前言

網路比較像社群的產物，而不只是一種技術。

我設計網路的目的，主要是為了社群的效應（協助人們合作），而不只是把它當成技術性的玩物。Web 網路最終的目標，就是支援與改善世界上任何與網路有關的體驗。我們每個人總是身處於家庭、各種組織和公司之中。如今我們有可能更相信千里之外的人，而不見得相信近在咫尺的人。

— Tim Berners-Lee，《*Weaving the Web*》
（一千零一網：網際網路 *WWW* 發明人的思想構圖；*Harper* 出版社）

來自作者 Matthew Russell 的提醒

自從我最後一次處理《社群網站的資料探勘》（*Mining the Social Web*；第二版）的文稿至今，已經過了五年多，這段期間發生了很多變化。我在生活中學習到許多新事物，各種技術也以驚人的速度不斷發展，而各種社群網站也成熟到了一定程度，如今政府已經在制定各種有關如何收集、分享與使用資料的法律政策。

雖然我知道自己的時間安排，很可能無法滿足新版本製作、內容更新與擴展的各種需求，但是我完全相信，再也沒有比現在更好的時刻，更適合本書所要傳達的訊息，因此我覺得現在正是找一位共同作者合作的最佳時機，這樣才有機會讓本書順利傳達給對於挖掘社群網站充滿好奇的下一波企業家、技術人員和電腦駭客。我花了一年多才找到一位共同作者，對這個主題抱有共同的熱情，而且擁有寫書所需的技能與決心。

越視覺化函式庫典型應用的冒險嘗試，甚至構建出最先進的技術，這些全都不在本書的範圍之內。如果你是為了達成其中一個目標而購買本書，最後恐怕會很失望。不過，本書確實提供許多回答問題所需的基本要素，並且提供了一個跳板，這很可能正是你構建殺手級應用或進行研究時所需的要件。建議你先略讀幾章，再自己做個判斷。本書內容確實涵蓋許多的基礎。

特別重要一定要注意的是，API 總會不斷變化。社群媒體才剛出現沒多久，甚至連當今看來最成熟的平台，也都還在適應大家的使用習慣，而且面臨各種安全與隱私的新威脅。因此，我們的程式碼與各平台 API 之間的界面，也有可能隨時發生變化；這也就表示，本書所提供的範例程式碼，將來很有可能無法持續正常運作。我們已經盡可能針對一般用途與應用程式開發者，建立一些實用的範例，其中有些範例還是必須先把應用程式提交給官方，以供審核與批准。我們會盡可能加上有用的註釋，不過請別忘了，API 服務條款隨時都有可能改變。雖然如此，但只要你的應用程式遵守服務條款，應該就能獲得官方批准。你的努力絕對是值得的。

以 Python 為中心的技術

本書所有範例程式碼，全都刻意採用 Python 程式語言。Python 本身直觀的語法及其相應的套件體系，再加上 JSON（*http://bit.ly/1a1kFaF*）核心資料結構，讓各種 API 存取與資料處理變得非常簡單，因此它成為了一種出色的教學工具，不但功能強大，而且非常容易安裝和運行。如果這樣還不足以讓 Python 成為挖掘社群網站教學實務上絕佳的選擇，可以再搭配 Jupyter Notebook（*http://bit.ly/2LOhGvt*）這款功能強大的交互式程式解譯器，它可以在你的 Web 瀏覽器中提供絕佳的使用者體驗，讓你把程式碼執行、輸出、文字說明、數學排版、圖表等全都結合起來。在學習環境方面，已經很難想像還有其他更好的使用者體驗，因為它把範例程式碼許多方面的問題變得更簡單，讀者可以隨心所欲理解並執行程式碼，而不會遇到什麼麻煩。圖 1 顯示的是 Jupyter Notebook 的控制面板（*dashboard*）的畫面，本書每一章相應的 Jupyter Notebook 都會用到這樣的界面。圖 2 顯示的則是打開一個 Notebook 檔案之後相應的畫面。

些使用者。舉例來說，只要連往 *https://twitter.com/timoreilly/following* 就可以看到 Tim O'Reilly 登入 Twitter 之後，他的首頁時間軸所看到的推文內容。

像 TweetDeck 這類的應用程式，還提供了一些可自定義的視圖，讓你可以更容易檢視各種混亂的推文（如圖 1-1 所示），如果你還在使用 Twitter.com 傳統使用者界面，建議你可以嘗試一下這種不同的存取方式。

時間軸是更新速度相對較低的推文集合，而「串流」（streams）則是在 Twitter 上以即時方式出現的公開推文。大家都知道，在總統辯論或大型體育賽事這類特別受到廣泛關注的活動期間，公開的 firehose 所有推文經常可達到每分鐘數十萬條推文的峰值（*http://bit.ly/2xenpnR*）。Twitter 的公開 firehose 所發出的資料實在太多，其相關處理已超出本書範圍，但它確實是個有趣的工程挑戰，而讓消費大眾更容易接受的許多 firehose 大量資料處理方式，也是各大第三方商業供應商與 Twitter 合作的主要原因之一。另外一種做法，也可以只取公開時間軸的少量隨機樣本（*http://bit.ly/2p7G8hf7*），這樣就可以讓 API 開發人員透過篩選的方式，取得足夠多的公開資料，以開發出功能強大的應用。

圖 1-1：TweetDeck 提供各種可高度客製化的使用者界面，有助於分析 Twitter 上正在發生的事，並展示 Twitter API 可存取的各類資料

- 以程式碼的方式存取公開網頁（例如一些品牌和名人的頁面）的動態訊息（feeds）
- 提取出幾個社群相關的關鍵數據（例如「讚」、留言與分享的數量），藉此衡量使用者的參與度
- 使用 pandas DataFrames 來處理資料，然後再以視覺化方式呈現結果

2.2 探索 Facebook 的 Graph API

從深度與廣度來說，Facebook 平台絕對是個成熟、穩固且文件充分的一個網路門戶大站（gateway），它可以帶領我們通往有史以來最全面、組織最完整的訊息資料庫。其廣泛之處在於其使用者群，大約佔了全球人口的四分之一，而且針對任何特定的使用者，所瞭解的訊息量也很深入。Twitter 的特點是「非對稱的」關係模型，其模型是開放的，而且是以「無需特定許可，即可跟隨其他使用者」為前提；Facebook 的關係模型則是「對稱的」，使用者之間必須雙方同意，才能看到彼此的互動情況與活動。

此外，Twitter 除了使用者之間的私人訊息以外，幾乎所有互動都是公開的推文，但 Facebook 可以進行更細緻的隱私控制，可以組織朋友關係並用列表來進行維護，而且可以針對任何特定活動，為朋友提供不同等級的可見度。舉例來說，你可以選擇只與特定列表裡的朋友（而不是整個社群網路）分享連結或照片。

身為一個社群網站挖掘者，你要從 Facebook 提取資料的唯一方法，就是註冊一個應用程式，然後用這個應用程式做為 Facebook 開發者平台的一個入口點。此外，應用程式唯一可取得的資料，就是使用者明確授權可存取的任何資料。舉例來說，當你在編寫 Facebook 應用程式時，你就是登入該應用程式的使用者，而這個應用程式此時就可以存取到所有你已經明確獲得授權可存取的任何資料。從這個角度來看，做為 Facebook 使用者的你，也可以把你的應用程式想像成是你的某個 Facebook 朋友，而你完全可以控制該應用程式能夠存取到什麼樣的內容，也可以隨時撤銷存取權限。Facebook 平台政策（Facebook Platform Policy；*http://bit.ly/1a1lm3C*）是所有 Facebook 開發者都必須閱讀的文件，因為它針對所有 Facebook 使用者，提供了全面性的權利義務相關說明，並為 Facebook 開發者提供了相關的法律精神和法律條文。如果你還沒看過的話，值得花點時間重新檢視一下 Facebook 的開發者政策，並把 Facebook Developers 主頁（*http://bit.ly/1a1lm3Q*）加入書籤，因為這是進入 Facebook 平台及其相關文件的重要入口點。另外請記住，API 隨時都有可能發生變化。基於安全性和隱私的考量，你在嘗試使用

Facebook 平台時，可以使用的權限一定會受到某些限制。如果想要存取某些功能和 API 端點（endpoints），你可能都必須提交你的應用程式（*http://bit.ly/2vDb2B1*）以供審核並獲得批准。不過只要你的應用程式遵守服務條款（*http://on.fb.me/1a1lMXM*），應該就沒問題了。

如果你只是以開發者身份挖掘自己帳號裡的資料，應該不會有什麼問題，你完全可以讓應用程式自由存取自己帳號裡的所有資料。不過請注意，如果想開發出一個成功的「託管」（*hosted*）應用程式，應用程式就應該只針對完成任務所需的最小資料量提出請求，因為使用者很有可能並不信任你的應用程式，去做出一些超出授權所允許的動作（而且這樣才是正確的）。

雖然我們會在本章稍後透過程式碼存取 Facebook 平台，但 Facebook 其實提供了許多有用的開發者工具（*http://bit.ly/1a1lnVf*），其中包括 Graph API Explorer 應用程式（*http://bit.ly/2jd5Xdq*），我們會先使用這個應用程式（*http://bit.ly/2jd5Xdq*）來初步熟悉一下社群圖譜。這個應用程式針對社群圖譜提供了一種直觀、便捷的查詢方式，而且你如果對社群圖譜的運作方式感到滿意，就可以把相關查詢直接轉換成 Python 程式碼以達到自動化的效果，進一步處理也會變得很自然而容易。雖然我們在討論過程中就會運用到 Graph API，不過一開始先看一下 Graph API 總覽（Graph API Overview；*http://bit.ly/1a1lobU*）這份寫得很好的文件，先建立一個全面性的認識，對你應該很有幫助才對。

請注意，目前 Facebook 已終止對 Facebook 查詢語言（FQL，Facebook Query Language；*http://bit.ly/1a1lmRd*）的支援。從 2016 年 8 月 8 日開始，FQL 已無法再進行查詢了。開發者必須改用 Graph API。如果你之前曾用 FQL 開發過應用程式，Facebook 也提供了一個 API 升級工具（*http://bit.ly/2MGU7Z9*），可用來更新你的應用程式。

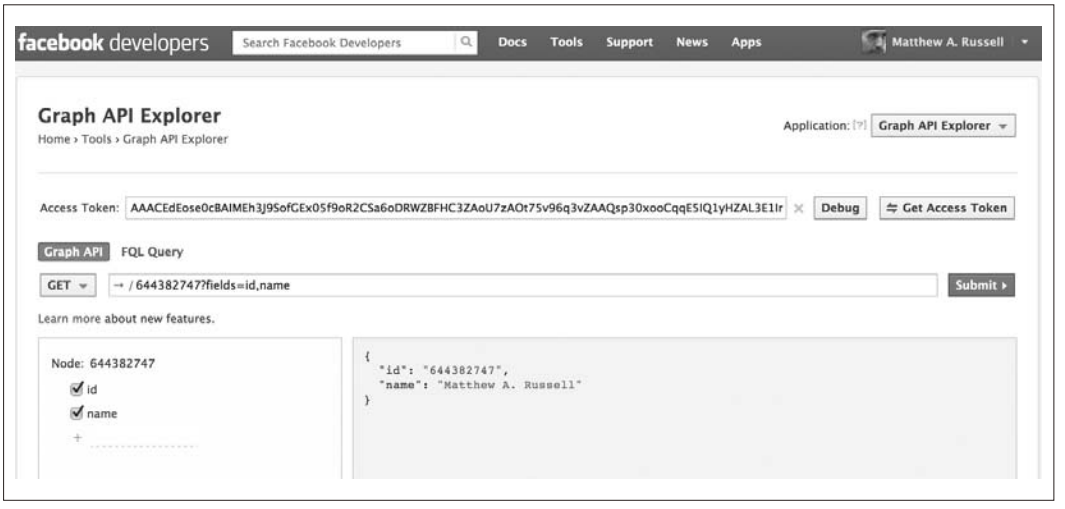

圖 2-1：運用 Graph API Explorer 應用程式，查詢社群圖譜中的節點

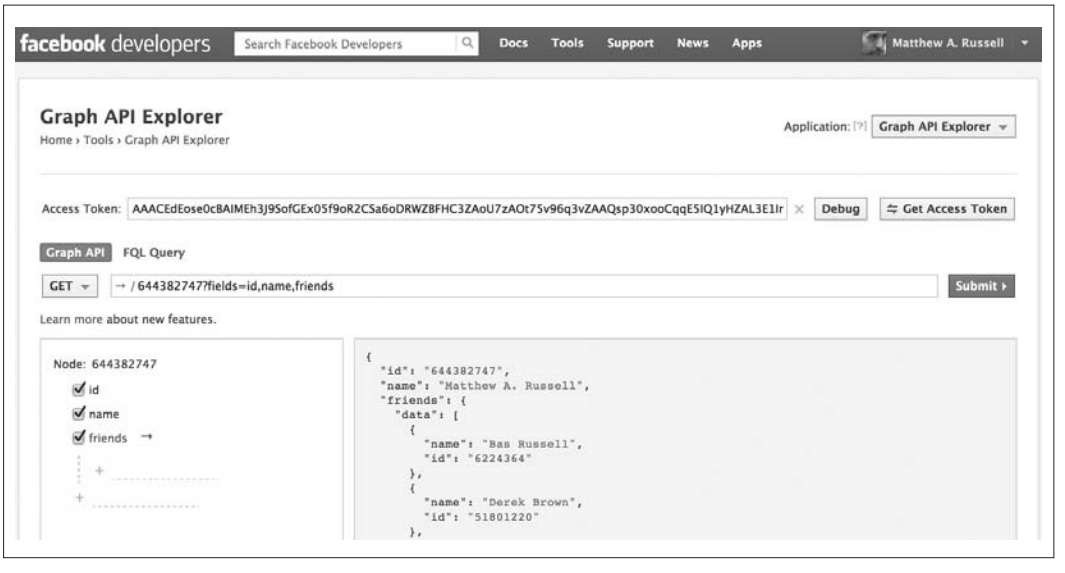

圖 2-2：運用 Graph API Explorer 應用程式，逐步建立節點與朋友連結的查詢。別忘了有些特定的資料可能需要先取得權限才能進行存取，而沙盒化的應用程式通常只能存取到非常有限的資料。這幾年 Facebook 針對資料政策，進行了許多調整。

圖 2-4：電影《絕地任務（The Rock）》的 IMDB 頁面，實現了 OGP 的概念

早在 2013 年，Facebook 就實作了一個名為 Facebook Graph Search 的語義搜尋引擎。這是一種可以讓使用者直接在搜尋欄中用自然語言進行查詢的做法。舉例來說，你只要在搜尋欄中輸入「住在倫敦而且喜歡貓的朋友」，這樣就可以從住在倫敦的朋友、以及喜歡貓的朋友之中，找出兩群朋友之間的交集。但是這種查詢 Facebook 的新方法，並沒有持續存在很長一段時間。2014 年底，Facebook 就放棄了這種語義搜尋功能，轉而採用以關鍵字為基礎的做法。儘管如此，可以把 Facebook 上的物件連結到其他東西的 graph 圖，目前依然存在。2017 年 11 月，Facebook 推出了一個名為 Facebook Local 的獨立移動應用程式，這個應用程式會以某個實際地點為中心，把圍繞該地點的事件連結起來，並使它產生社群化的效果。舉例來說，這個應用程式會告訴你，哪些餐廳在你 Facebook 的朋友之間最受歡迎。此應用程式之所以擁有這樣的能力，其背後的技術幾乎可以肯定就是 Facebook 的社群圖譜。

在往下進行 Graph API 查詢之前，我們先簡單看一下實作 OGP 的要點。OGP 文件中的標準範例，示範了如何使用如下的命名空間，把《絕地任務》的 IMDb 頁面轉換成 XHTML 文件，使它成為 OGP 開放圖譜協定中的一個物件：

```
<html xmlns:og="http://ogp.me/ns#">
<head>
<title>The Rock (1996)</title>
<meta property="og:title" content="The Rock" />
<meta property="og:type" content="movie" />
```

我們根據頁面 ID，用迴圈檢視每個藝人的頁面，取得藝人的姓名、粉絲數量，並從頁面動態訊息中取出最新的 10 篇貼文。然後再用一個內部的 for 迴圈逐一檢視這 10 篇貼文，並計算讚、分享和留言的總數量，然後再計算出這些數字佔所有粉絲總數量的百分比。這些訊息全都寫在 DataFrame 的一行（row）之中。只要把 ignore_index 關鍵字設為 True，我們就可以讓每一行不帶有預先決定的索引，而 pandas 則會採用枚舉（enumerate）的方式，處理所添加的每一行資料。

這個範例的最後一個迴圈，修改了其中幾個欄位資料的型別，好讓 pandas 知道這些數值資料全都是整數，而不是浮點數或其他的格式。

執行這段程式碼可能需要一些時間，因為資料是透過 API 從 Facebook 整合過來的，不過最後我們應該會得到一個很不錯的表格。pandas DataFrame 定義了一個很方便的 method 方法 .head()，可以讓你預覽一下表格的前五行（請參見圖 2-6）。我們強烈建議你可以在 Jupyter Notebook（*http://bit.ly/2omIqdG*）中，執行各種探索性質的資料分析工作（例如在本書隨附的 GitHub 儲存庫（*http://bit.ly/Mining-the-Social-Web-3E*）中所提供的那些程式碼）。

	藝人名	粉絲總人數	貼文編號	貼文日期	貼文標題	點讚數量	分享數量	留言數量	相對點讚比例	相對分享比例	相對留言比例
0	泰勒絲	73862332	1	2017-12-19T17:07:33+0000	Check out a key moment in Taylor writing "This...	33134	1994	1373	0.044859	0.002700	0.001859
1	泰勒絲	73862332	2	2017-12-17T16:42:38+0000		8282	19	353	0.011213	0.000026	0.000478
2	泰勒絲	73862332	3	2017-12-17T03:51:04+0000		11083	8	383	0.015004	0.000011	0.000519
3	泰勒絲	73862332	4	2017-12-16T20:19:52+0000	The Swift Life is available for free worldwide...	39237	925	1012	0.053122	0.001252	0.001370
4	泰勒絲	73862332	5	2017-12-15T13:18:45+0000	#TheSwiftLife App is available NOW for free in...	60721	1895	2105	0.082210	0.002566	0.002850

圖 2-6：「musicians」這個 pandas DataFrame 的前五行資料

pandas DataFrames 也定義了一些繪圖函式，這些函式可以讓你快速完成視覺化呈現資料的工作。這些函式會在後端調用 `matplotlib`，因此請務必確定已安裝該函式庫。

從範例 2-10 可以看到，pandas 具有一種很不錯的索引功能。我們使用的是 musicians 這個 DataFrame，然後把 Name 這個欄位的值為 Drake 的幾行資料，用索引的方式取出來。這樣我們就可以只針對「Drake」（德瑞克）相關的資料（而不必針對「泰勒絲」或「碧昂絲」相應的那幾行資料）做出進一步的操作。

範例 *2-14*：每位藝人每則貼文被點讚的相對比例長條圖

```
# 針對每位藝人的粉絲在最新 10 則貼文中所表現出來的參與程度，畫出各自相應的長條圖
plot = musicians.unstack(level=0)['Rel. Likes'].plot(kind='bar', subplots=False,
                                                     figsize=(10,5), width=0.8)
plot.set_xlabel('10 Latest Posts')
plot.set_ylabel('Likes / Total Fans (%)')
```

最後所得到的長條圖，就顯示在圖 2-11。

我們可以注意到，雖然與碧昂絲或泰勒絲相較之下，德瑞克在 Facebook 上的粉絲數量少得多，但在他的 Facebook 頁面中許多的貼文，都會吸引到比較高比例的粉絲。雖然這樣還不足以做出定論，不過這有可能表示，德瑞克的粉絲人數雖然比較少，但他們在 Facebook 似乎顯得比較熱情、更加活躍。另一方面這也有可能表示，德瑞克（或他的社群媒體經理）非常擅長在 Facebook 發佈高度吸引人的內容。

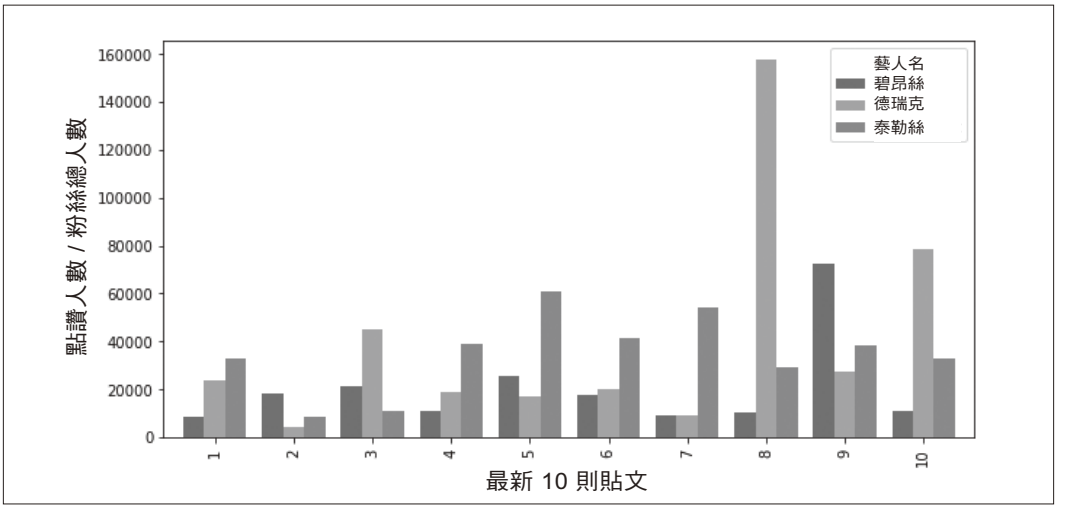

圖 2-11：每位藝人最新 10 則貼文被點讚的相對比例

如果要進行更深入的分析，可能還要考慮每則貼文的內容、內容類型（文字、圖片或影片）、留言所使用的語言，以及所收到的其他回應（除了讚之外）。

3.4.3 使用預訓練神經網路，對照片進行物體識別

各種神經網路（尤其是具有非常多複雜隱藏層的「深度學習」系統）正在改變整個產業。如今一般人所認為的「人工智慧」，通常大多（不過並不侷限於）被用來指那些深度學習系統的各種應用方式，包括機器翻譯、電腦視覺、自然語言理解、自然語言生成等等。

如果你想進一步瞭解人工神經網路的工作原理，Michael Nielsen 的書《Neural Networks and Deep Learning》（神經網路和深度學習；*http://bit.ly/2IzOycW*）是一個很好的資源，這本書從線上就可以取得。另外也可以參閱「Machine Learning for Artists」（針對藝術家的機器學習；*http://bit.ly/2rChK9i*）其中的內容，尤其是標題為「Looking Inside Neural Networks」（深入神經網路；*http://bit.ly/2Kimo3c*）的章節。

想要建立、訓練一個可用於物體識別的人工神經網路，並不是件容易的事。首先必須選擇正確的神經網路架構，然後必須整理出大量預先標記好的圖片來訓練網路，再挑選出正確的超參數，最後還要讓網路訓練足夠長的時間，才能提供準確的結果。幸運的是，已經有人完成這項工作，而且這項技術已被妥善包裝起來，現在任何人都可以使用到這些最先進的電腦視覺 API 了。

我們會使用的是 Google Cloud Vision API，這個工具可以讓開發者使用 Google 強大的預訓練神經網路，來分析各式各樣的圖片。它可以偵測出圖片中的各種物體、偵測出臉部、提取出文字、標註出明顯的內容，還有一些其他的功能。

各位可以到 Cloud Vision API（*http://bit.ly/2IEmOny*）頁面，然後向下滾動到「Try the API（嘗試 API）」。如果你還沒有帳號，則需要在 Google Cloud Platform 上進行註冊。在撰寫本文的當下，Google 提供了一個免費套餐（*http://bit.ly/2wCXKbC*），在 12 個月內有 $ 300 美元的免費額度可供運用。註冊時你還是必須填寫信用卡訊息，不過只是做為識別之用。如果你沒有信用卡，填寫銀行帳號也可以。

接下來，你必須在 Google Cloud Platform 建立一個專案（create a project；*http://bit.ly/2rE0Zul*）。你可以把這些專案想像成你在 Instagram 開發者平台上所建立的客戶端程式。

只要進入「Cloud Resource Manager」（雲端資源管理器；*http://bit.ly/2wyV5zD*）頁面，就可以建立專案。請馬上就去建立一個，然後幫它取個你自己喜歡的名字（例如「MTSW」）。接著把它連結（Attach）到你的「帳單帳號」（*billing account*）。就算你並沒有使用 Google 的免費套餐，前 1000 次的 Cloud Vision API 調用也是免費的。

```
        print ('[{0:3.0f}%]: {1}'.format(label['score']*100, label['description']))

    return

# 最後針對貓咪的圖片，調用圖片標籤相關函式
label_image(cat)
```

與 Vision API 之間互動的方式可能有點棘手，因此範例 3-10 的程式碼把物體偵測與標記圖片所需的全部工具全都納入，其中圖片有可能儲存在本地的硬碟中，也有可能透過網路進行存取。

只要執行範例 3-10 的程式碼，就會開啟一個圖片檔案，這是一張貓咪的照片（請參見圖 3-8）。讀取了圖片檔案之後，圖片資料就會透過 API 傳遞給 Google。在 `service_request` 這個變數中，我們載入了一個「執行圖片標記」的請求。

圖 3-8：雪地上的貓咪照片（Von.grzanka（*https://bit.ly/2obUbly*）所提供的圖片，CC BY-SA 3.0（*https://bit.ly/1pawxfE*）或 GFDL（*http://2wlkjool*），取自 *Wikimedia Commons*）

表 5-4：TF-IDF 範例查詢所牽涉到的計算（根據範例 5-3 的計算）

文件	tf 術語頻率（mr.）	tf 術語頻率（green）	tf 術語頻率（the）	tf 術語頻率（plant）
語料庫 corpus['a']	0.1053	0.1053	0.1053	0
語料庫 corpus['b']	0	0.1111	0	0.1111
語料庫 corpus['c']	0	0.0625	0	0.0625

idf 逆文件頻率（mr.）	idf 逆文件頻率（green）	idf 逆文件頻率（the）	idf 逆文件頻率（plant）
2.0986	1.0	2.099	1.4055

	tf-idf（mr.）	tf-idf（green）	tf-idf（the）	tf-idf（plant）
語料庫 corpus['a']	0.1053*2.0986 = 0.2209	0.1053*1.0 = 0.1053	0.1053*2.099 = 0.2209	0*1.4055 = 0
語料庫 corpus['b']	0*2.0986 = 0	0.1111*1.0 = 0.1111	0*2.099 = 0	0.1111*1.4055 = 0.1562
語料庫 corpus['c']	0*2.0986 = 0	0.0625*1.0 = 0.0625	0*2.099 = 0	0.0625*1.4055 = 0.0878

每一個查詢相應的結果就顯示在表 5-5，其中 TF-IDF 的值是針對每個文件相加而來。

表 5-5：查詢範例相應加總後的 TF-IDF 值，這些值都是透過範例 5-3 計算而得的（粗體值代表的就是三個查詢所對應的最高分數）

查詢	語料庫 corpus['a']	語料庫 corpus['b']	語料庫 corpus['c']
green	0.1053	**0.1111**	0.0625
Mr. Green	0.2209 + 0.1053 = **0.3262**	0 + 0.1111 = 0.1111	0 + 0.0625 = 0.0625
the green plant	0.2209 + 0.1053 + 0 = **0.3262**	0 + 0.1111 + 0.1562 = 0.2673	0 + 0.0625 + 0.0878 = 0.1503

如果從定性的角度來看，這樣的查詢結果相當合理。查詢「green」時，勝利者是語料庫 `corpus['b']` 文件，而語料庫 `corpus['a']` 則只以分毫之差落敗。在這個例子中，關鍵因素在於語料庫 `corpus['b']` 的長度比語料庫 `corpus['a']` 的長度短了許多，因此就算「green」在語料庫 `corpus['a']` 中出現了兩次，但歸一化 TF 分數還是讓「green」只出現一次的語料庫 `corpus['b']` 勝出。由於在三個文件中都有出現「green」這個單詞，因此 IDF 這個項在計算中發揮了主要的效果。

不過要注意的是，在某些 IDF 的實作中，如果看到「green」送回來的是 0.0 而不是 1.0，由於計算過程中 TF 會與這個 0.0 相乘，因此「green」這個單詞針對這三個文件所