

自然語言處理



本章中，我們來看看，要如何利用以自然語言來表達的大量知識。

人類(學名 *homo sapiens*，意為有智慧之人)因其具有語言的能力，使得他在物種間顯得與眾不同。100,000 年前的某處，人類已經學習到如何說話，然後大約 7,000 年前人類已經學習到如何書寫。儘管黑猩猩、海豚以及其他一些哺乳動物也表現出包含數百個符號的辭彙，但是只有人類能夠在任何的主題上使用離散符號，確實地傳達不限量的不同本質的訊息。

當然，人類還有其他一些獨一無二的屬性：沒有其他物種會穿衣服、能夠創作藝術，或者是一天看三個小時的電視。但是當 Turing 提出他的測試時(參見第 1.1.1 節)，他將之建立在語言的基礎上，而非藝術或者電視。我們之所以為想要電腦代理人具有處理自然語言的能力的兩個主要原因有第一，可以與人類溝通，我們將在第 23 章中繼續這個主題。第二，從書寫出來的語言中獲取訊息，這是本章的重點。

在網路上有幾兆個資料頁面，幾乎都以自然語言來表達。代理人若想做**知識獲取**必須要能夠懂得(至少部分懂得)人類使用的充滿歧義且難以處理的語言。我們檢驗這個問題會從特定的資訊尋找任務的觀點來看：文本分類、資訊檢索和資訊擷取。在敘述一些任務的一個共同因數是**語言模型**的使用：預測語言表達的機率分佈的模型。

22.1 語言模型

正規語言，就如 Java 或 Python 程式語言一樣，有精確定義的語言模型。**語言**可以被定義為字串的集合；「`print(2 + 2)`」在 Python 語言裡是合法的程式，然而「`2)+(2 print`」就不是了。既然存在有無限數量的合法程式，它們就不是可列舉的；取而代之的是它們以規則的集合來被定義，稱做**文法**。正規語言也有定義程式含義或**語意**的規則；舉例來說，該些規則陳述「`2 + 2`」的「含義」是 4，而「`1/0`」的含義是錯誤提示。

自然語言如英文或西班牙文，並不能被描繪成是一套明確語句的集合。大家都能同意「Not to be invited is sad」是英文中的一個句子，但是人們可會不同意「To be not invited is sad」在是合乎文法的。因此將自然語言模型定義成語句的機率分佈，比起說它是一套明確定義集合來的有成效多了。也就是說，比較起詢問一個由詞語組成的字串是否為定義語言的一個項，我們不妨用 $P(S = Words)$ ——一個隨機語句為詞語的機率為何——來詢問。

自然語言也充滿歧義。「He saw her duck」可以代表了他看到屬於她的水禽或是他看到她逃避某些事情。所以，再次說明了我們無法說出一個語句所代表的單一意義，而是可能的意義的機率分佈。

最後，自然語言因其大量且不斷的改變而非常難以處理。因此，我們的自然語言模型充其量是一個近似形。我們會從最簡單的可能近似形開始討論起，並且以其為基礎討論下去。

22.1.1 N 元字元模型

最終來說，書面文字是由**字元**所組成——英文(或其他語言裡更多的異國字元)的字母、數字、標點符號和空白。因此，一個最簡單的語言模型就是字元序列上的機率分佈。如同第 15 章中所示，我們用 $P(c_{1:N})$ 來表示 N 字元的一串序列的機率，從 c_1 到 c_N 。在一個網路集合中， $P(\text{「the」}) = 0.027$ 和 $P(\text{「zgq」}) = 0.000000002$ 。長度為 n 的一串的書寫符號稱為一個 n 元(希臘字根，用以書寫或當作字母)，特殊例子有「一元(unigram)」代表 1 元，「二元(bigram)」代表 2 元，「三元(trigram)」代表 3 元。一個由 n 字母序列的機率分佈模型因此稱做 n 元模型(但是要注意到： n 元模型可以是組成在詞語序列、音節或其他一元上，而非僅於字元上)。

n 元模型定義為 $n-1$ 階的**馬可夫鍊**。回想 15.1.2 節所提到的，馬可夫鍊中字元 c_i 的機率僅參考其前一個字元，任何其他字元都不對其產生影響。於是在三元模型(2 階的馬可夫鍊)我們可得

$$P(c_i | c_{1:i-1}) = P(c_i | c_{i-2:i-1})$$

我們可用三元模型來定義字元序列的機率 $P(c_{1:N})$ ，根據連鎖律的第一因素然後再使用馬可夫假設：

$$P(c_{1:N}) = \prod_{i=1}^N P(c_i | c_{1:i-1}) = \prod_{i=1}^N P(c_i | c_{i-2:i-1})$$

對於三元字元模型來說一個有 100 個字元的語言， $\mathbf{P}(C_i | C_{i-2:i-1})$ 會有一百萬個條目，且可以透過計算一千萬或更多字元的文本體中所含之字元序列，精確地估計出來。我們將文本體稱做**語料庫(plural corpora)**，在拉丁語中這個字是身體的意思。

我們能夠利用 n 元字元模型做到什麼呢？其中一個很適合它們的任務是**語言辨識**：給定一段文字，判斷它是用哪種自然語言寫成的。相對來說這是一個簡單的任務，因為即使是一段簡短的文字如「Hello, world」或「Wie geht es dir」，它輕易地就能辨認出前者為英文與後者為德文。電腦系統能以大於 99% 的準確率來辨認語言；但偶爾，相近似的語言例如瑞典文與挪威文就會搞混了。

一種語言辨識的方法是為每個候選語言先建立一個三元字元模， $P(c_i | c_{i-2:i-1}, \ell)$ ，其中變數 ℓ 在各語言中變動。對於每個 ℓ 來說，模型的建立是透過在該語言的語料庫中計算三元(每個語言需要大約 100,000 個字)。這給了我們一個 $\mathbf{P}(\text{Text} | \text{Language})$ 的模型，但是我們想要選擇出最有可能的語言的文字，於是我們應用貝氏法則再接馬可夫假設來得到最有可能的語言：

$$\begin{aligned} \ell^* &= \operatorname{argmax}_{\ell} P(\ell | c_{1:N}) \\ &= \operatorname{argmax}_{\ell} P(\ell) P(c_{1:N} | \ell) \\ &= \operatorname{argmax}_{\ell} P(\ell) \prod_{i=1}^N P(c_i | c_{i-2:i-1}, \ell) \end{aligned}$$

三元模型可以透過語料庫來學得，那事前機率 $P(\ell)$ 呢？我們也許已經有這些值的一些估計；舉例來說假設我們選擇一個隨機網路頁面，我們知道最有可能的語言是英文，且它是馬其頓語的機率將會低於 1%。我們為這些事前(priors)所選擇的確切數量並非是關鍵，因為三元模型通常選擇一種語言會比起其它的更甚好幾階的大小。

其它字元模型的任務包括拼寫錯誤更正、流派分類與專有名詞辨識。流派分類意指決定一段文字是新聞故事、法律文件、科學文章等。雖然有很多特徵有助於此分類，標點符號的計數與其它字元的 n 元特徵性仍走了一段很長的路(Kessler 等人, 1997)。專有名詞辨識是在文件中找尋事物名稱，且決定其屬於何種分類的任務。例如，一段文字「Mr. Sopersteen 開了 aciphex 處方」(Mr. Sopersteen was prescribed aciphex)中我們應該要辨識出「Mr. Sopersteen」是人的名字且「aciphex」是藥品的名稱。字元層級模型對於這個任務來說是有利的，因為它們可以將字元序列「ex┐」(「ex」後接著空白)與藥品名稱連結，「steen┐」與人的名字連結，並且因此而判斷出這些之前從未發現的字詞。

22.1.2 平滑 n 元模型

n 元模型的主要複雜性在於語料庫的訓練僅提供真實機率分佈的估計。對於一般的字元序列如「┐th」，任何英文語料庫都可以有一個很好的估計：所有三元項中大約 1.5%。另一方面來說，「┐ht」很罕見——字典中沒有以 ht 為開頭的字。有可能在訓練過的標準英語語料庫中這個序列的計數為零。這代表了我們應該假設 $P(\text{「┐th」}) = 0$ 嗎？如果我們做這樣假設，那麼這段文字「The program issues an http request」為英語的機率將會是零，這看起來並不正確。對於一般化我們遇到了問題：我們希望我們的語言模型對於沒有見過的文字可以將之一般化得宜。僅因我們之前未遇過「┐http」並不能代表我們的模型於是宣稱這段文字是不可能的。因此，我們會將我們的語言模型做些調整使得被訓練過的語料庫中，計數為零的序列會被賦予很小的非零機率(且其它的計數值也會被輕微的向下調整使得機率總和仍為 1)。調整低頻率的計數機率的程序稱做平滑。

最簡單的平滑型式是在 18 世紀由拉普拉斯(Pierre-Simon Laplace)所建議的：他指出在缺乏更進一步的資訊前，如果隨機布林變數 X 在所有目前為止的 n 個觀察中為假，那麼 $P(X = \text{true})$ 的估計應該為 $1/(n+2)$ 。也就是說，他假設在兩次以上的測試中，其一可能為真而另一為假。拉普拉斯平滑(也稱做加 1 平滑)是正確方向的往前一步，但其表現相對來說很差。較好的是**退回模型**，其中我們從估計 n 元計數開始，一旦遇到有低(或零)計數的特定序列我們就退回到 $(n-1)$ 元。**線性插值平滑(linear interpolation smoothing)**是一種退回模型，該方法用線性插值將三元模型、二元模型和一元模型結合起來。它將機率估計定義為：

$$\hat{P}(c_i | c_{i-2:i-1}) = \lambda_3 P(c_i | c_{i-2:i-1}) + \lambda_2 P(c_i | c_{i-1}) + \lambda_1 P(c_i)$$

這裡 $\lambda_3 + \lambda_2 + \lambda_1 = 1$ 。參數值 λ_i 可以是固定的，或者是由最大期望演算法(expectation-maximization algorithm)對其進行訓練。也有可能根據計數而得到 λ_i 的值：如果三元中有高的計數，我們就給予相對較高的權重；如果僅是低計數值，那麼我們就賦予二元與一元模型較高的權重。一個研究陣營已經發展出更加複雜的平滑模型，然而另外的陣營建議的是蒐集更大的語料庫來使簡單的平滑模型也可運作的很好。雙方都是為了達成同一個目的：減少語言模型中的變化幅度。

一種複雜化：注意到表達式 $P(c_i | c_{i-2:i-1})$ 會得到 $P(c_1 | c_{-1:0})$ ，當 $i = 1$ 。但 c_1 之前不存在任何字元。我們可以引入人工字元進來，例如定義 c_0 為空字元或者特殊的「起使文字(begin text)」字元。或者我們可以回頭使用低階的馬可夫模型，可達到定義 $c_{-1:0}$ 為空序列所以得 $P(c_1 | c_{-1:0}) = P(c_1)$ 。

22.1.3 評價模型

利用這麼多可能的 n 元模型——一元、二元、三元、不同 λ 值的插值平滑等——我們要怎麼挑選模型呢？我們可以使用交叉驗證來評價模型。把你的語料庫分割為一個訓練語料庫和一個驗證語料庫。根據訓練資料確定模型的參數。然後在驗證語料庫上評估模型。

評價可以是一個特定任務指標，如同語言辨識的正確性量測。另外，我們可以有一個無關任務模型的語言質量：然後，透過模型計算賦予測試語料庫的機率；機率越高越好。這個指標顯的不方便因大型語料庫的機率佔非常小的數量，且浮點下溢也造成了一個問題。另一個不同描述序列機率的方法是利用一種稱做**混亂度**的度量，定義如下：

$$\text{Perplexity}(c_{1:N}) = P(c_{1:N})^{-\frac{1}{N}}$$

混亂度可以被看做是由序列長度來做正規化的機率排序。它也可看成帶有權重的模型的平均分支因數。假設我們語言中有 100 個字元，且我們的模型假設這些字元都同等相似。於是對任意長度的序列來說，混亂度為 100。假如其中一些字元比起其它字元更為相似，且模型也反映出來，那麼這個模型的混亂度就低於 100。

22.1.4 N 元單字模型

現在我們從字元 n 元模型轉向單字。對於單字與字元模型方法的應用是相同的。主要的差異在於詞彙——組成語料庫與模型的符號集——較大。多數語言只含有 100 個字元，且有時我們建構字元模型時甚至是更為嚴格的，例如將「A」「a」看成是同樣的符號，或者將所有的標點看程式同樣的符號。但單字模型我們有至少數萬甚至上百萬的符號。這麼寬的範圍是因為一個單字的組成並不明確。在英語中由空白所區隔開之字母的排列稱為單字，但在某些語言裡如中文來說單字並非由空白所區隔，即使是在英文裡也需要對單字的邊界更明確的策略來判定其為一個單字：「ne'er-do-well」裡有幾個單字？或在「(Tel:1-800-960-5660x123)」中呢？

單字 n 元模型需要處理**詞彙以外的單字**。對字元模型來說，我們並不擔心誰又從字元表(alphabet)裡發明出一個新的字。但對於單字模型來說總有訓練語料庫中不曾見的單字出現的可能，所以我們需要對語言模型將之以明確地模型化。只是加入一個新單字到詞彙中就可達到：<UNK>，表示未知的單字。我們可以利用這個技巧估計<UNK>的 n 元計數值：搜尋訓練語料庫，且在第一次發現之前未知的單字時用<UNK>這個符號來取代。接著出現的全部單字保持不變。然後照常做語料庫的 n 元計數值的計算，對待<UNK>如所有其它單字一般。於是在測試集中出現一個為之單字時，我們以<UNK>來查閱其機率。有時會多個未知的單字符號用以不同的分類。舉例來說任意數字的字串以<NUM>來取代，或者用<EMAIL>來取代電子郵件地址。

要獲得單字模型能做到什麼的理解，本書中我們建立單字的一元、二元與三元模型並且對模型裡的單字序列做隨機取樣。結果是

一元：logical are as are confusion a may right tries agent goal the was...

二元：systems are very similar computational approach would be represented...

三元：planning and scheduling are integrated the success of naive bayes model is...

即使是這個小的樣本，很清楚看出一元模型對英語或者 AI 教科書內容兩者來看都是非常糟糕的近似，二元和三元好多了。這些模型支持下述評價：一元模型的混亂度為 891，二元模型為 142，三元模型為 91。

隨著 n 元模型——基於字元或者單字——基礎的建立，我們現在可以轉向一些語言任務了。

22.2 文本分類

我們要更入探討**文本分類**的任務，也被稱為**範疇化**：給訂某類型的文本，判斷出它屬於預先定義好的類別集合中的哪一類。語言辨識和流派分類即是文本分類的例子，就如情緒分析(判斷一部電影或產品評論是正面性或反面性的)與**垃圾郵件偵測**[將電子郵件訊息分類為垃圾郵件(spam)或非垃圾郵件(not-spam)]。既然「非垃圾郵件」是很冗長的，研究者創造**正常郵件**(ham)這個詞來代表非垃圾郵件。我們可以把垃圾郵件的偵測當成受監督學習中的一種問題。訓練集隨時就緒：正面性(垃圾郵件)例子放在我的垃圾郵件資料匣裡，負面性(ham)例子則放在我的收件匣。下面是一段摘錄：

Spam: Wholesale Fashion Watches -57% today. Designer watches for cheap...

Spam: You can buy ViagraFr\$1.85 All Medications at unbeatable prices!...

Spam: WE CAN TREAT ANYTHING YOU SUFFER FROM JUST TRUST US...

Spam: Sta.rt earn*ing the salary yo,u d-eserve by o'btaining the prope,r crede'ntials!

Ham: The practical significance of hypertree width in identifying more...

Ham: Abstract:We will motivate the problem of social identity clustering:...

Ham: Good to see you my friend.Hey Peter, It was good to hear from you....

Ham: PDS implies convexity of the resulting optimization problem (Kernel Ridge...

從這段摘錄我們可以開始蒐集到在哪些可能是好的特徵，可加進受監督的學習模型中。詞語的 n 元性「for cheap」與「You can buy」似乎是垃圾郵件的指標(儘管它們在正常郵件中的機率也不是零)。字元層級的特徵也似乎也很重要：垃圾郵件有更多的可能是帶有全部都是小寫的詞語，和一個詞語中夾雜了標點符號。顯然垃圾郵件發送者有想到二元詞語「you deserve」太過表現為垃圾郵件，因此改以「yo,u d-eserve」來代替。字元模型應該會偵測的出來這點。我們可以為垃圾郵件或正常郵件建立一個完全字元的 n 元模型，或者也可手動定義如「詞語中夾雜的標點符號數量」。

請注意到我們有兩個互補的方法來討論分類。在語言模型的方法中，我們透過垃圾郵件匣中的訓練來為 $P(\text{Message} | \text{spam})$ 定義一個 n 元語言模型，透過收件匣中的訓練來為 $P(\text{Message} | \text{ham})$ 定義模型。接著我們利用貝氏法則來分類新的郵件訊息：

$$\operatorname{argmax}_{c \in \{\text{spam}, \text{ham}\}} P(c | \text{message}) = \operatorname{argmax}_{c \in \{\text{spam}, \text{ham}\}} P(\text{message} | c) P(c)$$

$P(c)$ 是僅以計算垃圾郵件與正常郵件訊息總和來做估計。這個方法對於垃圾郵件偵測運作的很好，正如它在語言辨識所展現的。

在機器學習方法中我們將郵件訊息表示成特徵/數量對(feature/value pairs)的集合，並且對特徵向量 \mathbf{x} 採用分類演算法 h 。將 n 元想成是特徵，我們可以使得透過一元模型來看再容易不過了。特徵是詞彙中的詞語：「a」「aardvark」等，這些值是每個詞語在郵件訊息中出現的次數。這會使得特徵向量變得巨大且稀疏。如果在語言模型中有 100,000，那麼特徵向量的長度就是 100,000，幾乎所有的特徵在簡短郵件訊息來說計數都為零。這種一元表示(unigram representation)已被稱為**詞袋模型**。你可以把此模型想像成把訓練語料庫裡的詞語放在一個袋子中並且依序一次只從裡頭挑出一個出來。詞語順序的概念消失後，一元模型賦予一段文字的任意排列有相同的機率。更高階的 n 元模型則維持局部的詞語順序概念。

使用二元或三元特徵的數量是平方或三次方，而我們可以再增加一些非 n 元的特徵：訊息被傳送的時間點、訊息中是否包含 URL 或是圖片、訊息中寄件者的 ID 號碼、寄件者上一封垃圾郵件和正常郵件的號碼等等。挑選所要使用的特徵是產生好的垃圾郵件偵測器最重要的部分——比起選擇使用什麼演算法來處理這些特徵來的重要的多。某種程度上來說這是因為有相當多的訓練資料，因此如果我們可以提出一項特徵，這些資料就可以精確地確定此特徵的好壞。因為垃圾郵件偵測是一項**對抗性任務**，持續的更新特徵是必須要的；垃圾郵件發送者會針對垃圾郵件偵測器的改變來調整發送的垃圾郵件。

對於相當巨大的特徵向量運行演算法來計算的代價所費不貲，所以通常**特徵選擇**的程序在使用上，是用於維持那些區分出是否為垃圾郵件與否的特徵。舉例來說，二元組「of the」在英語中相當常見，可能在垃圾郵件與正常郵件中也一樣常見，所以對其計數是沒有意義的。通常來說大約最高的一百項特徵就可以區分出這兩個類型。

一旦我們選擇出一組特徵集合，我們可以應用所有我們所知的受監督學習技巧；文本分類的普遍技巧包括有 k -最近鄰演算法(k -nearest-neighbors)、支援向量機(support vector machines)、決策樹(decision trees)、樸素貝氏(naïve Bayes)與羅吉斯迴歸(logistic regression)。這些技巧已經都被應用到垃圾郵件偵測上，通常來說準確率範圍達 98%至 99%。使用精心設計的特徵集和，那麼準確率甚至可超過 99.9%。

22.2.1 透過資料壓縮來做分類

分類的另外一種看法是資料壓縮裡的問題。無失真壓縮演算法使用一串符號的序列，偵測其中重複的模式(pattern)然後用比原來更緊湊的序列來描述。舉個例子，「0.142857142857142857」這段文字可被壓縮為「0.[142857]*3」。壓縮演算法基於對一段文字先建立子序列的字典，然後再將文字與此字典中的項目產生對應。剛剛的例子中，字典只含一個項目即「142857」。

實際上，壓縮演算法會產生語言模型。LZW(Lempel-Ziv-Welch, LZW)演算法特別直接地將最大熵機率分配給模型化。要利用壓縮來做分類，所有的垃圾郵件訓練訊息合在一起然後將之當成同一個單元(unit)來做壓縮。對於正常郵件的作法也是一樣。之後當有一個新訊息要做分類時，先附加到垃圾郵件訊息並將結果做壓縮。這個新訊息同時也加入正常郵件訊息並做壓縮。無論那一個分類壓縮的好些——在該新訊息之後加入較少的額外幾個位元(bytes)——那就是預測中的分類。這個想法是一個垃圾郵件訊息會傾向於與其它垃圾郵件訊息共用字典項目，因而當將之附加到一個已含有垃圾郵件字典集合中會壓縮的較好。

利用壓縮分類方法在一些文本分類的標準語料庫——20-新聞群組(20-Newsgroups)資料集、路透社新聞-10(Reuters-10)語料庫、工業部門(Industry Sector)語料庫——上之實驗都指出當運行現成的(off-the-shelf)壓縮演算法如 gzip、RAR 和 LZW 可能會相當慢，其準確度可以比擬傳統的分類演算法。就其自身而言這相當有趣，同時也用以指出，對於直接使用 n 元字元而無文本預處理或特徵選擇的演算法，是有個承諾：它們似乎正在捕捉一些真實模式。

22.3 資訊檢索

資訊檢索(information retrieval)的任務是尋找與使用者需要的資訊相關的文件。資訊檢索系統的一個眾所周知的例子就是全球資訊網上的搜尋引擎。全球資訊網使用者將類似[AI book][AI 書籍]²的查詢資訊輸入到搜尋引擎，並得到相關網頁的列表。在本節中，我們將看到如何建立這樣的系統。一個資訊檢索(即 IR)系統有如下特徵：

1. **文件語料庫(a corpus of documents)**。每個系統都必須決定它要處理的文件是什麼：一個段落、一頁還是多頁的文本。
2. **以查詢語言提出的查詢(queries posed in query language)**。查詢指定出使用者想知道的內容。查詢語言可以是單字的列表，如 [AI book]；可以指定一個必須相鄰的單字組成的片語，如[「AI book」]；可以包含布耳運算符，如[AI AND book]；或者包括非布耳運算符，如[AI NEAR book]以及[AI book site: www.aaai.org]。
3. **結果集合(result set)**。該集合是 IR 系統判定與查詢**相關(relevant)**的那部分文件子集。「相關」的含義是對提出查詢的人有用，針對查詢中表達的特定資訊需求。
4. **結果集合的表示**。結果集合可簡單如一份文件標題的排序列表，或複雜如將結果集合投射至三維空間時的旋轉色彩映對圖(表示成二維)。

最早的 IR 系統在布林關鍵字模型(Boolean keyword model)的基礎上運轉。文件集中的每個單字都被當作一個布林特徵，如果這個單字出現在某文件中，那麼它對該文件的值為真，反之為假。所以特徵「檢索」對本章的值為真，對第十五章則為假。查詢語言是關於特徵的布林運算式的語言。只有運算式的取值為真時，文件與查詢才是相關的。例如，查詢[information AND retrieval]對本章的值為真，而對第十五章為假。

這個模型的優點在於容易解譯和實作。但是，它也存在一些缺點。首先，由於文件的相關度只用一個位元表示，所以無法指導如何對相關文件的表示進行排序。其次，不是程式設計人員和邏輯學家的使用者也許不熟悉邏輯運算式。使用者可以發現相當直觀的是，當想要知道在 Kansas 與 Nebraska 洲裡的種植知識，他們需要使用查詢[farming (Kansas OR Nebraska)]。第三，對一個適當的查詢進行形式化，即使是對熟練的使用者來說也可能是很困難的。假設我們試圖查詢[information AND retrieval AND models AND optimization]，得到了一個空的結果集合。那麼我們可能會嘗試查詢[information OR retrieval OR models OR optimization]，但是如果回傳很多的結果，我們將不知道下一步應該如何入手了。

22.3.1 IR 計分函數

大多數的 IR 系統已不使用布林(Boolean)模型而改使用基於詞語計數統計值的模型。我們會敘述 **BM25 計分函數**，其來源為倫敦城市學院(London's City College)的 Stephen Robertson 與 Karen Sparck Jones 的 Okapi 計畫，這已經被使用於如開源 Lucene 計畫(open-source Lucene project)等搜尋引擎。

計分函數採用一份文件與一個查詢並且回傳數字分數；最有價值的文件有最高的分數。在 BM25 函式中，分數是每一個詞語組成的查詢之線性權重分數組合。有三個因素會影響查詢詞的權重：第一，查詢詞在文件中出現的頻率(又稱為 *TF*, term frequency)。對查詢[farming in Kansas]來說，常常提及「farming」會有較高分數。第二，詞的文件反頻率，或稱 *IDF*。「in」這個字幾乎在每個文件都會出現，所以它有高的文件頻率，或低的反文件頻率，所以因此這個字並不如查詢「farming」或「Kansas」來的重要。第三，文件的長度。一份百萬字的文件裡頭非常可能提及了全部的搜尋詞語，但是也許跟查詢一點也沒相關。一份短的文件所提及的所有詞語會是較好的候選文件。

BM25 函式將以上三者都列入考慮。假設我們在語料庫中創造一個 N 份文件的索引讓我們可以查閱 $TF(q_i, d_j)$ ，即出現在文件 d_j 中的詞語 q_i 次數的計數。我們也假設了一份文件頻率計數表 $DF(q_i)$ ，代表含有詞語 q_i 的文件之數量。於是，給定文件 d_j 和一個包括 $q_{1:N}$ 個詞語的查詢，得到

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^N IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot (1 - b + b \cdot \frac{|d_j|}{L})}$$

其中， $|d_j|$ 詞語中文件 d_j 的長度， L 是在語料庫中文件的平均長度： $L = \sum_i |d_i| / N$ 。我們有兩個參數， k 和 b 可以利用交叉驗證來調整；典型的值為 $k = 2.0$ 與 $b = 0.75$ 。 $IDF(q_i)$ 是詞語 q_i 的反文件頻率，由下式所得

$$IDF(q_i) = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}$$

當然，將 BM25 計分函數應用到語料庫中的每份文件是很不實際的。反而，系統提前產生一個**索引**，對於每個字彙詞語列出包含此詞語的所有文件。這稱做詞語的**命中表**。於是給定一個查詢，我們將搜尋詞語的命中表產生交集，並僅對交集集中的文件評分。

22.3.2 IR 系統評價

我們如何知道一個 IR 系統的性能是否很好？我們可以做一個實驗，系統已知一個查詢集合，結果集合透過人進行相關性判斷得到評分。傳統上在評分中有兩種度量方法：召回率和準確率。我們將透過一個例子的幫助來解譯它們。想像某個 IR 系統已經對一個單獨的查詢回傳了一個結果集合，其中我們知道在由 100 篇文件組成的語料庫中哪些文件是相關的，哪些文件是無關的。每個類別內的文件計數結果如下表所示：

	在結果集合中	不在結果集合中
相關的	30	20
無關的	10	40

準確率(precision)度量結果集合中實際相關的文件所佔的比例。在我們的例子中，準確率是 $30/(30 + 10) = 0.75$ 。而誤判率是 $1 - 0.75 = 0.25$ 。**召回率**(recall)度量的是結果集合裡的相關文件在文件集合裡的所有相關文件中所佔的比例。在我們的例子中，召回率是 $30 / (30+20) = 0.60$ 。而漏報率是 $1 - 0.60 = 0.40$ 。在諸如全球資訊網之類的超大規模文件集合中，召回率是很難計算的，因為沒有簡單的方法檢查每個網頁的相關性。我們所能做的，就是透過取樣估計召回率，或是完全忽略召回率而只判斷準確率。在網頁搜尋引擎的情況裡，結果集合中可能有幾千份文件，所以對許多不同規模如「P@10」(前十個結果的準確率)或「P@50」度量準確率會比對整個結果集合評估準確率來的有意義多了。

根據調整回傳的結果集合規模在準確率和召回率之間折衷是可行的。在極端情況下，回傳文件集合中所有文件保證有 100% 的召回率，但是精確率很低。另一方面，如果一個系統只能回傳唯一的文件，會有很低的召回率，但是卻有很大的機會得到 100% 的精確率。兩種度量方法的總和稱為 F_1 分數，此為單一數字表示精確率和召回率的調和平均數，寫成 $2PR/(P + R)$ 。

22.3.3 IR 的改進方法

這邊描述了很多系統可能的改進方法。在網頁搜尋引擎發現新的方法或者隨著網頁成長與改變下，它們也的確是持續地在更新它們的演算法。

一個常見的改進是基於相關度，文件長度所會產生之影響的較佳模型。Singhal 等(1996)觀察到簡單的文件長度正規化機制傾向於過度偏愛短文件，而長文件則顯的不足。它們提出一個關鍵的文件長度正規化機制；這個想法表達的是某一文件長度運行在舊式正規化為正確時，這個長度是關鍵的；比該長度還短的文件會得到強化改進而比之長的文件會得到懲罰。

BM25 計分函數使用的詞語模型模型把所有詞語視為獨立，但是我們知道一些詞語是相關的：「couch」與「couches」和「sofa」關係都很密切。很多 IR 系統都試圖考慮到這些相關性。

例如，如果某個查詢是 [couch]，那麼如果從結果集合中排除那些提到「COUCH」或者「couches」而不是「couch」的文件則是不應該的。很多 IR 系統都要進行**大小寫合併(case folding)**，將「COUCH」轉換為「couch」，還有很多系統採用**詞幹化(stemming)**演算法把「couches」還原為其詞幹形式「couch」。典型情況下，這能稍微提高一點兒召回率(對英語大約是 2%)。然而，它對準確率卻有不良影響。例如，「stocking」取詞幹的結果是「stock」，雖然這能夠提高有關倉庫儲存的查詢的召回率，但是卻傾向於降低有關腳的覆蓋物(編註：「stocking」有長襪的意思)以及金融手段(編註：「stock」有股票的意思，不過是名詞；而「stocking」可能是動詞「儲備」的現在進行時)的查詢的精確率。基於規則(例如：去除「-ing」)的取詞幹演算法不能避免這個問題，但是基於詞典(如果這個單字已經列在詞典中，則不去除「-ing」)的一些較新的演算法則可以解決該問題。雖然詞幹化的方法在英語中影響不大，但是它在其他語言中能起重要作用。例如在德語中，類似於「Lebensversicherungsgesellschaftsangestellter」(人壽保險公司雇員)這樣的單字是不常見的。諸如芬蘭語、土耳其語、因紐特語以及愛斯基摩語等語言都具有遞迴的構詞規則，從而在原則上能夠產生無限長度的單字。

下一步是區別類似「sofa」與「couch」這樣的**同義詞(synonym)**。如同使用詞幹化方法，有少量提高召回率的潛力，但卻可造成精確率的傷害。使用者給定的查詢[Tim Couch]會想看到有關橄欖球而非沙發的結果。問題在於「語言憎恨絕對的同義詞，就像自然界憎恨真空一樣」(Cruse, 1986)。也就是說，無論何時當有兩個表示同一件事物的單字存在時，使用這種語言的人總會企圖更改它們的含義從而消除混淆。相關詞語而非同義詞在排序上也扮演了重要的角色——如「leather」、「wooden」或「modern」可幫助確認文件指的就是「couch」。同義詞與相關語可以在字典裡，或可在尋找文件或查詢的相關性時找到——假設我們發現很多使用者要求一個查詢[new sofa]接在查詢[new couch]之後，未來我們可將[new sofa]更改為[new sofa OR new couch]。

作為最終的改進，IR 能夠透過考慮**詮釋資料(metadata)**——文件文本之外的資料——以得到改進。詮釋資料的例子包括人提供的關鍵字以及公開的資料。在網頁中，文件間的超文本**連接**是資訊至關重要的來源。

22.3.4 頁排序演算法

頁排序(PageRank)^[3]為兩個原創想法之一，在 1997 年引入時，它使得 Google 的搜尋不同於其它網頁搜尋引擎(另一個創新是錨文本(anchor text)——超連結裡的底線文字——於頁面索引的使用，即便當錨文本所在頁面不同於被索引的頁面)。PageRank 被發明來解決 *TF* 分數的專橫問題：如果查詢是[IBM]，我們如何能保證 IBM 的首頁 ibm.com 是第一個查詢結果，即使有其它的頁面更頻繁地提及「IBM」？其想法為 ibm.com 有很多 in-links(到該頁的連結)，因而它應該被排名較高：每一個 in-link 都為連結至(linked-to)頁面的品質投下一票。但如果我們只計算 in-links，那麼垃圾網頁發送者也有可能創造一個很多頁面的網路並且將之全部指向他所選擇的頁面，藉此來增加他選擇之頁面的分數。

因此，頁排序演算法被設計成對高品質(high-quality)的站台(site)裡的連結有更高的權重。那什麼是高品質站台呢？即一個被其它高品質站台所連結的站台。這個定義是遞迴的，但我們可以看到遞迴如何適當地推演至最底端。頁面 p 的頁排行定義為：

$$PR(p) = \frac{1-d}{N} + d \sum_i \frac{PR(in_i)}{C(in_i)}$$

$PR(p)$ 表示頁面 p 的頁排行， N 是語料庫裡的總頁面數， in_i 是將 in 連結到 p 的頁面， $C(in_i)$ 是 in_i 頁面中對外連結(out-links)的所有數目的計數。常數 d 為阻尼係數。這可透過隨機衝浪模型(random surfer model)來瞭解：想像網路衝浪者(Web surfer)從某些隨機頁面開始探索。有機率 d (我們假設 $d = 0.85$) 衝浪者會點選一個頁面中的連結(公平地從它們之中擇一)，有 $1-d$ 的機率她對該頁面感到無聊不做點選並且從網路上任一隨機頁面重新開始。頁面 p 的頁排序於是成為隨機衝浪者在任何時間點處在的頁面 p 之機率。頁排序可以經由疊代程序計算而得：從任意頁面開始的 $PR(p) = 1$ ，重複演算法並更新排序直到排序收斂。

22.3.5 HITS 演算法

超連結引導主題搜索演算法(Hyperlink-Induced Topic Search algorithm)，亦稱「集中站台與權威(Hubs and Authorities)」或是 HITS，是另一具有影響力的連結分析(link-analysis)演算法(見圖 22.1)。HITS 在許多方面都不同於頁排序。首先，它是依賴查詢(query-dependent)度量：它參考查詢來評價頁面。這意味著每個查詢都必須重新計算——大多數搜尋引擎都不選擇使用的計算負擔(computational burden)。給定一個查詢，HITS 會先去找尋與這個查詢相關的頁面之集合。它使查詢詞語的命中表產生交集，然後加入這些網頁裡頭相鄰的鏈結頁面——從原始相關集合裡其中一個連結到或者連結出去的頁面。

```

function HITS(query) returns pages with hub and authority numbers

  pages ← EXPAND-PAGES(RELEVANT-PAGES(query))
  for each p in pages do
    p.AUTHORITY ← 1
    p.HUB ← 1
  repeat until convergence do
    for each p in pages do
      p.AUTHORITY ←  $\sum_i \text{INLINK}_i(p).$ HUB
      p.HUB ←  $\sum_i \text{OUTLINK}_i(p).$ AUTHORITY
    NORMALIZE(pages)
  return pages

```

圖 22.1 圖示為 HITS 演算法，應用在計算關於一個查詢的集中式站台數與影響力。相關頁面(RELEVANT-PAGES)提取出符合查詢的頁面，延伸頁面(EXPAND-PAGES)加入此頁面中連結出去的所有頁面或任何連到此頁面的所有相關頁面。正規化(NORMALIZE)將每個頁面的分數除以所有頁面分數的平方和(影響力與集中式站台分數分開計算)

此集合中的每一個頁面在搜尋時的**影響力**係其他相關集合中的頁面指向它的程度。頁面被認為是**集中站台**指向相關集合中的權威頁面的程度。就如 PageRank，我們不想僅是計算鍵結的數量；我們想做的是給予高品質的集中站台與影響力更高的評價。因此，正如 PageRank 所為，我們更新一個頁面的影響力分數，使其等於所指向頁面的集中式站台分數之總和，還有使集中式站台分數等於所指向頁面的影響力分數之總和，並反覆這個過程。假如我們接著將分數正規化並且重複 k 次，這個過程會達到收斂。

PageRank 與 HITS 在發展我們對網路資訊檢索(Web information retrieval)理解上扮演著很重要的角色。這些演算法與其延伸被應用在排序(ranking)每天數億次的查詢，然而搜尋引擎仍穩定地在發展更佳方法提取搜尋相關性的更精細之訊號。

22.3.6 問答

資訊檢索是找出相關於一個檢索的文件之任務，而查詢可能是一個問題，或者只是一個主題範圍或是概念。**問答**是一個有點不同的任務，此任務中查詢真的就是一個問題，而回答並非是有序文件，而更像是短回應——一個句子或者甚至是一個詞組。1960 年代以來已經有一些問答 NLP(自然語言處理，natural language processing)系統，但直到 2001 年才使用在網頁資訊檢索，從根本上增加其涵蓋之廣度。

ASKMS 系統(Banko 等，2002)就是一個傳統的網頁基礎問答系統。它係基於直觀上認為大多數的問題在網頁上都會被回答好幾次，所以問答應該被思考成精確的問題，而非召回率。我們不必去處理敘述一個問題可能的所有方向——我們僅需找出其中之一。舉例來說，考慮[Who killed Abraham Lincoln?]這個查詢。假設一個系統必須要回答這個問題僅能透過一部百科全書，此百科全書關於林肯的描述是

John Wilkes Booth 用一顆子彈改變了歷史。他永遠會被稱為結束 Abraham Lincoln 生命的那個人。

要透過這個通道來回答問題，系統必須要知道結束生命可能是殺害的意思，「他」則是指 Booth，還有幾個其他的語言學與語義的事實。

ASKMSR 不會嘗試這種類型的複雜性——它完全不曉得代名詞參照，或殺害，或任何其他動詞。它是曉得 15 個不同種類的问题，以及如何改寫這些問題成為搜尋引擎的查詢。它曉得[誰殺害 Abraham Lincoln]可以被寫成查詢[*killed Abraham Lincoln]以及[Abraham Lincoln was killed by*]。它發出這些改寫的查詢並且檢驗回傳的結果——並非全部的網頁，而是符合查詢語附近簡短的概要文字。結果被斷分為 1-元、2-元和 3-元 且吻合結果集合中的出現頻率與權重：一個從非常特定查詢改寫(如符合[「Abraham Lincoln was killed by*」]的精確詞組)回傳的 n -元所得到的權重會比一般查詢改寫來的多，如[Abraham OR Lincoln OR killed]。我們會期待「John Wilkes Booth」出現在檢索到的高度排名之 n -元中，但「Abraham Lincoln」、「被暗殺」與「Ford 劇院」也會同時出現。

一旦 n -元被評分，它們就會被預期的類型所過濾。如果原始查詢起始以「是誰」，我們會過濾出人名；以「多少個」我們就得到數量，以「幾時」我們就得到日期或時間。同時也會有一個過濾是指出出現答案並非是問題的一部分；綜合這些就能夠讓我們得出「John Wilkes Booth」(而非「Abraham Lincoln」)是最高分數的回應。

某些情況中答案會比三個詞語還長；既然組成回應最多是 3-元，更長的回應就必須是從較短的來拼湊而成。舉例來說，在僅使用 2-元的系統中，「John Wilkes Booth」這個答案可能是從高分數的「John Wilkes」與「Wilkes Booth」二者拼湊而成。

在信息檢索評測會議(Text Retrieval Evaluation Conference, TREC)中，ASKMSR 被分類為頂尖系統之一，擊敗競爭者做到更複雜的語言理解的能力。ASKMSR 憑藉在網路上內容的廣度更勝於其理解的深度。它無法處理複雜的推理模式如「誰殺害」與「結束某生命」的關聯性。但它能知道網頁是如此巨大以致於它可負擔的是忽略如上述複雜的通道，並等待它能夠掌握的簡單通道。

22.4 資訊擷取

資訊擷取(information extraction)是一個獲取知識的過程，其透過瀏覽文字，並尋找特定類別物件的出現以及這些物件之間的關係。我們可以嘗試從網頁中為包括街名、城市名、州名以及郵遞區號等在內的資料庫欄位擷取位址實例，或是從天氣預報中為包括溫度、風速以及降雨量等在內的資料庫欄位擷取暴風雨實例。在有限域中，這可非常高精確地達成。但一旦域變得更廣，更多複雜的語言學模型與更複雜的學習技巧就有需要了。我們在第 23 章中會看到如何定義英文中，短語結構(名詞詞組與動詞詞組)的複雜語言模型。但到目前為止仍無此類完整模型，對目前的資訊擷取有限之需要，我們定義出有限模型來近似完整的英語模型，並僅致力手邊所需要用到的部份。本節中我們所描述的近似型方法，如同圖 7.21 中之簡單 1-CNF 模型為完整、扭曲、邏輯模型之近似型。

本節中我們敘述六種不同的資訊擷取方法，以供不同維度上增加之複雜性：確定性到隨機性，特定域到一般域，手工到學習，與小範疇到大範疇。

22.4.1 資訊擷取的有限狀態機

最簡單類型的資訊擷取系統被稱為**基於屬性的擷取系統**，因為它假設全部文本談的是單一物件，而系統的任務就是擷取該物件的屬性。舉例來說，我們注意到第 12.7 節中提取文字「IBM ThinkBook」的問題。我們的價錢：\$399.00」屬性集合{製造商 = IBM，型號 = ThinkBook970，價錢 = \$399.00}。我們可以處理這個問題透過為每個要擷取的屬性各自定義出模版(又稱為模式)。這個模版是透過有限狀態機來定義，最簡單的例子是**正規式**，或稱 regex。正規式被用於諸如 grep 這類 Unix 命令中，或是類似 Perl 這樣的程式語言中，以及類似微軟公司的 Word 這類文字處理軟體中。這些工具在細節上有輕微的差別，所以最好從適當的手冊中學習，不過這裡我們要說明的是如何建立一個針對用美元表示價格的正規式，以及一些常見的子運算式：

[0-9]	相符於任何 0 到 9 的數字
[0-9]+	相符於一或多個數字
[.][0-9][0-9]	相符於一串之後接兩個數字
([.][0-9][0-9])?	相符於一串之後接兩個數字，或者無
[\$][0-9]+([.][0-9][0-9])?	相符於\$249.99 或 \$1.23 或\$1000000 或...

模版通常以三個部分定義：前綴 regex，目標 regex 與後綴 regex。對於價錢來說，目標 regex 就如上述，前綴會尋找「price:」這個字串而後綴可能為空。這個概念來自於屬性的一些線索來自於屬性值自身，一些來自於周圍的文字。

一旦某條正規式與文本完全匹配，我們就可以抽出表示屬性值的那部分文本。如果沒有匹配的情況，我們能做的是給予一個預設值，或者讓屬性空著；但是如果有多個匹配的情況，我們需要一個在匹配中間挑選的過程。一個策略是對每個屬性有多條正規式，按照優先等級排序。所以，舉個例子來說，價格的最優先模版可能會尋找前綴是「our price:」；但如果沒有找到的話，我們尋找前綴是「price:」且如果再沒找到，前綴則為空。另一個策略是接受所有的匹配情況，並尋找某些方式從中選擇合適的。例如，我們認為最低的價格是最高價格的 50%以內。那麼就會從「定價\$99.00，特價\$78.00，運費\$3.00」這些文字中選取\$78 作為目標。

比基於屬性擷取系統更近一步的是**關係擷取系統**，它處理多個物件與其間相互之關係。因此，當這些系統看到「\$249.99」時，它們不僅要判斷該文本表示價格，而且還要判斷是哪個物件的價格。FASTUS 是一個典型的基於關係(relational-based)擷取系統，它處理的是有關公司合併和獲利的新聞。它能夠閱讀如下新聞：

Bridgestone Sports Co.said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.

並且擷取出其間關係：

$$\begin{aligned}
 e \in & \text{JointVentures} \wedge \text{Product}(e, \text{"golf clubs"}) \wedge \text{Date}(e, \text{"Friday"}) \\
 & \wedge \text{Member}(e, \text{"Bridgestone Sports Co"}) \wedge \text{Member}(e, \text{"a local concern"}) \\
 & \wedge \text{Member}(e, \text{"a Japanese trading house"}).
 \end{aligned}$$

關係擷取系統可以透過一連串的**串接式有限狀態轉換器**來建立。也就是說，它們由一系列的有限狀態自動機(FSA)組成，其中每個自動機接受文本作為輸入，把文本轉換成一種不同的格式，並將其送往下一個自動機。FASTUS 系統由 5 個階段所組成：

1. 符號化。
2. 複合單字處理。
3. 基本片語處理。
4. 複合片語處理。
5. 結構合併。

FASTUS 的第 1 階段是**符號化**，它將字元串分割為符號(單字、數位以及標點)。對於英語來說，符號化過程是很簡單的，僅僅將符號流在空格處或是標點處切斷即可。某些符號化程式還要處理諸如 HTML、SGML 以及 XML 等標記語言。

第 2 個階段是處理**複合單字**，包括諸如「set up」和「joint venture」等單字搭配，以及「Prime Minister Tony Blair」和「Bridgestone Sports Co.」等專用名詞。這些是透過結合使用詞典條目和有限狀態語法規則進行識別的。例如，一個公司名稱可以透過如下規則進行識別：

CapitalizedWord+ (「Company」 | 「Co」 | 「Inc」 | 「Ltd」)

第 3 階段是處理**基本片語群**，即名詞片語群和動詞片語群。基本想法是將它們分成能在後續階段處理的單元。我們在第 23 章中會看到如何編寫一個名詞與動詞組的複雜描述，而這裡我們用一些簡單的規則來近似英語的複雜性，卻擁有可被有限狀態機描述的優點。以下是本階段可能出現的標籤分組例句：

1 NG: Bridgestone Sports Co.	10 NG: a local concern
2 VG: said	11 CJ: and
3 NG: Friday	12 NG: a Japanese trading house
4 NG: it	13 VG: to produce
5 VG: had set up	14 NG: golf clubs
6 NG: a joint venture	15 VG: to be shipped
7 PR: in	16 PR: to
8 NG: Taiwan	17 NG: Japan
9 PR: with	

其中 NG 表示名詞片語，VG 表示動詞片語，PR 是介詞，CJ 是連詞。

第 4 個階段是將基本片語合併為**複合片語**。再一次，其目標是得到有限狀態的規則，並因此能夠進行快速處理，得到無歧義(或近乎無歧義)的輸出片語結果。合併規則中的一種類型是處理領域特定事件的。例如，規則

Company+ SetUp JointVenture (「with」 Company+)?

捕捉到一種對聯合投資(joint venture)的組成進行描述的方法。該階段是系列的第一步，其中輸出既被放到資料庫模板中，同時也被放到輸出中。最後一個階段是**結構合併**，即合併前一步驟中產生的結構。如果下一句話是「The joint venture will start production in January」，那麼這一步就會注意到對聯合投資的兩次引用，它們應該被合併為一個。這是第 14.6.3 節討論的**身份不確定性問題**之實例。

一般來說，資訊擷取在有限領域中運轉比較好，因為在有限領域中預先決定會討論的主題以及用什麼方式提及它們是可能的。系列式轉換器能幫助模組化必要的知識，且減輕系統的建構。當系統是由程式產生出的逆向工程文本時，其運作會特別的好。舉例來說，當購物網站的產生是由程式透過取得資料庫條目並且將格式化為網頁形式；基於模版(template-based)擷取器於是就能夠回復出原始的資料庫。有限狀態資料擷取器用在高度變數格式裡要回復資料就比較不那麼成功，如人對於各式主題的撰寫文本。

22.4.2 機率模型應用於資訊擷取

當資訊擷取是要從雜訊資料與多樣輸入中做嘗試，簡單的有限狀態方法變得很差。要取得所有正確的規則與優先權太過困難；比起基於規則模型，使用機率模型會好得多。對於隱狀態序列的最簡單的機率模型是隱馬可夫模型(hidden Markov model，或 HMM)。

回想第 15.3 節，HMM 透過一序列的隱狀態 \mathbf{x}_t ，與每個步驟的觀察 \mathbf{e}_t 等，建模出一個前進(progression)。要應用 HMMs 於資訊擷取，我們可以替所有的屬性建立一個大的 HMM，或者也可以替每個屬性建立分別的 HMM。我們採用後者來做。文本中的詞語為觀察，我們處在不論是屬性模版的目標、前綴、後綴部分，或者在背景(非屬性模版的部份)，為隱狀態。舉例來說，這裡的簡短的文本與該文本兩個 HMMs 最大之可能(Viterbi)路徑，一是訓練成辨識出演講公告中的演講者，另一是訓練成辨識出日期。「-」指的是背景狀態：

文字：	There	will	be	a	seminar	by	Dr.	Andrew	McCallum	on	Friday
演講者：	-	-	-	-	PRE	PRE	TARGET	TARGET	TARGET	POST	-
日期：	-	-	-	-	-	-	-	-	-	PRE	TARGET

HMMs 之於 FSAs 在擷取上有兩大優勢。第一，HMMs 是機率性的，所以對雜訊有高容忍度。在正規式中，如果任一個字元遺失，regex 就無法與之相符；採用 HMMs 則對於字元/字詞的遺失能夠得體的退讓，於是我們可以達到相符程度的機率，而非布林式的符合與不符合。第二，HMMs，可由資料得到訓練；它並不需要費力的模版工程，因此當文本隨時改變時它更能輕鬆地保持在最新。

請注意到我們已假設 HMMs 模版中結構的特定程度：它們包含一或多個目標狀態，且任何其前綴狀態必須先於目標，後綴必須跟在目標之後，其餘狀態必須式在背景中。這樣的結構使得從例子中學習 HMMs 變得很簡單。利用部分特定結構，前後向演算法(forward-backward algorithm)可用來學習狀態間的轉移機率 $\mathbf{P}(\mathbf{X}_t | \mathbf{X}_{t-1})$ 與觀察模型 $\mathbf{P}(\mathbf{E}_t | \mathbf{X}_t)$ ，也說明了每個詞語在每個狀態中的可能。舉例來說，「星期五」這個詞語在日期 HMM 中的一或多個目標狀態有很高的機率，在其他 HMM 中就很低了。

給定足夠的訓練資料，我們會直觀的發現 HMM 自動學習到日期的結構：日期 HMM 也許有一個目標狀態中高機率的詞語是「星期一」、「星期二」等，且可能有很高機率轉移到一個目標狀態含有「Jan」，「January,」 「Feb,」等。圖 22.2 表示從資料的學習，得到演講公告的演講人的 HMM。前綴涵括了表達式「Speaker:」與「seminar by,」且目標存在一個狀態是涵括頭銜與名字，其他狀態涵括縮寫與姓。

一旦 HMMs 已經得到學習，我們可將之應用到文本上，使用 Viterbi 演算法經由 HMM 狀態來找到最可能路徑。一種方法是分別應用每個屬性 HMM；在這種情況你會期待大多數的 HMMs 花費其多數時間在背景狀態裡。當擷取是稀疏時這種方法是很適當的——擷取後的詞語比起文本的長度還小。

其它的方法係結合所有的個別屬性成爲一個大的 HMM，然後會找到一個路徑徘徊在不同的目標屬性間，先是找到演講者目標，接著是日期目標等等。分開的 HMMs 對於我們只期待一個屬性來說較佳，一個大的 HMM 對於文本爲較自由的格式與密集的屬性來說較佳。不論任一種方法，我們最終會得到目標屬性觀察，接著必須要決定如何處理這些觀察。假如每個預期的屬性都具有一個目標充填(filler)那麼做決定就很簡單：我們有一個所求關係的實例。如果具有多個充填，我們就必須決定選擇何者，如我們討論的基於模版系統。HMMs 具有提供機率數字的優點，可幫助做出決定。假設一些目標遺失，我們必須要決定這是否是實例所需之全部的關係，或者目標爲誤報(false positives)。機器學習演算法可以被訓練來做決定。

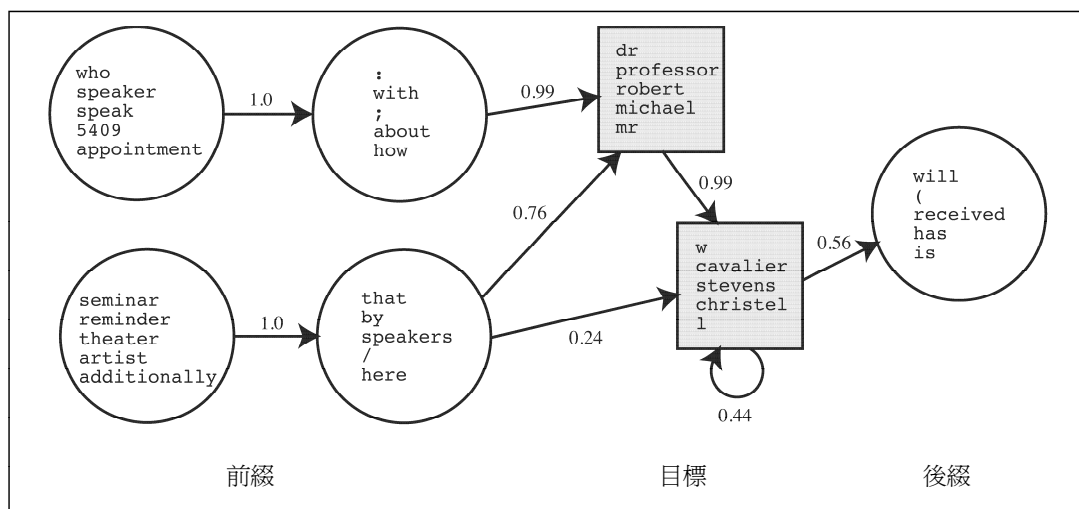


圖 22.2 隱馬可夫模型應用於演講公告的主講者。兩個正方塊狀態為目標(注意到第二個目標狀態帶有自迴圈，因此目標可以為符合任何長度的字串)，左邊四個圈圈是前綴，右邊圈圈是後綴。對每個狀態來說，僅顯示高機率的詞語。來源為 Freitag 及 McCallum(2000)

22.4.3 條件式隨機域應用於資訊擷取

一個 HMMs 應用到資訊擷取任務的議題是它建模出很多我們不見得需要的機率。HMM 是生成模型；它建模出觀察的完整的組合機率與隱狀態，因此可被用於產生樣本。而這是說，HMM 模型不僅可應用在解析文本並回復出主講人與日期，也可以產生出包含主講人與日期的文本之隨機實例。既然我們對於該任務不感興趣，很自然地我們要問是否可以更好地去利用一個模型，但不干擾到該模型建模其機率。所有我們對於了解文本所需要的就是**辨識模型**(discriminative model)，其在給定的觀察(文本)下，會為隱屬性的條件式機率建模。給定文本 $\mathbf{e}_{1:N}$ ，條件式模型會找出隱狀態序列 $\mathbf{X}_{1:N}$ 使得 $P(\mathbf{X}_{1:N} | \mathbf{e}_{1:N})$ 最大化。

對其建模直接地給了我們一些自由度。我們不需馬可夫模型中的獨立假設——我們可得 \mathbf{x}_i 獨立於 \mathbf{x}_1 之外。這種模型的類型之架構為**條件式隨機域**(Conditional random fields)，或 CRF，給定一個觀察變數集合，建模出一個目標變數的集合之條件機率分布。如同貝氏網路，CRFs 可以表示許多變數間許多依存性的不同結構。一個普遍的結構是**線性鏈**(linear-chain)條件式隨機域，用以代表時序序列中，變數間的馬可夫依存性。因此，HMMs 為原始貝氏模型的時序版本，而線性鏈 CRFs 是羅吉斯回歸(logistic regression)的時序版本，其中預測目標是一個完整的狀態序列，而非單一的二元變數。

令 $\mathbf{e}_{1:N}$ 為觀察(例如, 文件中的詞語), 而 $\mathbf{x}_{1:N}$ 是隱狀態的序列(例如, 前綴、目標與後綴狀態)。線性鏈條件式隨機域機率分佈:

$$P(\mathbf{x}_{1:N} | \mathbf{e}_{1:N}) = \alpha e^{\left[\sum_{i=1}^N F(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{e}, i) \right]}$$

α 是正規化因子(用以確認機率總和為 1), F 是一個特徵函式, 定義為 k 元件特徵函式集合之權重和:

$$F(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{e}, i) = \sum_k \lambda_k f_k(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{e}, i)$$

λ_k 參數值是利用 MAP(maximum a posteriori)估計程序學得, 其會最大化訓練資料的條件式概似。特徵方程為 CRF 中最關鍵的部分。函式 f_k 可以存取相鄰狀態對 \mathbf{x}_{i-1} 與 \mathbf{x}_i , 也可存取整個觀察(詞組)序列 \mathbf{e} , 還有時序序列中的目前位置 i 。這給予我們在定義特徵時有很多的彈性。我們可以定義一個簡單的特徵函式, 舉例來說假設目前的詞語是 ANDREW 且目前狀態在 SPEAKER, 會產生數值等於 1:

$$f_1(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{e}, i) = \begin{cases} 1 & \text{若 } \mathbf{x}_i = \text{SPEAKER 且 } \mathbf{e}_i = \text{ANDREW} \\ 0 & \text{其他} \end{cases}$$

像這些特徵要如何使用?它取決於他們相對應的權重。假設 $\lambda_1 > 0$, 那麼不管 f_1 是否為真, 增加隱狀態序列 $\mathbf{x}_{1:N}$ 的機率。還有一個「CRF 模型對於詞語 ANDREW 的偏好目標狀態 SPEAKER」的說法。假設另一方面 $\lambda_1 < 0$, CRF 會試著避免這個關連, 而如果 $\lambda_1 = 0$, 這個特徵就會被忽略。參數值可以被手動設定或者從資料中學得。現在考慮第二個特徵函式:

$$f_2(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{e}, i) = \begin{cases} 1 & \text{若 } \mathbf{x}_i = \text{SPEAKER 且 } \mathbf{e}_{i+1} = \text{SAID} \\ 0 & \text{其他} \end{cases}$$

假設目前狀態是 SPEAKER 且下一個詞語是「said」, 特徵為真。因此跟著此特徵會接著期待正的 λ_2 值。有趣的是, 注意到 f_1 與 f_2 在如「Andrew said...」可以保持在同一時間。在這個例子來看, 這兩個特徵相互重疊且兩者在 $\mathbf{x}_i = \text{SPEAKER}$ 都增強了信度。因為各自獨立的假設, HMMs 無法重疊特徵; 但 CRFs 卻可。更甚者, CRF 中的特徵可以使用 $\mathbf{e}_{1:N}$ 序列中的任何部分。特徵可以定義成佈於狀態間的轉移。我們這裡定義的特徵是二元的, 但一般來說, 特徵函式可以是任何實數值函式。對於我們有一些想要納入的特徵型態之知識的領域, CRF 形式化方法給了我們很大的彈性來定義它們。這個彈性可以導致更高的精確度, 比起較不彈性的模型如 HMMs。

22.4.4 從巨大語料庫中做本體擷取

到目前為止我們已經把訊息擷取想像成尋找關係的特定集合(例如, 主講者, 時間, 地點)在特定的文本中(例如, 演講公告)。不同的擷取技術的應用係建立了巨大的知識基礎上或是從語料庫中建立事實的本體論。有三個方面是不同的: 第一個是開放性問題(open-ended)——我們希望獲得的是關於所有類型領域的事實, 而不只是一個特定的領域。第二, 透過大型語料庫, 這個任務是由高精度所支配, 並非召回率——如同網頁裡的問答(第 22.3.6 節)。第三, 統計的結果可以從多個來源的匯總而收集到, 不只是擷取一個特定的文本。

舉例來說，Hearst(1992)觀察學習概念範疇的本體論，與從大型語料庫的次範疇學習問題。(在1992年時，巨大語料庫指的是1000個頁面的百科全書；到今天指的可能是一億個頁面的網路語料庫)。此工作專心於一般化(不綁定於一個特定領域)的模版且有高精確度(當相符時往往是正確無誤的)，但低召回率(並非永遠相符)。下面舉的是最多產模版其中之一：

NP such as NP (, NP) (,)? ((and | or) NP)?*

這邊粗體字與逗號必須要在文字中逐字出現，括號是用來標示一組，星號表示零或多個重複，問號表示可有可無。NP 是變數，代表了一個名詞組；第23章描述如何便是名詞組；現在僅需假設我們知道一些詞語是名詞，其它詞語(如動詞)我們可以可靠地假設其非簡單的名詞組部分。這個模版相符於文本「狂犬病等疾病影響你的狗」且「支援網絡協議如 DNS」可推斷出狂犬病是一種疾病且 DNS 是一種網路協議。相類似的模版可以用關鍵字「including」，「especially」與「or other」來建構。當然這些模版會無法相符很多有關的段落，如「狂犬病是一種疾病」。這是有意的。「NP is a NP」模版確實有時候意味著子範疇關係，但通常其代表其它意思，就如在「There is a God」或者「She is a little tired」。使用大型語料庫我們可以負擔得起挑剔，如僅使用高精確度的模版。我們會遺失許多子範疇關係的聲明，但最有可能的是我們會在語料庫中的某些地方，找到可使用聲明的轉述(paraphrase)。

22.4.5 自動模版建構

子範疇關係是這麼的基礎以致於值得手工處理一些模版，來在自然語言文本中幫助辨識所發生的實例。但世界上其它數以千計的關係怎麼處理？世界上沒有足夠的 AI 研究生可以為所有的模版創造與解問題。幸運的是，透過幾個例子來學習模版是可能的，然後使用這些模版來學習更多的例子，從這些例子又可以學習更多的模版等等。從首次幾個試驗的其中之一，Brin (1999)從一個僅含五個例子的資料集合開始：

(“Isaac Asimov”, “The Robots of Dawn”)
 (“David Brin”, “Startide Rising”)
 (“James Gleick”, “Chaos—Making a New Science”)
 (“Charles Dickens”, “Great Expectations”)
 (“William Shakespeare”, “The Comedy of Errors”)

很顯然地這些是作者-書名關係的範例，但學習系統沒有任何關於作者或者書名的知識。這些例子裡頭的詞語被用來在網頁語料庫中搜尋，產生了199條結果。每個相符被定義為七個字串為一組。

(*Author, Title, Order, Prefix, Middle, Postfix, URL*)
 (作者，書名，排序，前綴，中間字元，後綴，網路位址)

當作者首先出現，*Order* 為真而當書名首先出現，*Order* 為假，*Middle* 表示作者與書名中的字元，*Prefix* 是相符詞的前十個字元，*Suffix* 是相符詞的後十個字元，*URL* 是相符詞出現其中的網路位址。

給定一組相符詞的集合，簡易的模版生成方法(template-generation scheme)可以找到模版來解釋這些相符詞。模版的語言被設計來表示相符詞自身緊密的對應，服從自動化學習，且強調高精確度(可以會處於較低召回率的風險)。每個模版都有相同的七個組成成為一個相符詞。*Author* 與 *Title* 為包含任意字元(但起始與結束為文字)的 *regexes*，且約束成帶有最小例子長度一半到兩倍於最大長度。前綴，中間字元與後綴被限制成文字字串，非 *regexes*。中間字元最容易學習：相符詞集合中的每個不同的中間字串即是不同的候選模版。對於每個這種候選者，模版的 *Prefix* 於是定義成相符詞中所有前綴裡最長的一般後綴，*Postfix* 於是定義成相符詞中所有後綴裡最長的一般前綴。如果前述二者任一長度為零，該模版就被退回。模版的 *URL* 定義成相符詞中，URLs 最長的後綴。

Brin 所做的實驗運行中，最初的 199 個相符產生了三個模版。最有生產力的模版是

```
<LI><B> Title </B> by Author (
```

```
URL: www.sff.net/locus/c
```

這三個模版接著被使用來檢索 4047(作者，書名)個更多的例子。這些例子接著被用來產生更多的模版等等，最終生產了超過 15,000 條目。給定一個好的模版集合，系統就可以搜集到一個好的例子集合。給定一個好的例子集合，系統就可以建構一個好的模版集合。

這個方法的最大弱點在於對雜訊很敏感。假使前幾個模版之一不正確，錯誤會快速地繁殖。限制這個問題的一個方法是不要接受新的例子，一直等到有多個模版驗證過才接受，且也不要接受新的模版，一直等到它發現多個例子同時被其它的模版所發現。

22.4.6 機器閱讀

從手工模版建構到自動板模建構是非常大的一步，它仍須一把有關係的有標的例子來起始。要建立一個帶有數千個關係的巨大本體，即使工作量非常的艱鉅；我們仍想有一個不需人用任何形式的輸入的擷取系統——此系統可以自行閱讀且建立其資料庫。這種系統是獨立關係(relation-independent)；可以運行於任何關係。實行上，因為大型語料庫的 I/O 需求，這些系統平行運作於各種關係。他們表現的較不像傳統的資訊擷取系統那樣把目標定在幾個少數的關係上，其更像人類閱讀者從文本裡頭學習；正因如此這個領域被稱為**機器閱讀**。

一個代表性的機器閱讀系統是 TEXTRUNNER (Banko 與 Etzioni, 2008)。TEXTRUNNER 採用共同訓練(cotrainning)來加速其表現，但它需要一些可以使其開始自力更生的東西存在。Hearst (1992) 的例子裡，特定的模式(例如，*such as*)提供其自力更生，且對於 Brin(1998)來說，那是五對作者-書名的集合。對 TEXTRUNNER 來說，最初的靈感來自於一個很籠統的分類八句法模板，如圖 22.3 所示。有部分認為像這樣少數的模版，可以涵括大部分用英語表達的關係的方法。真正自力更生開始於有標例子，這些例子擷取自 Penn Treebank，一個解析過句子的語料庫。舉例來說，從「Einstein received the Nobel Prize in 1921」這句句子的解析，TEXTRUNNER 是有能力擷取關係(「Einstein」「received」「Nobel Prize」)。

Type(類型)	Template(模版)	Example(範例)	Frequency(頻率)
Verb	NP_1 Verb NP_2	X established Y	38%
Noun-Prep	NP_1 NP Prep NP_2	X settlement with Y	23%
Verb-Prep	NP_1 Verb Prep NP_2	X moved to Y	16%
Infinitive	NP_1 to Verb NP_2	X plans to acquire Y	9%
Modifier	NP_1 Verb NP_2 Noun	X is Y winner	5%
Noun-Coordinate	NP_1 (, and - :) NP_2	X-Y deal	2%
Verb-Coordinate	NP_1 (, and) NP_2 Verb	X, Y merge	1%
Appositive	NP_1 NP (: ,)? NP_2	X hometown :Y	1%

圖 22.3 八個一般性模版可以涵括 95%英語中表達關係的方法

給定這種類型的有標例子集合，TEXTRUNNER 訓練一個線性鏈 CRF 從無標的文本中擷取更進一步的例子。這個 CRF 中的特徵包含了函式字元如「to」、「of」與「the」，但並非名詞與動詞(也非名詞組或者動詞組)。因為 TEXTRUNNER 為 domain-independent(域獨立)，它無法依賴在預定義的名詞與動詞列表。

TEXTRUNNER 在大的網頁語義庫中，可達成 88%的精確度且 45%的召回率(60%的 F_1)。TEXTRUNNER 已經從一個五千萬網頁語料庫中擷取數千個百萬的事實。舉例來說，即使它不具有預定義的醫藥知識，它可透過查詢[what kills bacteria]來獲得超過 2000 個答案；正確解答包含 antibiotics, ozone, chlorine, Cipro, 與 broccoli sprouts。可疑的答案包含了「water」，來自於這個句子「Boiling water for at least 10 minutes will kill bacteria」將其屬性改成「boiling water」會比僅是「water」來的好的多。

利用本章所敘述的技術和持續不斷的新發明，我們開始離機器閱讀的目標越來越近了。

22.5 總結

本章要點包括：

- 基於 n 元的機率語言模型能夠獲得有關語言的為數驚人的資訊。它們在語言識別、拼字校正、體裁分類和命名實體辨識如此多樣的任務上表現良好。
- 這些語言模型具有百萬種的特徵，所以為降低雜訊的特徵選擇及資料預處理是很重要的。
- 文本分類可以用簡單貝氏 n 元模型或用先前討論過任何的分類演算法。分類在資料壓縮也被視為一個問題。
- 資訊檢索系統採用一個基於詞袋的非常簡單的語言模型，但仍能設法在超大規模的文本語料庫上表現出良好性能(從召回率和準確率的角度而言)。在網路語料庫連結分析演算法改善了效能。
- 對於在語料庫中有多種答案的問題，問答可以用資訊檢索的方法來掌控。當多個答案存在於語料庫之中，我們可以使用強調精準的技術而非召回。
- 資訊擷取系統採用一個更為複雜的模型，其包含了句法和語義的有限概念(以模版形式)。它們可從有限狀態自動機、HMM 或條件隨機場建立，並且可從實例學習。
- 在建立一個統計語言系統時，最好設計一個能很好地利用資料的模型，即使該模型看起來極其簡單。

◎ 參考文獻與歷史的註釋 BIBLIOGRAPHICAL AND HISTORICAL NOTES

針對語言建模的 n 元字元模型是由馬可夫(Markov, 1913)提出的。克勞德·香農(Shannon 及 Weaver, 1949)是第一個建立英語 n 元單字模型的。杭士基(Chomsky, 1956, 1957)指出了有限狀態模型與上下文無關模型相比較的局限性，並推斷「機率模型不能提供關於某些句法結構的基本問題的特殊見解」。這是對的，機率模型確實能夠為某些其他基本問題提供見解這一事實——那些問題是 CFG 忽略的。杭士基的評論產生了災難性的後果，使得很多人不敢去碰機率模型長達 20 年的時間，直到這些模型再次湧現出來，被用於語音識別中(Jelinek, 1976)。

Kessler *et al.*(1997)表示如何應用 n 元模型特徵來做體裁分類，且 Klein *et al.*(2003)描述以特徵模型命名實體辨識。Franz 和 Brants(2006)描述從網頁本文 1 兆個單字中 1300 萬單一字彙的 Google 的 n 元語料庫，現在已經公開。**詞彙袋**模型從段落得到名稱，語言學家 Zellig Harris(1954)說「語言不只是詞彙袋也是具有特定屬性的工具」。Norvig(2009)給出一些可以用 n 元模型完成的任務實例。

附加平滑首先是 Pierre-Simon Laplace(1816)提議的而由 Jeffreys(1948)公式化，使用於語音識別的內差平滑是由 Jelinek 和 Mercer(1980)提出。其他技術還包括 Witten-Bell 平滑(1991)及古德-圖靈(Good-Turing)平滑(Church 和 Gale, 1991)。Chen 和 Goodman(1996)及 Goodman(2001)研究平滑技術。

簡單的 n 元字元和單字模型不是唯一可能的機率模型。Blei 等人(2001)描述了一個被稱為**潛在狄利克雷分配**(latent Dirichlet distribution)的機率文本模型，它將文件視為主題的混合物，每個主題都有自己的單字分佈。該模型可以被視為 Deerwester 等人(1990)的**潛在語義索引**(latent semantic indexing)模型的一種擴展和合理化實作(參見 Papadimitriou 等人, 1998)，而且該模型也與 Sahami 等人(1996)的多因混合模型有關係。

Manning 和 Schütze (1999)及 Sebastiani(2002)研究本文分類技術。Joachims(2001)利用統計學習理論和支持向量機給出了何時分類能夠成功的理論分析。Apté等人(1994)在將路透社新聞文章分到「收入」類別的過程中達到了 96%的準確率。Koller 和 Sahami(1997)用朴素貝氏分類器達到了最高 95%的準確率，而用一個考慮了特徵中的某些依賴性的貝氏分類器則達到了最高 98.6%的準確率。Lewis(1998)對朴素貝氏技術在文本分類和檢索中 40 年的應用進行了綜述。Schapire 和 Singer (2000)表示簡單的線性分類器可達到與複雜模型相同的精確度並且更有效率地評估。Nigam *et al.*(2000)表示如何使用 EM 演算法來標示未經標示的文件，因此學習較佳的分類模型。Witten *et al.*(1999)敘述用於分類的壓縮演算法，並表示介於 LZW 壓縮演算法和極大熵的語言模型的深層連結。

許多 n 元模型技術同樣地使用在生物資訊問題中。生物統計學和機率 NLP 越走越近，因為處理的都是從成份表中選出的結構化長序列。

資訊檢索領域正越來越受到重視，這是被 Internet 搜尋的廣泛應用激發的。Robertson(1977)給出了一個早期的全面評述，並引入了機率排序原則。Croft *et al.*(2009)以及 Manning *et al.*(2008)是第一本如同傳統 IR 談論到網頁基礎的搜尋之著作。Hearst(2009)涵蓋了對於網頁搜尋的使用者介面。由美國政府的國家標準與技術協會(NIST)組織的 TREC 會議，舉辦每年一次的 IR 系統競賽，並出版包含結果的會議論文集。在競賽的頭 7 年中，系統性能差不多提高了一倍。

最流行的 IR 模型是**向量空間模型**(Salton *et al.*, 1975)。Salton 的工作在該領域的早年佔據了主導地位。有兩種不同的機率模型，其中一個是來自於 Ponte 和 Croft(1998)而另一個是由 Maron 和 Kuhns(1960)以及 Robertson 和 Sparck Jones(1976)。Lafferty 和 Zhai(2001)證明了這些模型是基於相同的聯合機率分佈的，但是模型的選擇暗含了對參數的不同訓練方法。Craswell *et al.*(2005)描述 BM25 計分函數且 Svore 和 Burges (2009)描述 BM25 如何可以合併點選資料加強機器學習方法——過去實例搜尋串列與點選的結果。

Brin 和 Page (1998)描述 PageRank 演算法與網頁搜尋引擎的建構。Kleinberg(1999)描述了 HITS 演算法。Silverstein 等人(1998)研究了一個有十億條 Web 搜尋的日誌。《資訊檢索》(*Information Retrieval*)期刊和每年的 SIGIR 會議的論文集涵蓋了該領域中新進的發展。

早期的資訊擷取程式包括 GUS(Bobrow 等人, 1977)以及 FRUMP(DeJong, 1982)。近來的資訊擷取由於每年一次的「訊息理解會議」(Message Understanding Conferences, MUC)得到推動，該會議得到了美國政府的資助。Hobbs *et al.*(1997)完成了 FASTUS 有限狀態系統。它的部分構想是基於 Pereira 和 Wright (1991)使用 FSA 為片語結構文法的近似。對於樣版為基礎的系統之研究由 Roche 和 Schabes (1997)、Appelt(1999)以及 Muslea(1999)。大型資料庫的論據由 Craven *et al.*(2000)、Pasca *et al.*(2006)、Mitchell (2007)以及 Durme 和 Pasca(2008)所得到。

Freitag 和 McCallum(2000)討論對於 HMM 的資訊擷取 CRF 是由 Lafferty *et al.*(2001)所推行，一個它們使用資訊擷取的實例在(McCallum, 2003)中描述，(Sutton 及 McCallum, 2007)提供了一個實務指引的指南。Sarawagi(2007)做了綜合性的研究。






Banko *et al.*(2002)提出 ASKMSR 問答系統，另一個相似的系統是由 Kwok *et al.*(2001)提出。Pasca 和 Harabagiu(2001)探討一個競賽獲獎問答系統。兩個在自動化知識工程最近具有影響力的方法由 Riloff(1993)，他表示了自動地建構辭典，如同小心地人工製作特殊領域辭典一樣地好，而 Yarowsky(1995)表示詞義分類的任務(參考 18.11.2 節)可經由在未標示本文的語料庫，不受監督學習的精準度如同受監督學習一樣地好。

Blum 及 Mitchell(1998)將同步擷取樣板的概念與實例從少量的標示實例，發展為獨立且同步地，將它稱為**合作訓練**(cotraining)且 Brin(1998)將它稱為 DIPRE(Dual Iterative Pattern Relation Extraction)。你可以看到為何合作訓練動彈不得。早期相似的研究，Jones *et al.*(1999)在名稱的引導之下完成。這方法由 QXTRACT(Agichtein 和 Gravano, 2003)以及 KNOWITALL(Etzioni *et al.*, 2005)系統發展。機器閱讀是 Mitchell (2005)及 Etzioni *et al.*(2006)所引導，且焦點在於 TEXTRUNNER 這一個專案(Banko *et al.*, 2007；Banko 和 Etzioni, 2008)。

本章的焦點在於自然語言本文，但也可能基於實體結構或本文佈局進行資訊擷取，而不是在語言結構。HTML 列舉及表格同樣在 HTML 和相關資料庫可以被擷取並且合併(Hurst, 2000；Pinto *et al.*, 2003；Cafarella *et al.*, 2008)。

計算語言學協會(ACL)定期舉行會議並出版《計算語言學》(*Computational Linguistics*)期刊。同時還有一個「計算語言學國際會議」(COLING)。Manning 和 Schütze (1999)的書涵蓋到統計語言處理，而 Jurafsky 和 Martin(2008)對於語音和自然語言處理有全面的介紹。

❖ 習題 EXERCISES

- 22.1  本習題將探索 n 元語言模型的性質。找到或建立一個包含大約 100,000 個單字的詞語語料庫。對其進行分詞，並計算每個單字的頻度。有多少個不同的單字？同時，也計算二元(兩個連續的詞)以及三元(三個連續的詞)的頻度。現在，利用這些頻度產生語言：根據頻度隨機選擇單字，依次用一元、二元、三元模型產生一段 100 個詞的文本。將這 3 段產生的文本與實際語言相比較。最後，計算每個模型的混亂度。
- 22.2  試寫出一個程式來斷出沒有間隔的單字。給定一個 URL 的字串「thelongestlistofthelongeststuffatthelongestdomainnameatlonglast.com,」回傳元件單字的列表：[「the,」 「longest,」 「list,」 ...]。這個任務對於解析 URL 很有用，當單字混和在一起或中文字之間沒有空格的要拼字校正時候。當一元或二元單字模型的時候可被解決，且動態規劃演算法相似於維特比(Viterbi)演算法。
- 22.3  [改編自 Jurafsky 和 Martin(2000)的著述]。在本習題中，我們將開發一個作者分類器：給定一段文字，分類器將預測兩個候選作者中是誰寫這段文字。事先得到來自兩個不同作者的文字樣本。將它們分為訓練集和測試集。現在於訓練集中訓練個語言模型。你可以選定使用哪個屬性， n 元的單字或是字母是最先的，且你可以依照覺得可能有幫助來加入額外的屬性。然後在每個語言模型計算出本文的機率並且選擇最可能的模型。評估該技術的準確率。當你更動屬性的集合後精準度會如何改變？語言學的這個子領域被稱為**語言風格分析**；它的成功例子包括對於受爭議的《聯邦主義者文集》(*Federalist Papers*)(Mosteller 和 Wallace, 1964)作者的鑒別，以及對於某些有爭議的莎士比亞作品的鑒別(Hope, 1994)。Khmelev 和 Tweedie(2001)以簡單字母二元模型產生好的結果。
- 22.4  本習題是關於垃圾郵件的分類。分別建立一個垃圾郵件以及一個非垃圾郵件的語料庫。檢查每個語料庫，並決定哪些出現的特徵對分類有用：一元字(unigram words)？雙元？訊息長度、發信人、到達時間？然後在訓練集上訓練一個分類演算法(決策樹、樸素貝氏或者其他某個你選擇的演算法)，並報告它在測試集上的準確率。
- 22.5 建立一個包含 5 個查詢的測試集，把它們提交給 3 個主要的 Web 搜尋引擎。估計每個在 1, 3 和 10 個文件的準確度。可否試解釋各引擎之間的差異？
- 22.6 試著根據前一道習題確定哪個搜尋引擎採用了大小寫同一、取詞幹、同義詞以及拼寫錯誤更正技術。
- 22.7  編寫一個正規式或者一個簡短的程式用來擷取公司名稱。並在商業新聞文章語料庫上對其進行測試。報告你的準確率和召回率。
- 22.8 考慮一個嘗試評估 IR 系統品質的問題，其回傳個經過評比的答案(類似大多數的網路搜尋引擎)。品質的適當措施根據於假設模型其中檢索者努力達成它所使用的策略。對於下列模型，試提出相符的數字測量。

- a. 檢索者會察看前 20 個回傳的答案，儘可能得到較相關的資訊。
- b. 檢索者僅需要一個相關文件，且會持續地向下尋找直到發現第一個。
- c. 檢索者將均勻地縮短查詢且驗證所有答案檢索。它必須確定看過文件中每件與它搜尋相關的事(例如，律師希望確定它已經找到所有相關判例，並且要運用相當多的資源在上面)。
- d. 檢索者需要一個與搜尋相關的文件，並且請研究助理花時間看過這些結果。助理可以在每個小時看過 100 筆檢索文件。助理將會更換檢索者不論它立刻找到的或是在時間的結束。
- e. 檢索者將會找過所有的答案。驗證這些文件的花費是 $\$A$ ，找出相關文件具有 $\$B$ 的價值，搜尋相關文件失敗的花費是 $\$C$ ，對於每個沒有找到相關文件。
- f. 檢索者希望儘可能地蒐集所有相關文件，但需要持續地鼓勵。它依序看過所有的文件。若它找到目前為止最好的文件，則會繼續下去，否則將會停止。

本章註腳

- [1] 隨著 T. Geisel(1955)開創性研究的可能例外。
- [2] 我們將搜索查詢標註為[query]。使用方形括弧而不是用引號可以區別出搜尋[「two words」]和[two words]。
- [3] 這命名是基於網頁和它的共同發名人 Larry Page(Brin 及 Page, 1998)。

