A LaTeX Book Template for Chinese Scientific Publication

Designed for Pandoc

Nim Chimpsky

目錄

第 1 章

語料中的搭配現象

語言是一個能夠攜帶並表達意義的系統。為了能傳達意義...

一般來說，語言中的規律可以出現在任何一個語言單位上 (語音、詞素、字、詞彙、句子、篇章等)[1]，但傳統上，語料庫語言學關注的焦點是**詞彙**...要理解這兩種組合的差異相當簡單—我們知道礦泉水通常是以「瓶子」去盛裝，而較少以鍋子盛裝[2]...

上述所舉的例子在語言學中稱為**搭配語** (collocation)。定義上...：$p(\text{瓶}) \times p(\text{礦泉水}) = \frac{41}{50000} \times \frac{13}{50000} = 2.132 \times 10^{-7}$ ，「鍋」與「礦泉水」搭配出現的機率會是：$p(\text{鍋}) \times p(\text{礦泉水}) = \frac{39}{50000} \times \frac{13}{50000} = 2.028 \times 10^{-7}$ 。此時，我們便可以**將隨機排序當作基準**與**實際的資料**進行比較：

$$\begin{aligned} \frac{p(\text{瓶}, \text{礦泉水})}{p(\text{瓶})p(\text{礦泉水})} &= \frac{11/13}{2.132 \times 10^{-7}} = 3968826.671 \\ \frac{p(\text{鍋}, \text{礦泉水})}{p(\text{鍋})p(\text{礦泉水})} &= \frac{1/13}{2.028 \times 10^{-7}} = 379305.113 \end{aligned} \tag{1.1}$$

這些數值可以直觀地詮釋為...

值得注意的是，搭配現象不只適用在詞彙單位上...至**構式** (construction)。

[1] 因為語言不論在哪個單位...

[2] 當然，在現實世界中或許礦泉水**沒有**較...

1.1　詞彙之間的搭配：搭配語 (Collocation)

1.1.1　操作化「搭配」現象：列聯表 (Contingency Table)

舉例來說，下方的資料中共有 6 筆觀察值，記錄了 6 個人的資訊，並包含兩個類別變項 (1) **教育程度**以及 (2) **性別**：

id	教育程度	性別
1	國中	男
2	高中職	女
3	碩士	女
4	大專	女
5	大專	男
6	高中職	男

若想知道教育程度與性別有無關聯...

	國中	高中職	大專	碩士
男	1	1	1	0
女	0	1	0	2

1.2　搭配語的程式實作

1.2.1　節點詞與搭配詞

前文在介紹搭配現象時，並未特別區分節點詞 (node word) 與搭配詞 (collocate) ...

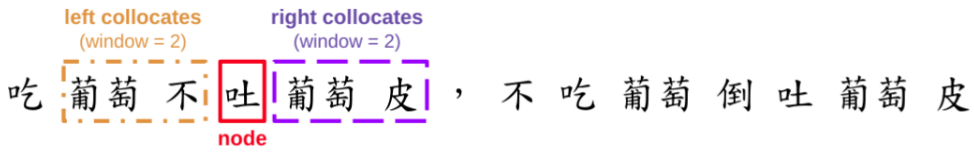

圖 1.1: 節點詞與搭配詞

了解「節點詞/搭配詞」的定義後，我們以滑動窗口的演算法計算句中各種 (節點詞，搭配詞) 組合出現的次數 (見圖 1.2)...

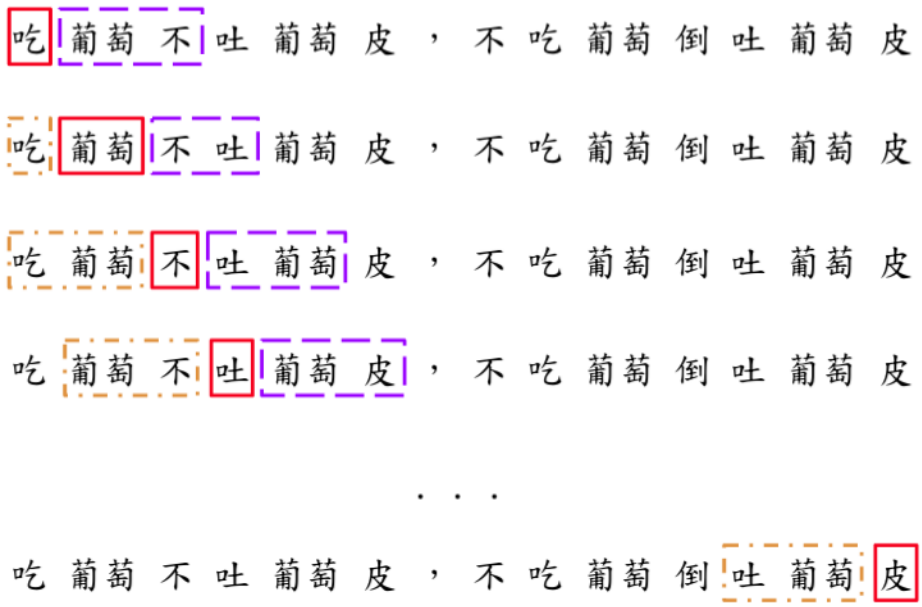

圖 1.2: 以滑動窗口演算法計算句中「節點詞/搭配詞」組合次數

```
def count_freq_sent(sentence, left_window, right_window):
    cooccur_freq = {} # 共現頻率資料
    sent_len = len(sentence)
    return cooccur_freq
```

在 `count_freq_sent()` 函數中，內層 for 迴圈的第三行是 `k = (node, collocate)`。因此，詞組內的第一個詞是節點詞，第二個詞是搭配詞，並非依照詞彙於句中出現的順序排列。例如：`('葡萄', '吃')` ...

在上方程式碼中，我們將左右窗口大小皆設為 2，因此搭配詞可能會出現在節點詞之前或之後。事實上，左右窗口的大小不一定要相同，依據目的不同，可以填入不一樣的左右窗口。舉例來說，有時候我們感興趣的是出現在節點詞**後方**的搭配詞...

1.2.2 量化搭配強弱

我們要計算的列聯表如表 1.3 與表 1.4。O_{11}、O_{12}、O_{21}、O_{22} 為各種組合之下的頻率觀察值；R_1、R_2、C_1、C_2 則為邊際頻率 (marginal frequency)。

表 1.3: 節點詞/搭配詞列聯表 (實際頻率)

	Wc	~Wc	Total
Wn	O_{11}	O_{12}	R_1
~Wn	O_{21}	O_{22}	R_2
Total	C_1	C_2	N

表 1.4: 節點詞/搭配詞列聯表 (頻率期望值)

	Wc	~Wc	Total
Wn	$E_{11} = \frac{R_1 C_1}{N}$	$E_{12} = \frac{R_1 C_2}{N}$	R_1
~Wn	$E_{21} = \frac{R_2 C_1}{N}$	$E_{22} = \frac{R_2 C_2}{N}$	R_2
Total	C_1	C_2	N

如果我們想知道 (節點詞：電影，搭配詞：日本) 這個組合在 corpus 中的搭配情形，可以先列出以下列聯表：

$$MI = log2(\frac{O_{11}}{E_{11}})$$

$$\begin{aligned}\chi^2 &= \sum_j \frac{(O_j - E_j)^2}{E_j} \\ &= \frac{(O_{11} - E_{11})^2}{E_{11}} + \frac{(O_{12} - E_{12})^2}{E_{12}} + \frac{(O_{21} - E_{21})^2}{E_{21}} + \frac{(O_{22} - E_{22})^2}{E_{22}}\end{aligned}$$

$$\begin{aligned}G^2 &= 2\sum_j O_j log\frac{O_j}{E_j} \\ &= 2(O_{11} log\frac{O_{11}}{E_{11}} + O_{12} log\frac{O_{12}}{E_{12}} + O_{21} log\frac{O_{21}}{E_{21}} + O_{22} log\frac{O_{22}}{E_{22}})\end{aligned} \tag{1.2}$$

第 2 章

構式中的搭配：Collostruction

在前面的幾個段落裡，我們了解搭配語是出現在特定窗口大小 內的詞彙組。這個判斷方式已足以觀察到語料中一些穩定的現象，但卻未能利用語言的一項重要特性—語法關係 (syntactic relation) ...

2.1 關於

在語料庫語言學裡，這方面的研究有兩個略有差異的方向。這個方向就是語料庫工具 Sketch Engine 中的詞彙素描 (word sketch) 背後的基礎 (Huang et al., 2005; Kilgarriff et al., 2014; Kilgarriff, Rychlý, Smrz, & Tugwell, 2004)，本書第八章有詳細介紹。第二個方向 (Gries & Stefanowitsch, 2004; Stefanowitsch & Gries, 2005; Stefanowitsch & Gries, 2003) 則是受**構式語法** (Construction Grammar) (Goldberg, 1995) 的啟發，關注的是特定構式 之下語法單位間的搭配關係。本章將著重介紹第二個方向。

2.1.1 資料準備

由於在搭配現象中引入了「語法」的概念，我們需要語法相關的資訊，亦即詞類標記 (part of speech, PoS) ...

```
pip install -U ckiptagger[tf,gdown]
pip install -U concordancer
```

2.1.2　搜尋「把」字句

有了剛剛的準備，我們便能開始從語料中搜尋抽象的...

「把」字句的結構是「把 + (賓語) + (動作)」，例如，「把 **時間** (賓語) **花** (動作) 在」。下方是「把」字句的 CQL 結構，[] 內是在描述該詞...

上方這條搜尋語法裡面，定義了 3 個不同的 token：1. `[word=" 把" & pos="P"]` 「把」這個詞（P 介詞，詳見中研院詞類表）2. `[pos="N[abcd].*"]` 名詞（以 `Na/Nb/Nc/Nd` 開頭的詞類）3. `[pos="V.*"]` 動詞（所有以 `V` 開頭的詞類)

語言複雜的程度總是超過我們的想像。事實上，上方的搜尋語法只能搜尋到一**部分的「把」字句**，因為它忽略了其它的可能，例如「**把**寶貴的 **時間 花**在」這個把字句不符...

第 3 章

又是一章

3.1 簡介

接下來，讓我們利用「把」字句的資料進行構式中的搭配分析 (Collostruction Analysis)。以下將會介紹三種量化方式，分別為 Collexeme Analysis (Stefanowitsch & Gries, 2003)、Covarying Collexeme Analysis (Stefanowitsch & Gries, 2005)，以及 Distinctive Collexeme Analysis (Gries & Stefanowitsch, 2004)。

3.1.1 構式與詞彙的互相吸引

Collexeme Analysis 用於衡量構式與其空槽 (lexical slot) 內的詞彙的共現傾向。例如：可以藉由此種分析方式得知什麼詞彙特別容易做為**賓語**出現在「把」字句中。Collexeme Analysis 透過列聯表量化構式與詞彙互相吸引的程度。以表 3.1 為例，透過此列...

表 3.1: 「把字句」(構式) 與「錢」(賓語) 的列聯表

	錢	~錢	Total
把字句	O_{11}	O_{12}	R_1
~把字句	O_{21}	O_{22}	R_2
Total	C_1	C_2	N

要計算出表 3.1 內每個項目的頻率，除了搜尋得來的「把」字句的共現環境 (concordance) 之外，還需要一些額外的資訊。首先，`concordancer` 所找出的把字句筆數，

即是表 3.1 的 R_1 ，而 O_{11} 則是賓語「錢」在這些把字句中出現的頻率。透過 R_1 與 O_{11} ，即可以算出 O_{12}，也就是 R_1 - O_{11}。C_1 則...

```
{
    'L1': { 'o11': freq, 'all': freq },
    'L2': { 'o11': freq, 'all': freq },
    'L3': { 'o11': freq, 'all': freq },
    ...
}
```

3.1.2　構式之中的詞彙搭配：Covarying Collexeme Analysis

衡量同一句式下的兩個 lexical slots 內的詞彙的共現傾向 e.g., 「把」字句中的賓語與動作，如：把時間 (slot1) 花 (slot2) 在...

3.1.3　量化相似構式之間的差異：Distinctive Collexeme Analysis

比較兩種 (or 多種) 句式中，相應位置之 lexical slot 的偏好 e.g., 「把」字句 vs. 「將」字句，句中之動詞使用偏好

3.1.4　Custom Box

自訂區塊標題

這是 **Box** 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。

Markdown content and other blocks are enabled, e.g., code:

```
print("Custom code block")
```

這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。這是 Box 裡面的段落。This is a custom block. This is a custom block. This is a custom block.

附錄 A

R 基礎

R 語言是從統計社群開發出來的語言，比較傑出的地方在於統計相關的計算與製圖。近年來由於幾位知名的語言學家的大力推廣，使 R 語言成為語料庫語言學研究的主流程式語言之一。開源的開發環境 RStudio 的出現，更是使得 R 語言的編程與專案協作，甚至與 web app (shiny) 的開發工作更加便利迅速。R 語言社群對於資料處理的套件開發速度也相當驚人，還有許多進階的統計計算與作圖套件方便研究者從事量化的語音與文本分析。

A.1 向量

R 以向量（vector）當作基本單位，在 Python 中的「一個變數的回傳值」，在 R 中的概念其實是「一個長度為 1 的向量」。而 R 的 vector 又分成 6 種類別，其中較常見的 4 種為 `integer`, `double`, `character`, `logical`。

1. integer vector 的元素由整數組成，可以是零、正數或負數。
    - 除了使用: 製造數列，也可以使用 `c()` (稱為 concatenate) 組出任意序列的 vector。
    - 使用 `c()` 製造 integer vector 時，每個整數數字後面必須接 `L`，若沒有加上 `L`，R 會將製造出來的 vector 視為 double vector。
2. double vector 的元素由浮點數組成，亦即含有小數點的數字 (e.g 1.2, -0.75)
    - double vector 和 integer vector 合稱 numeric vector，通常不太需要區別兩者，因為 R 在運算時，會將這兩種資料類型自動轉換成合適的類型。

3. character vector 的每個元素皆由一個字串組成。

    - R 也可以儲存字串 (string)。在 R 裡面，只要是被引號 (quote, ' 或" 皆可) 包裹的東西就是字串，放在引號內的可以是任何字元 (e.g. 空白、數字、中文字、英文字母)。
    - 如果字串內含有引號 "，需在字串內的引號前使用跳脫字元，以表示此引號是字串的一部分而非字串的開頭或結尾。也可以使用「不同的」引號，弱勢以「單引號」表示字串的開頭與結尾時，字串內就可以直接使用「雙引號」，反之亦然。

4. logical vector 的每個元素由 TRUE 或 FALSE 組成。

    - 可以使用 c() 一項項手動輸入製造 logical vector，另一個來源則是 logical test 的回傳值：
    - boolean operators (&, |, !, any(), all()) 可以整合多個 logical tests
    - logical operators: ==, !=, >, <, %in%

在 R 運算兩個或兩個以上的 vector 時，通常以 element-wise 的方式進行。若 vector 長度不相同，例如 `c(1, 2, 3) + 2`，R 會自動將長度較短 vector (2) 回收 (recycle)，也就是重複此向量內的元素，拉長到與另一個 vector 等長，再將兩個一樣長的 vector 進行 element-wise 的向量運算。vector 內的每個元素，其資料類型 (data type) 必須相同。資料類型即是前面提到的 integer, double, character, logical。若資料類型不一致（例如：將不同資料類型的元素放入 c()），R 會根據某些規則，自動進行資料類型的轉換。

A.1.1　向量中的元素

我們可以透過 3 種方式來取出 vector 裡的元素，傳回一個新的 vector。第一種方式是透過元素在 vector 中的位置次序 (index)：

```
# LETTERS 是 R 內建變數: 一個包含所有大寫英文字母的 character vector

LETTERS[1]
#> [1] "A"

LETTERS[1:5]
#> [1] "A" "B" "C" "D" "E"

```

```r
LETTERS[c(1, 3, 5)]
#> [1] "A" "C" "E"
```

A.2 前處理與 stringr 套件

正如本書第三章介紹在 Python 的文字前處理方式，我們也可以使用 R 來進行。透過各程式語言通用的 Regex 以及 R 的 stringr 套件，我們可以撰寫合適的 function 來清理語料。

我們先來認識 stringr 這個套件。它是 tidyverse 套件中負責處理字串的套件，比起 base R 的字串處理函數，stringr 中的函數名稱較一致（str_*）。在 R 裡面，Regex 是以字串（character）的資料類型來表徵，因此需注意：若是 Regex 包含反斜線，則需要在每一個反斜線之前再加上一個反斜線。舉例來說，在 Python 我們要跳過 . 這個字元時，會寫作 `today\.$`，但在 R 中就必須寫成 `today\\.$`。

`str_detect()` 是一個常用的 stringr 函數。若是我們想知道向量 `s <- c("cat", "bed", "car", "Mr.")` 中，哪些元素擁有字串 a 時，就可以使用 `str_detect(s, "a")`，則程式會回傳一個 logical vector `[1] TRUE FALSE TRUE FALSE`。`str_detect()` 也可結合 [ ] 或 dplyr 套件中的 filter() 函數，來篩選出元素的值：

```r
s <- c("cat", "bed", "car", "Mr.")
s[str_detect(s, "^c")]

#> [1] "cat" "car"
```

A.3 中文文本資料處理

A.3.1 斷詞

jieba 是一個用於中文斷詞的 Python 套件。jiebaR 則是 jieba 的 R 版本。透過 jiebaR 進行斷詞只需要兩個步驟：使用 `worker()` 初始化斷詞設定，再使用 `segment()` 將文字斷詞。`jiebaR::segment()` 會回傳一個 character vector，vector 內的每個元素都是一個被斷出來的詞：

```r
library(jiebaR)
seg <- worker()
```

```
txt <- " 失業的熊讚陪柯文哲看銀翼殺手"
segment(txt, seg)

#> [1] " 失業" " 的熊" " 讚" " 陪" " 柯文" " 哲看" " 銀翼" " 殺手"
```

遇到專有名詞或是特殊詞彙時，jiebaR 的斷詞可能會不太精準。若想避免這種情況，可以新增一份自訂辭典（儲存在一份純文字檔，每個詞佔一行），例如我們的自訂辭典 `user_dict.txt` 的內容如下：

A.3.2　tidytext 套件

至於 tidytext 則是一個較近期的 text mining 套件，它將 tidyverse 的想法運用到文本資料處理，換言之，它使用 data frame 的資料結構去表徵和處理文本資料。若是我們使用 tidytext 處理文本資料，能在文本分析的過程中結合 dplyr 與 ggplot2，快速地視覺化文本資料。

不過在 tidytext framework 之下，文章的內部（詞彙與詞彙之間的）結構會消失，因為一段文本對於 tidytext 來說就是 bag-of-words。由於 tidytext 儲存文本資料的格式是 one-token-per-document-per-row，在一個 data frame 中，每個 row 是一篇文章中的一個 token。舉例來說，如果我們有兩篇文章，第一篇被斷成 38 個詞彙，第二篇被斷成 20 個詞彙，則我們總共需要一個擁有 58 rows 的 data frame 來儲存這兩篇文章。

一般而言，tidytext 的架構適合詞頻相關的分析，像是計算文章的 lexical diversity，或是透過詞頻進行情緒分析。透過 `tidytext::unnest_tokens()`，我們可以將 `docs_df` 儲存的文本資料（已斷詞），變成 tidytext format，也就是 one-token-per-document-per-row 的 data

A.4　向量表徵

由於資料科學以及統計學方法上的限制，我們要對文本進行量化分析之前，常常要將原本以符碼 (文字) 表徵的文本轉成數值表徵，才能對文本進行相似度計算、分群、分類等等分析。將文本轉換成數值的表徵方式相當多，其中一種最簡單的方式，即是使用 document-term matrix 將文本以數值向量去表徵。

```
#' doc1:    "I baked the cake and the muffin"
#' doc2:    "I loved the cake"
```

```
#' doc3:    "I wrote the book"
#' TERMS:        I  baked loved wrote the and cake muffin book
dtm <- matrix(c( 1,   1,    0,    0,   2,  1,   1,    1,   0 ,
                 1,   0,    1,    0,   1,  0,   1,    0,   0 ,
                 1,   0,    0,    1,   1,  0,   0,    0,   1 ),
              nrow = 3, ncol = 9, byrow = TRUE)

dtm
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
#> [1,]    1    1    0    0    2    1    1    1    0
#> [2,]    1    0    1    0    1    0    1    0    0
#> [3,]    1    0    0    1    1    0    0    0    1
```

有了文本的向量表徵之後，我們就能去量化比較文本之間的相似度，方法是直接利用向量之間的距離公式 $d(\vec{p}, \vec{q})$ 以及相似度公式 $cos(\theta)$：

$$d(\vec{p}, \vec{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

$$cos(\theta) = \frac{\vec{p} \cdot \vec{q}}{\|p\|\|q\|}$$

A.4.1 Latent Semantic Anlysis (Dimensionality Reduction)

由於 document-term matrix 通常很稀疏，很多值會是 0，使文本向量無法抓到某些文本之間的語意關聯。比如 doc2 與 doc4 雖然語意相近，但此二文本的向量的相似度 (cosine similarity) 為零，因為這兩篇文本並未使用到相同的詞彙。面對這種情形，我們可以將高維的 document-term matrix 透過數學方式轉換成維度比較小的矩陣。在這個過程中，document-term matrix 中一些語意相近的詞彙會被壓縮到某個或是某些維度中，讓這個維度比較小的矩陣反而比較能表徵文本之間的語意關聯。這種方式稱為 Latent Semantic Analysis (LSA)，而用來將矩陣分解降維的數學方法稱為 Singular Value Decomposition (SVD)。

```
lsa_model <- quanteda.textmodels::textmodel_lsa(q_dfm, nd = 15)
dim(lsa_model$docs)
#> [1] 300  15

# Document similarity
doc_sim2 <- textstat_simil(as.dfm(lsa_model$docs), method = "cosine")
sort(doc_sim2["anti_1.txt", ], decreasing = T)[1:8]
#>   anti_1.txt    pro_2.txt anti_102.txt anti_146.txt   pro_94.txt   pro_84.txt
```

```
#>    1.0000000    0.9963039    0.9959590    0.9936474    0.9933724    0.9933206
#>  anti_94.txt  pro_133.txt
#>    0.9923490    0.9921601
```

參考文獻

Goldberg, A. E. (1995). *Constructions : A construction grammar approach to argument structure*. Chicago: University of Chicago Press.

Gries, S. Th., & Stefanowitsch, A. (2004). Extending collostructional analysis: A corpus-based perspective on ‘alternations' *International Journal of Corpus Linguistics*, *9*(1), 97–129. Journal Article. https://doi.org/10.1075/ijcl.9.1.06gri

Huang, C.-R., Kilgarriff, A., Wu, Y., Chiu, C.-M., Smith, S., Rychlỳ, P., … Chen, K.-J. (2005). Chinese Sketch Engine and the extraction of grammatical collocations. In *Proceedings of the fourth SIGHAN workshop on Chinese language processing*.

Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., … Suchomel, V. (2014). The Sketch Engine: Ten years on. *Lexicography*, *1*(1), 7–36. https://doi.org/10.1007/s40607-014-0009-9

Kilgarriff, A., Rychlý, P., Smrz, P., & Tugwell, D. (2004). The Sketch Engine. In G. Williams & S. Vessier (Eds.), *Proceedings of the 11th EURALEX international congress* (pp. 105–115). Lorient, France: Université de Bretagne-Sud, Faculté des lettres et des sciences humaines.

Stefanowitsch, A., & Gries, S. T. (2005). Covarying collexemes. *Corpus Linguistics and Linguistic Theory*, *1*(1), 1–43. https://doi.org/10.1515/cllt.2005.1.1.1

Stefanowitsch, A., & Gries, S. Th. (2003). Collostructions: Investigating the interaction of words and constructions. *International Journal of Corpus Linguistics*, *8*(2), 209–243. Journal Article. https://doi.org/10.1075/ijcl.8.2.03ste

索引