML 方法示意圖。

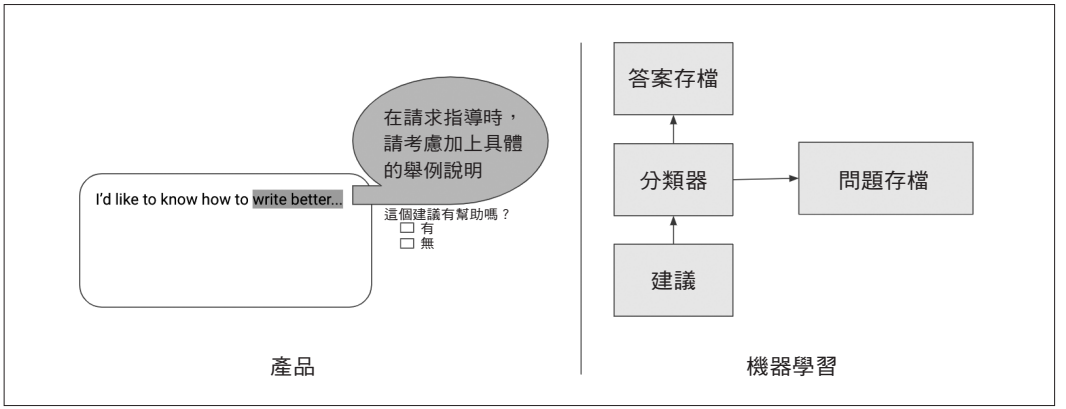

圖 I-1 從產品到 ML

此部分會先說明打造應用不同的可能方法，並從中選擇一個適合的方法。接著再深入探討使模型效能指標與產品要求一致的方法。

因此，我們將進入以下兩個主題：

第 1 章

在本章結束時，您會對打造應用有個構想、評估是否有可能解決、決定是否需要使用 ML，並清楚從哪類模型開始著手最為適合。

第 2 章

在本章中，我們會說明如何在您應用目標的情境下，準確地評估您的模型效能，以及如何使用此評估來達成定期的進展。

從產品目標到 ML 建構

ML 使機器從資料中學習並以機率的方式呈現，同時透過最佳化特定目標函數來解決問題。這有別於傳統程式設計，在傳統程式設計中，程式設計師需要撰寫出能描述**如何解決問題**、包含完整步驟的程式碼。因此，**當我們無法定義出一個啟發式**（*heuristic*）**解決方法時，建立 *ML* 的系統**會特別有用。

圖 1-1 說明了兩種方法來撰寫辨識貓咪的系統。左側的程式是由手動撰寫的解決步驟所組成；右側是 ML 利用已標籤的貓狗照片資料集，使模型學習從圖片到類別的映射（mapping）。在 ML 的方法中，沒有一定能達到成果的規則，只有一組輸入和輸出的實例。

圖 1-1 從定義演算步驟到呈現資料的例子

ML 功能強大而且能創造全新產品，但因為它是基於對資料中模式的辨識，所以帶來了一定程度的不確定性。重要的是，應該確定產品中哪個部分能受益於 ML，以及從如何最小化使用者體驗不佳的風險的角度，來制定模型的學習目標。

比如說，手動撰寫從像素值中自動偵測圖片裡有哪個動物的完整步驟程式，幾乎是不可能的（而且極為費時）。但是藉由給卷積神經網路（CNN）看數千張不同動物的圖片，我們就能建立一個比人類分類更準確的模型。這吸引大家開始使用 ML 來處理這個任務。

另外，ML 較不適用的反例是，自動計算稅金的應用程式需要依賴政府提供的規則，而且報稅表上若出現錯誤會帶給使用者相當負面的體驗，所以用 ML 自動產生報稅表令人有所顧慮。

您絕不會想用 ML 來解決可管理、具有明確規則組合的問題，可管理的意思是您可以有信心地寫出且在維護上並不複雜的規則組合。

所以當 ML 打開了不一樣的應用世界後，重要的是要去思考哪些任務**能夠**而且**應該**由 ML 解決。當打造一個產品時，您應該從具體的商業問題出發，決定是否需要使用 ML，然後再尋找適合的 ML 方法，這將使您能盡快疊代這個過程。

本章中會從以下方法開始說明此過程：評估什麼任務能夠由 ML 解決、不同產品目標適合哪種 ML 方法，以及如何處理資料的需求。我會透過在第 xiii 頁「我們的案例研究：ML 輔助寫作」提過的 ML 寫作輔助編輯器案例，以及在 Monica Rogati 的訪談中來闡明這些方法。

評估可行性

由於 ML 不需要人類提供逐步指令就能完成任務，所以 ML 在一些任務上（例如：從放射影像中偵測腫瘤或是下圍棋）能比領域專家表現更好，以及一些對人類而言不可能達成的任務（例如：從上百萬篇的文章中進行推薦，或是把音響中的聲音改成像其他人的聲音）。

ML 直接從資料中學習的能力，使它應用在廣泛的領域中都相當有用。但是這也使得人們更難正確地判斷 ML 適合解決什麼問題。所以，相對於那些研究論文或是企業部落格發表的成功結果，更多的是不在檯面上、數百個聽起來合理結果卻完全失敗的 ML 構想。

儘管目前沒有任何方法能確保 ML 應用的成功，然而有一些準則可以幫助您降低處理 ML 專案時的風險。首要的是，每次都從產品目標開始，然後再決定如何把它處理到最好。

在這個階段，都應該對任何方法保持開放的心態，不論這個方法是否有用到 ML。當考慮使用 ML 時，要確認這些方法有多適用於產品，而不只是空想這些方法多麼有趣。

確認 ML 方法有多適用於產品時，您需要依循兩個步驟：（1）以 ML 的方式制定產品目標。（2）評估 ML 任務的可行性。依據您的評估，您可以再次調整您的產品目標直到滿意為止。接下來讓我們來探討這些步驟。

1 **以 *ML* 的模式制定產品目標**：當我們打造一個產品時，我們必須先考慮要提供什麼服務給使用者。如同我們在導論中提過的，我們會以幫助使用者寫出更好問題的編輯器案例，來說明這些概念。這個產品的目標很明確：在使用者撰寫好問題之後，讓他們能夠收到可行又有用的修改建議。然而，相較於制定產品目標，ML 建立在完全不一樣的方式之上，ML 關注的是它自己如何**從資料中學習出一個函數**，例如：學習將句子翻譯成另一種語言。因此，對於一個產品目標而言，經常有很多不同實作難度的 ML 演算法。

2 **評估 *ML* 任務的可行性**：所有 ML 的應用問題都不一樣！當我們了解 ML 的發展現況，就能知道現在已經可以在數小時內建立一個能正確分辨貓狗照片的模型，但像是要建立與人聊天的系統，仍是待解決的研究問題。為了能有效率地建立 ML 的應用，重要的是，同時考慮多個可能的 ML 架構，並從一些我們認為最簡單的架構開始。其中一個評估問題困難度的好方法就是同時考慮它需要哪一種資料，以及現有的模型是否能夠運用這種資料。

為了提供不同 ML 架構的建議並評估可行性，我們應該檢視 ML 應用問題裡的兩個核心面向：模型和資料。

我們先從模型開始說明。

模型

ML 有很多常見模型，但我們在這裡不會詳細介紹它們，您可以自行參閱本書中第 x 頁「額外資源」以獲得更完整的說明。除了常見模型之外，每週都有許多創新的模型、架構和最佳化方法被發表出來，光是在 2019 年 5 月，就超過了 13,000 篇論文被提交到

ArXiv（*https://arxiv.org*），ArXiv 是一個熱門的線上研究論文庫，這裡常常會發表新模型的論文。

因為說明不同種類的模型，以及如何應用到不同的問題上能幫助您評估可行性。所以，我在這裡將依據模型解決問題的方式進行簡單的分類，這些分類可以指引您選擇特定 ML 應用問題的解決方法。因為模型和資料彼此在 ML 中是高度相關的，您會發現這一節的內容和第 14 頁「資料類型」有一些重複。

ML 演算法是根據它們是否需要標籤來分類，標籤指的是模型輸入資料後的預期輸出。監督式（supervised）演算法即是使用有標籤的輸入資料集，目標是去學習輸入對標籤的映射；非監督式（unsupervised）演算法則不需要標籤；最後，弱監督式（weakly supervised）演算法則使用不完全理想、但某種程度上相似的輸出。

很多產品目標能同時以監督式和非監督式演算法處理。例如：我們可以建立一個不需要標籤的模型，來偵測有異於一般交易的詐欺偵測（fraud detection）系統，也可以訓練一個已經將交易資料手動標籤為「詐欺」或「合法」的監督式模型。

對大部分的 ML 應用而言，由於我們能透過標籤來評估模型的預測品質，所以監督式方法更容易驗證，而且因為我們能夠取得預期的輸出，所以也更容易訓練。雖然一開始建立有標籤的資料集可能會很耗時，但這會使後續建立和驗證模型更加容易。基於以上理由，本書大部分的內容都會說明監督式演算法。

確認模型會採用哪種輸入並產生哪種輸出，能顯著地幫助您縮小 ML 可能方法的範圍。以下任一種監督式 ML 方法都可能適用於您的應用中：

- 分類和迴歸（Classification and regression）
- 知識提取（Knowledge extraction）
- 使用者目錄推薦（Catalog organization）
- 生成式模型（Generative models）

在接下來的小節中，我會在這些方法上做進一步的延伸。當我們在了解這些不同的建模方法時，我建議您想想可以使用或可以收集到哪些資料，因為最後模型的選擇往往會受限於能否取得資料。

分類和迴歸

一些專案主要是在將資料有效分類到兩種或多種類別，或是一個連續型量尺（指的是**迴歸**而非**分類**）中的數值。雖然迴歸和分類的技術是不同的，但處理它們的方法通常有許多重疊的部分，所以我們在這裡把它們放在一起介紹。

分類和迴歸相似的理由之一是，大多數的分類模型會先輸出一個機率值，經過歸納後再決定目標對象應該屬於哪個類別。廣義上來說，分類模型也可以視為在機率值上的迴歸。

一般來說，我們會對個別資料進行分類，例如：分類電子郵件是否是垃圾郵件的過濾系統、分類使用者是詐欺或合法的詐欺偵測系統，或是分類骨頭是否有骨折的放射影像電腦視覺模型。

在圖 1-2 中，您可以看到一個根據情感和媒體的句子分類範例。

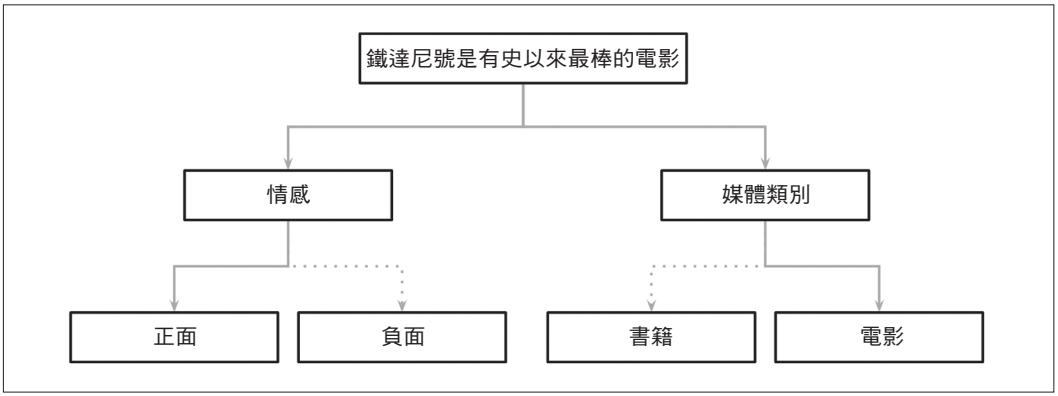

圖 1-2 分類一個句子到多個類別中

在迴歸的專案中，就不是把個別資料歸類到一個類別中，而是給予一個數值，比如：根據房子的房間數和座落地點等特徵來預測它的銷售價格。

在一些例子中，我們取得了一連串歷史的資料點（非單一個），來預測未來的某個事件，這種類型的資料通常被稱作「**時間序列**」（*time series*），而「**預測**」（*forecasting*）指的是利用這一連串的資料點預測未來的事件。時間序列資料能呈現病患的疾病史或是國家公園的持續參訪人數。因此，特徵和模型加上時間維度後，通常能對這類時間序列的專案有所幫助。

在其他例子中，我們試圖去偵測資料集中不尋常的事件，這就是所謂的「**異常偵測**」（*anomaly detection*）。當分類器試圖去偵測資料中的罕見事件，但卻難以準確預測時，這種情況就像是大海撈針一樣困難，所以通常需要使用其他方法。

好的分類和迴歸工作大多需要有意義的特徵工程（feature engineering）。在特徵工程中，特徵選擇（feature selection）是選擇出最有預測價值的特徵子集合，而特徵生成（feature generation）的任務是透過修改、組合既有特徵或資料集，進而產生有效預測目標的良好預測因子（predictor）。我們會在本書的第三部分更詳盡說明這些主題。

近年來，深度學習已經展現出為圖像、文字和聲音自動生成有用特徵的能力，將來它還可能會在簡化特徵生成和選擇上扮演重要角色。但就目前來說，特徵工程仍屬於 ML 流程中的一部分。

因此，我們能利用前面介紹過的分類或提供機率值的方法，來為 ML 寫作輔助編輯器產生有用的寫作建議。不過這需要仰賴模型本身的可解釋性（interpretability），這部分我們稍後再繼續討論！

並非所有應用問題的目的都是把資料歸於一個類別或數值。在一些案例中，我們希望以更細緻的層次從輸入中的某部分提取資訊，比如說知道一張圖片中哪裡有物體。

從非結構化資料中提取知識

結構化資料（*structured data*）指的是以表格形式儲存的資料，像是資料庫的表格和 Excel 的工作表。**非結構化資料**（*unstructured data*）則指的是非表格形式的資料集，包含文字（來自文章、評論、維基百科等）、音樂、影片、歌曲。

在圖 1-3 中，您可以看到左側結構化資料和右側非結構化資料的例子，知識提取模型著重在利用 ML 從非結構化資料中提取知識結構。

在文字資料的案例中，知識提取能增加評論的結構，例如可以訓練模型來提取評論中的不同面向，如：乾淨度、服務品質和價格，然後使用者就可以輕鬆取得那些他感興趣面向的評論。

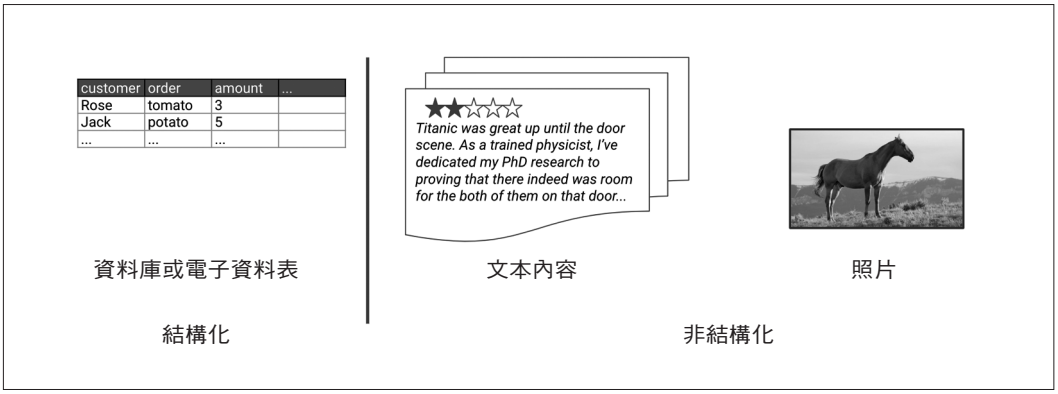

圖 1-3 結構化和非結構化資料類型的範例

在醫學領域中，我們能以醫學論文作為輸入的原始文本，來建立知識提取的模型，接著提取論文中所研究的疾病、相關的診斷與其成效資訊。在圖 1-4 中，模型輸入句子並提取哪些單詞是媒體類型、哪些單詞是電影標題。例如，對於粉絲論壇中常被討論的電影，這種模型能從它們的影評產生摘要。

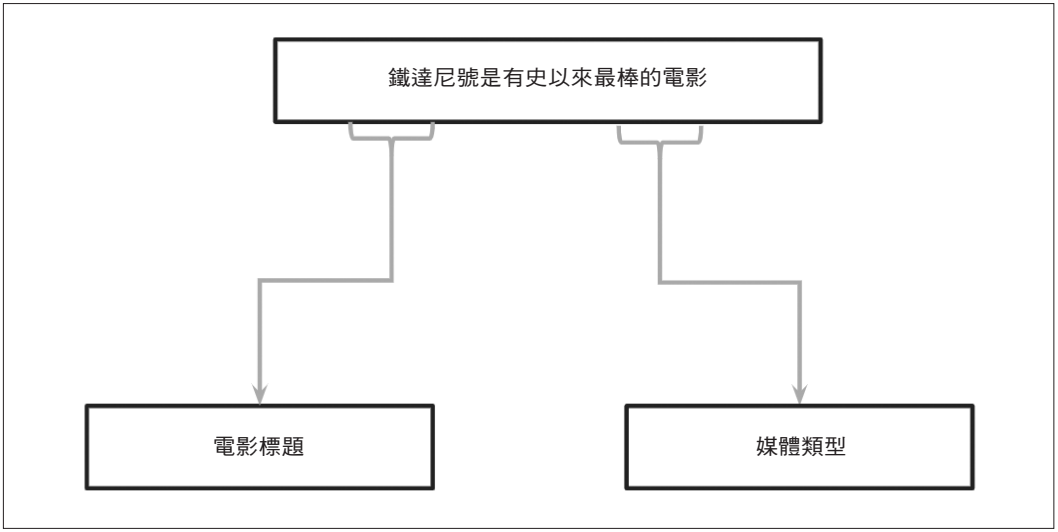

圖 1-4 從句子中提取媒體類型和標題

以圖片來說，知識提取任務通常包含尋找感興趣的區域（areas of interest）並分類它。圖 1-5 中描述了兩個常用的方法，第一個是物體偵測（object detection），這是一種較粗略的方法，它透過在感興趣的區域周圍繪製矩形（稱為**定界框**（*bounding box*））所組成；第二個是圖像分割（image segmentation），它可以更精確地在圖像中標籤每個像素的類別。

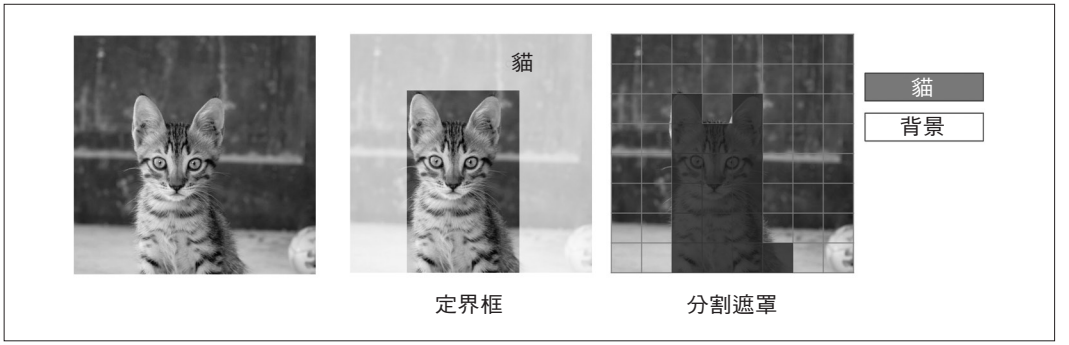

圖 1-5 定界框和分割遮罩（masks）

提取的資訊有時候還可當作另一個模型的輸入。例如：從瑜珈練習者的影片中使用第一個模型來偵測並提取姿勢的關鍵點，接著再將這些關鍵點輸入到第二個模型中，這個模型能基於關鍵點的標籤資料來分類姿勢是否正確。圖 1-6 顯示了能完成此任務的兩個連續模型範例，第一個模型從非結構化資料（照片）中提取結構化資訊（關節的座標），第二個模型獲取這些座標並分類成瑜珈姿勢。

目前為止，我們看到的模型都著重在特定輸入下產生輸出，但像是搜尋引擎或推薦系統，它們的產品目標則是顯示相關的項目，我們接下來將會介紹這個類型的模型。

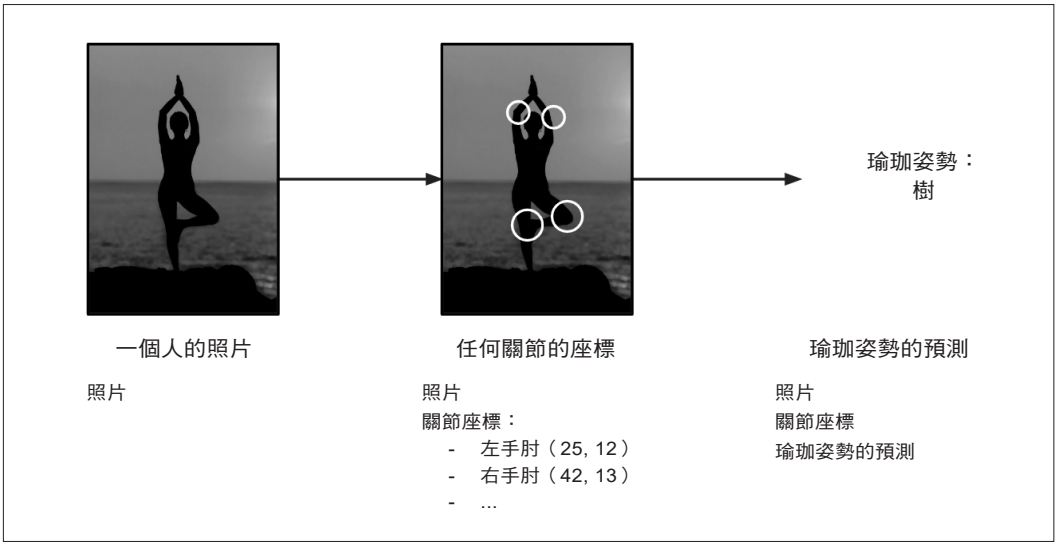

圖 1-6 瑜珈姿勢偵測

使用者目錄推薦

使用者目錄推薦模型通常會呈現給使用者一組結果，比如：以搜尋欄中輸入的文字、上傳的圖片，或是跟家庭助理說的一段話為條件所產生的結果。此外，在串流服務的案例中，還可以在使用者還沒提出要求時，就主動呈現他們可能會喜歡的內容。

圖 1-7 呈現了這種系統的範例，它根據使用者剛才看過的電影，主動推薦使用者可能喜歡的電影。

因此，這些模型根據使用者已表達興趣的項目來**推薦**其他項目（像 Medium 的文章或 Amazon 的商品），或是藉由目錄提供使用者有用的**搜尋**方式（允許使用者輸入文字或上傳他們自己的圖片來搜尋）。

這些推薦通常是從過去使用者的模式來學習，這種方法稱為**協同**（*collaborative*）推薦系統，推薦有時候是基於項目的特定屬性，這種方法稱為**基於內容**（*content-based*）的推薦系統。有些系統則同時利用協同和基於內容的方法。

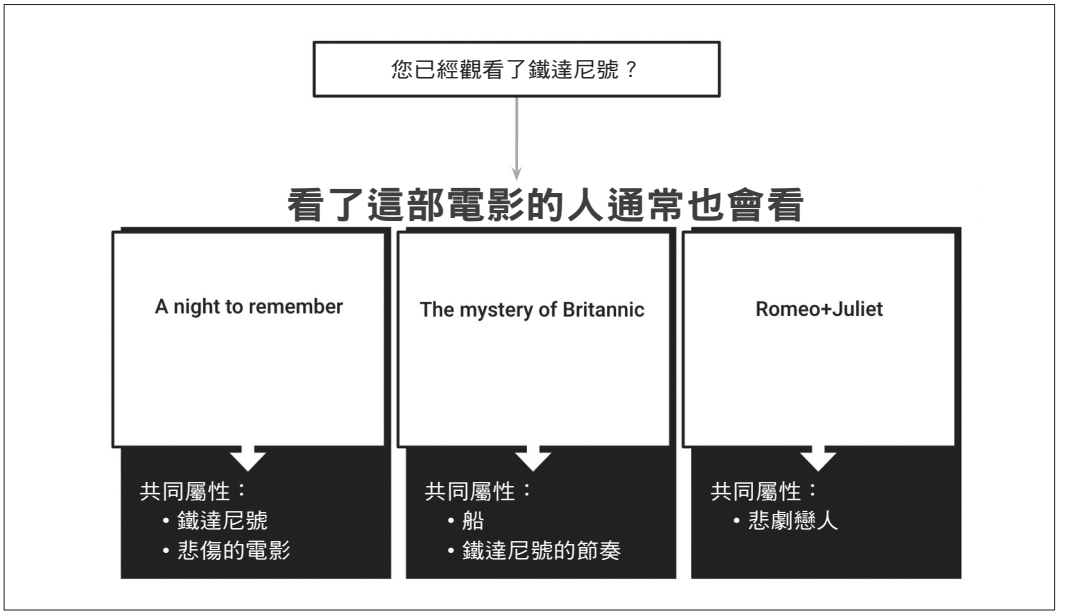

圖 1-7 電影推薦

最後，ML 也可以用於創意目的。 模型可以學習生成美觀的圖像、音訊甚至有趣的文字。 這樣的模型被稱為生成式模型。

生成式模型

生成式模型是根據使用者的輸入生成資料，而非分類、評分、提取或推薦資料。它們經常有範圍很大的輸出，這代表生成式模型是特定地去擬合任務，像是：翻譯，這些輸出的結果變化很大。

另外，生成式模型常用於訓練較少限制的輸出結果，這使它們成為生產環境中較高風險的模型。因此，除非它是達成目標的必要條件，否則我建議您從其他模型開始。但是，若您想要深入了解生成式模型，我推薦 David Foster 撰寫的書籍《*Generative Deep Learning*》。

具體的範例如：語句翻譯、文本摘要、將音軌對應到影片的字幕生成，以及從圖片對應到特定風格的神經網路風格轉換（neural style transfer）（參見 Gatys 等人的「A Neural Algorithm of Artistic Style」）（*https://oreil.ly/XVwMs*）

圖 1-8 是一個生成式模型的例子，透過給予中間下方小插圖的風格來轉換左側的照片，成為風格相仿的圖片。

圖 1-8 風格轉換的範例，源自 Gatys 等人的「A Neural Algorithm of Artistic Style」（https://oreil.ly/XVwMs）

您現在已經知道各類模型需要以不同類型的資料進行訓練，所以模型的選擇通常會受到您能夠取得的資料影響，即——資料可用性（data availability）經常驅動模型選擇。

接下來，讓我們介紹一些常見的資料使用情境和相關的模型吧。

資料

監督式 ML 模型利用資料中的模式來學習輸入對輸出的有用映射，如果資料集裡包含對目標具預測性的特徵，我們應該就能以合適的模型進行學習。然而，如果要訓練深度學習的端對端（end-to-end）模型，一開始我們通常沒有正確的資料可使用。

假設我們要訓練一個聽取客戶要求、理解客戶意圖，並依此執行的**語音辨識**（*speech recognition*）系統，當我們開始從事這個專案時，我們可能會定義一組我們想得知的客戶意圖，如「在電視上播放電影」。

為了訓練出能完成此任務的 ML 模型，我們需要聲音片段的資料集，它包含不同背景使用者以各自的用詞所提出播放電影的要求。因為任何模型都只能從我們提供給它的資料中學習，所以擁有一組具代表性的輸入相當重要，如果資料集只包含到總體的子集合，那麼產品只會對這個子集合有用。然而，如果是專業領域的應用，這類的資料集就不太可能已經存在了。

對於大部分的應用，我們需要去搜尋、整理並收集額外的資料，資料取得的流程和複雜度可能會因專案的具體情況而有很大的差異。所以為了應用的成功，事先評估挑戰是非常重要的。

首先，我們定義一些在搜尋資料集時的不同情況，這是決定後續如何進行的關鍵因素。

資料類型

將問題定義為**輸入對輸出的映射**後，我們能依此映射搜尋資料來源。

對於詐欺偵測，我們需要的資料可能是詐欺和合法的使用者案例，以及我們能用來預測他們行為的帳戶特徵。另一方面，對於翻譯來說，則需要成對的原始語句和目標領域語句的語料庫。至於在內容配置和推薦的應用中，需要的是搜尋和點擊的歷史資料。

我們很少能找到和預期完全一致的的資料，因此可以多考慮幾種不同的資料情況，我們把這視為不同層次的資料需求。

資料可用性

資料可用性從最理想到最具挑戰性的情境可以粗略分為三個等級。然而，您通常可以預設最有用的資料是最難找到的，讓我們來接著介紹吧。

已標籤的資料

這是圖 1-9 中最左側的類別。當使用監督式模型時，擁有**已標籤的資料集**是每個從業者的夢想。已標籤指的是有提供模型學習目標值的資料點，由於標籤代表了真實答案，這使得訓練並判斷模型品質變得更加容易。事實上，在網路上很少能找到符合自己需求又能自由使用的已標籤資料集，通常會把找到的資料集誤以為是自己需要的。

弱標籤的資料

這是圖 1-9 中的中間類別。有些資料集包含了不完全、但與建模目標又有某種程度相關的標籤。例如：在音樂串流服務中使用重播和跳轉的歷史紀錄，來預測使用者喜不喜歡這首歌。雖然聽眾並沒有把歌曲標註為不喜歡，不過如果他們播放時跳過了這首歌，那就表示他們有可能不喜歡它。雖然弱標籤在定義上較不精確，但通常比完全與建模目標一致的完美標籤更容易找到。

無標籤的資料

這是圖 1-9 中的右側類別。在某些情況下，我們雖然沒有獲得想要的已標籤資料集，但至少能夠獲取僅包含相關例子的無標籤資料集。以文本翻譯為例，我們或許能同時獲取包含兩種語言的大量文本資料集，然而資料集裡面並沒有輸入對輸出的直接映射。這代表我們需要為資料集上標籤，或尋找能學習這種無標籤資料的模型，或是以上兩種做法都各做一些。

我們需要獲得資料

無標籤資料集離我們無資料的情況只有一步之遙，所以只要去取得資料即可。多數情況下，因為我們沒有所需的資料集，所以需要找到能獲得所需資料的方法。一般人認為獲取資料是困難的任務，但其實現在已經有很多快速收集並標籤資料的方法，這在第 4 章會說明。

對於我們的 ML 寫作輔助編輯器，理想資料集應該是一組使用者輸入的問題以及一組用詞更好的問題，許多使用者問題的資料集有一些**弱標籤**來表示它們的品質，例如：「讚」（likes）或「贊同」（upvotes），所以這是屬於弱標籤資料集。這會幫助模型學習到使問題被判斷成好或壞的原因，但無法提供相同問題的編修版本。圖 1-9 中可以看到這些範例。

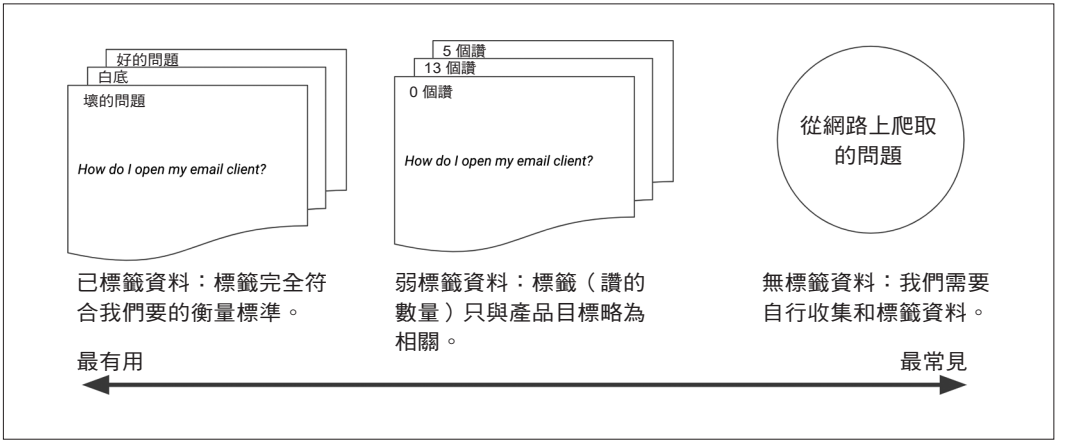

圖 1-9 資料可用性與資料有用性

在 ML 中，弱標籤資料集指的是包含有助於模型學習的相關標籤，但這些標籤並不完全貼近於真實的建模目標。然而，實際上我們所能收集到的資料集大多是弱標籤。

擁有不完美標籤的資料集是完全沒有問題的，而且這不應是阻止您前進的理由。ML 的流程本質上是疊代的，所以不論資料的品質如何，最好的前進方式是從資料集開始並獲得一些初步結果。

資料集是疊代的

由於您通常無法立即找到輸入直接對應到理想輸出的資料集，所以我建議逐步疊代您描述應用問題的方式，以便更輕鬆地找到合適的初始資料集。您探索並使用過的每個資料集都能提供有意義的資訊，使您可以用它們來選擇下一個版本的資料集，並為您的模型產生有用的特徵。

現在，讓我們深入案例研究，看看如何運用我們學到的知識來分辨可用的不同模型和資料集，並從中建立最合適的 ML 架構。

建構 ML 寫作輔助編輯器

接下來，我們來看看如何疊代產品使用案例以找出正確的 ML 建構方式，我們會概述從產品目標（幫助使用者撰寫出更好的問題）到建構 ML 模式的整個過程。

我們希望建立一個能接受使用者的問題，並幫助他們把問題寫得更好的寫作輔助編輯器。但在這個案例中，「更好」是什麼意思？讓我們先定義出更明確的產品目標。

許多人使用論壇、社群網站，例如 Stack Overflow（*https://stackoverflow.com/*）之類的網站來找到問題的答案。然而，問問題的方式與是否能收到有用的答案有很大的影響。問題如果表達不好，不論對於希望獲得問題答案的使用者，或未來可能有相同問題而想找到既存有用答案的使用者，都可能會導致不幸的情況。因此，我們的目標是**打造一個可以幫助使用者撰寫出更好問題的助手**。

我們已經有了產品目標，現在需要決定要使用哪種建模方法。為了做出決定，我們將會完整說明前面提過的模型選擇和資料驗證的疊代循環。

試著用 ML 做到這一切：端對端架構

在這個情境下，**端對端**指的是不需中間步驟的單一模型，即可從輸入直接對應到輸出。由於大多數的產品目標都是非常具體的，所以如果要嘗試以端對端來學習解決整個使用案例，通常會需要客製化前沿的 ML 模型。對於擁有足夠資源來開發並維護這種模型的團隊來說，這可能是正確的解決方案，但是我們通常還是先從更好理解的模型開始。

在我們的案例研究中，我們可以試著收集表達不好的問題資料集，以及問題的專業編修版本。然後，我們就可以使用端對端的生成式模型，將表達不好的問題直接轉換到另一個表達較好的問題。

圖 1-10 描述了這種轉換的過程。它是一個簡單的圖，左側是使用者的輸入，右側是預期的輸出，中間是模型。

圖 1-10 端對端方法

如您所見，這種方法將面臨巨大挑戰：

資料

為了獲得這樣的資料集，我們需要找到問題的意圖相同、但用詞品質不同的成對問題。這是相當難找到的資料集，而且因為我們需要專業編修人員的協助來產生這些資料，所以自己建置它的成本也很高。

模型

以上討論的端對端生成式模型類型中，從一個文字序列轉換到另一個文字序列的模型近年來有極大的發展。序列對序列（sequence-to-sequence）模型（如 I. Sutskever 等人在論文「Sequence to Sequence Learning with Neural Networks」（*https://arxiv.org/abs/1409.3215*）中所述）最初在 2014 年提出來應用在翻譯任務中，並且逐漸拉近了機器翻譯和人工翻譯之間的差距。然而，這些模型目前主要是在句子級（sentence-level）的任務上成功。它們不常用於翻譯比段落還長的文本，因為到目前為止，它們還無法捕捉到段落與段落之間的較長脈絡。另外，因為這種模型一般

來說具有大量的參數，所以它們是一種訓練最慢的模型。如果模型只訓練一次那不成問題，但如果需要每小時或每天再進行一次訓練，則訓練時間就成為重要的考量。

延遲（*Latency*）

序列對序列模型通常是**自迴歸模型**（*autoregressive model*），這代表它們需要接收到前一個單詞的模型輸出之後，才能再繼續處理下一個單詞。這樣雖然讓它們可以利用到鄰近單詞之間的資訊，但與簡單的模型相比，它們的訓練和推論速度較慢。與一秒內延遲的簡單模型相比，這類模型在推論時可能需要花費好幾秒才能得出答案，儘管可以最佳化這種模型讓它能更快地運行，但這會需要進行額外的工程。

實作的簡易程度

訓練複雜的端對端模型，其過程非常精巧而且容易出錯，因為它們具有許多可更動的部分，這代表我們需要權衡模型可能的效能和它在管線中增加的複雜度，這種複雜度將使我們在建構管線時的速度降低，同時帶來維護的負擔。如果我們預期到其他團隊成員可能需要疊代並改進您的模型，也許值得選擇一組更簡單也更容易理解的模型。

這種端對端的方法可能行得通但卻無法保證成功，而且需要在前期進行大量的資料收集與資料工程的工作。因此有必要探索其他替代方案，這部分我們將在後面介紹。

最簡單的方法：從演算法的角度

正如您將在本節中最後的訪談看到，對資料科學家來說，在實作演算法之前最好先**從演算法的角度**。換句話說，要了解如何很好地將應用問題自動化，請先試著手動解決問題。因此，如果我們自己能編修問題來提高可讀性和獲得答案的機率，那麼我們會如何做？

第一種方法是完全不使用資料，而是利用現有的技術來定義什麼使問題或文本內容寫得好。關於一般的寫作技巧，我們可以聯繫專業編輯或是去研究報紙的風格指南以了解更多資訊。

此外，我們應該深入研究資料集以了解個別實例和趨勢，並讓這些成為我們建模策略的參考，現在我們將暫時跳過此步驟，因為我們會在第 4 章中更深入地介紹怎麼做。

首先，我們可以去了解現有的研究（*https://oreil.ly/jspYn*），以辨別出一些我們可以用來幫助人們寫得更清晰的特徵，包含以下這些特徵：

語句簡單

我們經常建議寫作新手使用更簡單的單詞和語句結構。所以，我們能對合適的單詞和句子長度建立一套標準，並提供一些修改建議。

語氣

我們可以透過計算副詞、最高級和標點符號的使用，來評估文本的語氣傾向。根據問題的內容，太自以為是的問題可能會收到比較少的答案。

語句的結構特徵

最後，我們可以嘗試提取問題中的重要結構特徵，例如：問候語或問號的使用。

當我們辨識並生成了有用的特徵，就能開始建立一個簡單的解決方案。然後，使用它們來提供建議，這裡還沒有使用到 ML，但在這個階段有兩個理由使它非常重要：第一是它提供了能快速實作的基線（baseline），第二是它能當作評估模型的標準。

為了驗證我們對於如何檢測良好寫作的直觀想法，我們可以收集「好」和「壞」問題的資料集，以了解能否使用這些特徵區分出好和壞。

介於兩者中間：從我們的經驗中學習

由於上一部分，現在我們有了一個基線特徵組合，接著我們就可以試著使用它們建立**從文本內容學習出寫作風格的模型**。為此，我們可以收集資料並從中提取我們先前描述的特徵，接著用它們來訓練分類器來辨別好的問題和壞的問題。

當有了能對文本進行分類的模型，我們就可以檢查模型，以確定哪些特徵具有較高的預測性，並將那些特徵當作寫作的建議。我們會在第 7 章中實際了解如何來做。

圖 1-11 說明了這個方法。在左側，訓練好的模型把問題分類成好或壞；在右側，我們給訓練好的模型一個問題，模型會對修改後的問題進行評分，並推薦使用者評比最高分的問題表達方式。

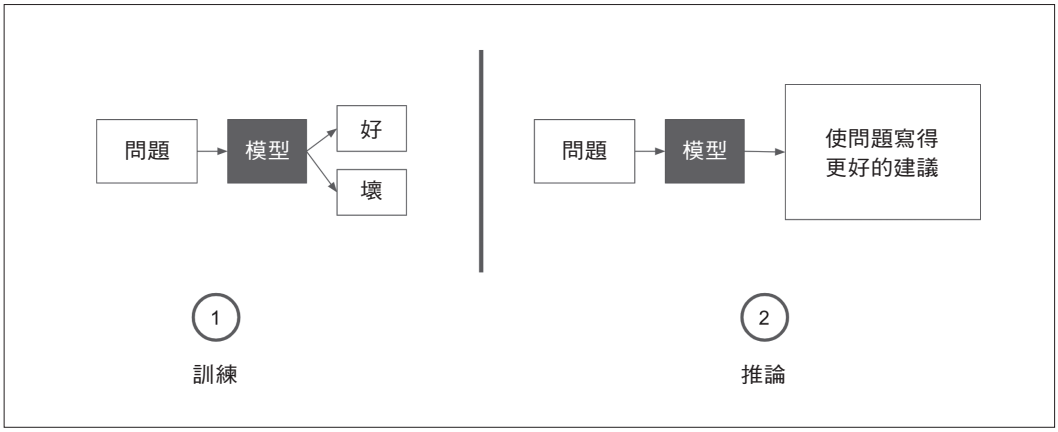

圖 1-11 介於手動和端對端的中間

讓我們檢視一下我們在第 17 頁「試著用 ML 做到這一切：端對端架構」中描述到的挑戰，看看分類器的方法是否有使挑戰變得更容易：

資料集

透過線上論壇收集問題並使用衡量品質的方式（例如：觀看次數或贊同數），我們可以獲得好和壞問題的資料集。與端對端方法不同，這種方法不要求我們獲取同一個問題的修訂版，我們只需要一組好和壞問題就能進行學習，而且這是比較容易找到的資料集。

模型

關於模型我們需要考慮兩件事情：第一是模型的預測能力如何（它能有效區分好的問題和壞的問題嗎？）；第二是從模型中提取特徵的容易程度（我們看得出哪個特徵能有效分類例子嗎？）。我們有多種可能的模型能使用，並從文本中提取出各種特徵，讓模型的預測更具解釋性。

延遲

大多數文本分類器都非常快，我們可以從一個簡單的模型（例如：隨機森林）開始，它可以在一般硬體上以不到 0.1 秒的速度回傳結果，並且在有需要時轉成更複雜的結構。

實作的簡易程度

文本分類相對於文本生成式模型更容易理解，這代表建立文本分類的模型應該比較快。網路上有許多文本分類管線的可行範例，而且許多模型已經部署到生產環境中了。

如果我們先從啟發式方法開始建立這個簡單的模型，我們很快就能擁有一個初始基線以及解決方案的第一步。此外，初始模型能告訴我們下一步要做什麼（第三部分會有更多的介紹）。

更多關於從簡單基線開始的重要性，我與 Monica Rogati 坐下來對談，她分享了在幫助資料團隊交付產品的過程中，所學到的一些經驗和教訓。

Monica Rogati：如何選擇並安排 ML 專案中的優先次序

Monica Rogati 取得電腦科學領域的博士學位後，她在 LinkedIn 開始了她的職業生涯，她從事核心產品的研發，例如：將 ML 導入「您可能認識的人」（People You May Know）演算法中，並建立公司職位與求職候選人配對的初版。接著，她成為了 Jawbone 的資料副總經理，然後創立並領導了整個資料團隊。Monica 現在是數十家公司的顧問，這些公司的員工人數從 5 人到 8,000 人不等。她大方地分享當 ML 產品在設計和執行時，她經常會給團隊的一些建議。

問：*您如何衡量一個 ML 產品？*

答：您必須確定您正在嘗試使用最適合的工具來解決問題，並且只有在有意義的情況下才使用 ML。

假設您想預測某個應用中的使用者想要做什麼動作，並推薦我們的預測。在此之前，您應該先整合關於建模和產品的討論，這包括在產品設計中如何優雅地處理 ML 的故障。

您可以從模型預測的信心度開始，根據信心度的分數來提出不同的建議，例如：如果信心度高於 90%，我們會強調這則建議；如果超過 50%，我們仍會顯示這則建議，但不會特別去強調它；如果信心度低於 50%，則不會顯示任何建議。

問：**您如何確定 *ML* 專案中的重點？**

答：您必須先找到 ML 專案中的**影響力瓶頸**，這代表如果您改善管線中的這個部分，產品將能產生最大的價值。

與一些公司合作時，我經常發現他們沒有著手於正確的問題上，或是未處於正確的成長階段。問題常常出現在模型外圍的功能，而找出這些問題的最好方法就是用簡單的東西替換掉模型，再為整個管線除錯。問題通常不會與模型的準確度有關，所以即使您的模型成功，您的產品往往也可能會無法運作。

問：**為什麼您建議從一個簡單的模型開始？**

答：我們計畫的目標應該是以某種方式降低模型的風險，而最好的方法是從基線開始評估最差情況下的效能，以我們之前的範例來說，可能只是單純建議使用者他曾經執行過的動作。

如果這樣做，我們的預測會多久正確一次？如果我們做錯了，我們的模型會對使用者造成多大的困擾？假設我們的模型沒有比基線好多少，我們的產品是否仍然有價值？

簡單的模型也適用於自然語言理解和生成的應用，例如：聊天機器人、翻譯、問答和文本摘要。例如，在文本摘要中，通常只要提取出一篇文章中前幾個最重要的關鍵字和類別，就足以滿足大多數使用者的需求。

問：**當建立好整個管線，您如何找出產品影響力的瓶頸？**

答：您先想像解決產品影響力瓶頸後的情況，然後評估一下，看這是否值得您投入心力來達成。我鼓勵資料科學家撰寫社群貼文、鼓勵公司在開始專案之前就撰寫新聞稿，並在內文中提到努力的成果和影響，這可以幫助他們避免進行一些不切實際的工作。

理想的情況是，無論結果如何，您都可以推銷到產品。即使您沒有獲得最佳的成果，產品是否仍具影響力？您從中學到了什麼？或是驗證到了什麼假設嗎？在此過程中，建立基礎設施能幫助您降低部署產品所需的工作量。

在 LinkedIn，我們可以取得一個非常有用的設計元件：帶有幾行文字和網址的小視窗，這讓我們能客製化產品。當設計被批准之後，我們就可以更輕鬆地為專案（例如：推薦工作）啟動實驗。因為投入資源較低，所以影響不會太大，還可以加快疊代週期。這樣一來，產品的障礙就變成了無關工程的問題，像是：道德、公平和品牌塑造。

問：您如何決定要使用哪種建模技術？

答：第一道防線是親自檢查資料內容，假設我們要建立一個向 LinkedIn 使用者推薦社團的模型。有一個天真的方法是：推薦社團標題中包含使用者公司名稱的最熱門社團。看了幾個例子之後，我們發現甲骨文公司最受歡迎的社團之一是「甲骨文很爛！」，如果真的向甲骨文員工推薦這個社團會很糟糕。

人工檢查模型的輸入輸出是有價值的，檢查看看裡面是否有任何異樣。我在 IBM 部門的負責人有個口頭禪是：在做任何工作之前，都要先進行一個小時的手動操作。

檢查資料可以幫助您構思良好的啟發式方法與模型，以及重新打造產品的方法。如果您依照出現頻率為資料進行排序，您甚至可以快速辨識並標籤 80%的使用案例。

例如：在 Jawbone，人們輸入「短語」來記錄他們吃了什麼，經過我們手動標籤前 100 名之後，我們就已經涵蓋了 80%的短語，而且對於要處理的主要問題（例如：各種文本編碼和語言）已經有了深刻的想法。

請多元化的員工來檢查建模的結果是我們最後一道防線，這樣一來，您就可以捕捉到模型表現出歧視性行為的例子，例如：把您的朋友標記為大猩猩；或者自認聰明地回顧「去年的這個時候」，卻沒有意識到這是使用者痛苦的經歷，而將它呈現出來。

總結

如我們所見，打造一個 ML 驅動的應用首先要判斷可行性並選擇一種方法。選擇一個監督式方法通常是最簡單的開始方式，包含：分類、知識提取、目錄配置或生成式模型都是最常見的範例。

在選擇一種方法前，您應該先確定是否易於取得強標籤或弱標籤的資料，或是任何其他類型的資料。接著，為了比較可能會使用的模型和資料集，您應該要定義產品目標並選擇最能達成目標的建模方法。

我們說明了 ML 寫作輔助編輯器的以上步驟，從簡單的啟發式方法和基於分類的方法開始。最後，我們介紹了 Monica Rogati 這樣的領導者如何去應用這些方法，並將 ML 模型成功地提供給使用者。

我們已經選擇了一個初步的方法，現在該是定義成功指標、創造行動計畫以取得定期進展的時候了，包含制定最低效能要求、深入研究可用的建模和資料的資源，以及建立簡單的原型。

我們將在第 2 章中介紹這些內容。