

Linguistic and python

吳沛馨、翁珮瑜

一、 動機

由於組員都來自中文系，我們有一個共同的經驗是，大一時學習了一整年的語言學。語言學是一門需要觀察及歸納的學科，需要分析大量的語料，有語音、語義、語法和語用這四個不同的大主題。

而語音學中，有很多題目是要尋找語音的變化，觀察某兩個音位的分布環境，根據其前後所接的語音環境，分析出這兩個音位的關係。若是出現在相同環境之中，兩個音位是「獨立音位」，若是出現在不同環境之中，則需要觀察其前後語音的性質，若經過歸納發現，某音位只會出現在特定性質的語音之前或之後，代表這個音位分布範圍特定，說明在特殊情況下，會使原本的音位改為此音位，兩音位稱為「互補分布」。

學習語音學時，我們花大量的時間在看語料上，親手寫出一個個音位的分布環境，寫好後再進行歸納整理，將重複的環境去除，整理好後分析兩個音位之間的關係。大量整理語料分析某兩個音位的互相關係，成為我們那段時間的日常。

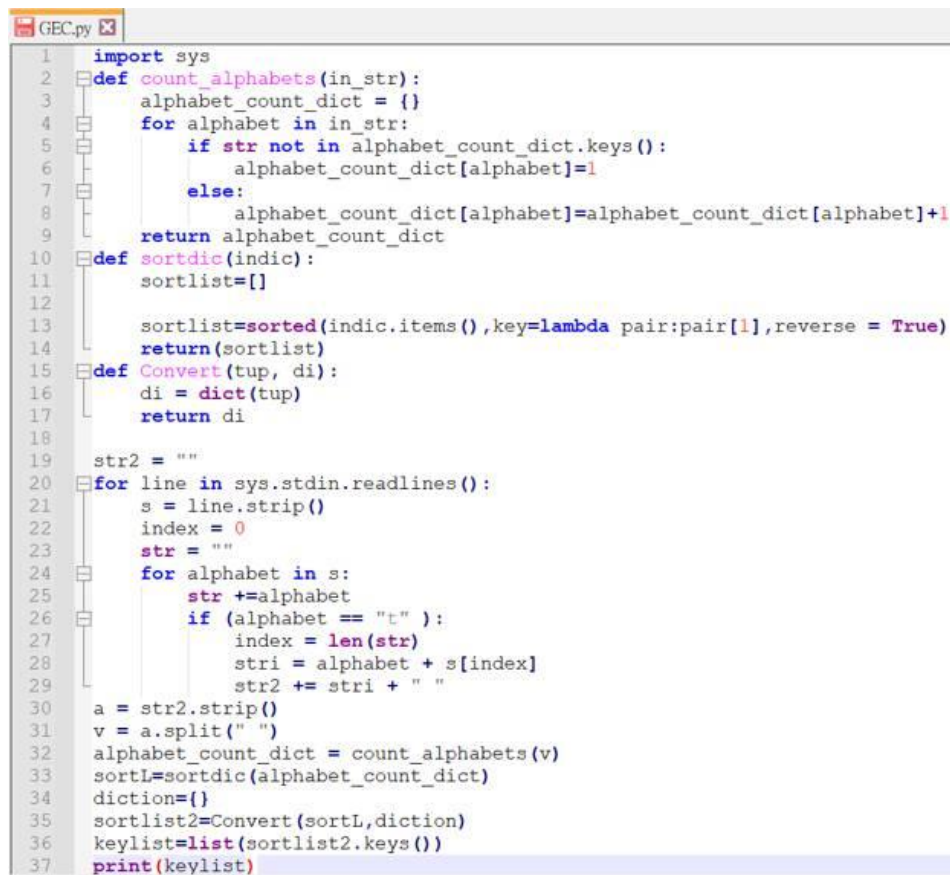
分析、歸納和整理，這些都是語言學的基礎功夫，如果將來想要踏足語言學的領域，觀察、統計、分析等技能不可或缺。因此，在學習到 `python` 的統計功能時，我們便思考如何將它運用在語言學上，簡化我們分析的過程。

由程式統計，給予我們語料的分布環境，我們再進行最後的分析步驟，如此可以節省大量依靠人力整理的時間，是我們研究的初衷。

二、 研究範例題目

從動機上看，耗費時間的理由之一，是有重複的環境分布，因此我們使用在這堂通識課「資訊的邏輯思考」上教過的各種 python 統計功能的知識。首先是 **dictionary** 的技巧，將關鍵字出現的次數進行統計，顯示出各個關鍵字的名稱與次數，再運用 **sort**，讓關鍵字依照次數由大至小一一排列，使版面更清楚簡潔。另外，我們想要的結果只有關鍵字而已，並不需要次數，因此我們另外上網尋找只呈現 **dictionary** 前面的關鍵字，而不要後面的次數的方法。找到這三個步驟的解答，理論上可以解決我們的問題，接下來是實際運行的結果。

(二) 程式碼



```
1 import sys
2 def count_alphabets(in_str):
3     alphabet_count_dict = {}
4     for alphabet in in_str:
5         if str not in alphabet_count_dict.keys():
6             alphabet_count_dict[alphabet]=1
7         else:
8             alphabet_count_dict[alphabet]=alphabet_count_dict[alphabet]+1
9     return alphabet_count_dict
10 def sortdic(indic):
11     sortlist=[]
12
13     sortlist=sorted(indic.items(),key=lambda pair:pair[1],reverse = True)
14     return(sortlist)
15 def Convert(tup, di):
16     di = dict(tup)
17     return di
18
19 str2 = ""
20 for line in sys.stdin.readlines():
21     s = line.strip()
22     index = 0
23     str = ""
24     for alphabet in s:
25         str +=alphabet
26         if (alphabet == "t" ):
27             index = len(str)
28             stri = alphabet + s[index]
29             str2 += stri + " "
30 a = str2.strip()
31 v = a.split(" ")
32 alphabet_count_dict = count_alphabets(v)
33 sortL=sortdic(alphabet_count_dict)
34 diction={}
35 sortlist2=Convert(sortL,diction)
36 keylist=list(sortlist2.keys())
37 print(keylist)
```

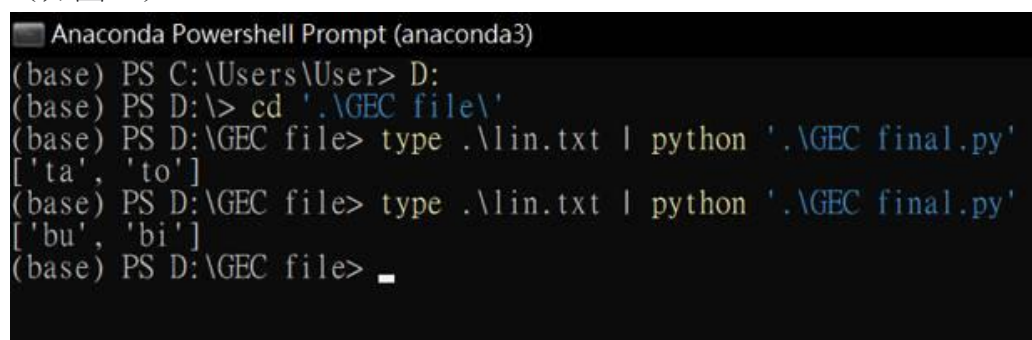
(圖二)

(三) 程式介紹與結果

「工欲善其事，必先利其器」，在程式上依然如此。因此我們將需要重複執行的動作用函式先定義清楚，使程式碼不至於冗長一串，**dictionary** 的用法是讓它可以將讀到的重複環境進行歸類與計算的動作，(圖二第 2 行至第 9 行)。再定義 **sort**，將次數由大到小排序整理(圖二第 10 行至第 14 行)，最後是定義一個函式，可以將排序好的資料，轉換成可以隨意編輯的狀態(圖二第 15 行至第 17 行)，需要這個函式是因為由 **dictionary** 統整出的資料會是無法隨意編輯的 **tuple**，而我們要將 **tuple** 轉成 **list** 的狀態，才可以將後面的次數資料刪除。

將工具都準備好後，開始準備讀取資料（圖二第 20 行），將讀取的資料去掉最後面可能會有的空白（圖二第 21 行），再建立一個參數（圖二第 22 行）與一個空字串（圖二第 23 行），接著根據題目要求建立迴圈，將字母通過迴圈一個一個加進空字串裡，如果讀取到\ɿ這個音位的時候，原有的參數將到此為止的單字長度記錄下來（圖二第 25 行），由於 python 在字串位置的讀取上是從 0 開始，因此，到\ɿ為止的單字長度，會等於\ɿ之後下一個的字母所在位置，設置一個新的變數，是\ɿ加上後一個字母（圖二第 28 行），如此一來，就找到了\ɿ的分布環境（圖二第 29 行）。

找到分布環境後，用事先定義好的函式將環境分類、排序、轉換，最後只取我們所需要的環境分布就好（圖二第 36 行），因此結果會從重複次數多到小排列顯示，接著將原本輸入\ɿ的位置改成\b\（圖二第 26 行），再讓程式從頭開始執行，因為程式碼一模一樣，所以不另外再附上圖片，實際運行後結果為：[ta,to]和[bu,bi]（如下圖三），與當時我們手動進行的結果完全一致（如圖一）。



```
Anaconda Powershell Prompt (anaconda3)
(base) PS C:\Users\User> D:
(base) PS D:\> cd '.\GEC file\'
(base) PS D:\GEC file> type .\lin.txt | python '.\GEC final.py'
['ta', 'to']
(base) PS D:\GEC file> type .\lin.txt | python '.\GEC final.py'
['bu', 'bi']
(base) PS D:\GEC file> _
```

（圖三）

觀察這四個環境，再運用語言學課堂上學習到的知識，可以得知\ɿ和\ɿ都是發聲時舌頭位置較高的高元音（元音即母音），由此可以歸納出，\ɿ只會出現在高元音之前，而\ɿ則出現在其他環境，因此兩音位為互補分布，\ɿ在高元音前會變成\ɿ的發音，在其他情況時仍然維持\ɿ不變。

四、結語

在思考如何解決問題的過程中，我們運用了老師教的方法，也學會上網搜尋資源，將自己的問題一步步剖析後，發現一個問題之中可以再拆分成其他問題。我們所寫出來的程式碼，也是根據從分解出來的源頭問題著手，以我們的問題為例：第一步問題，是如何找到音位後的環境，第二步則是分類，第三步排序，第四步是取分布環境，按部就班的解答每一步驟，就是程式的語言，也是我們在這堂課學習的邏輯思考。

最後，在看到程式結果後，我們只需要從這些結果之中進行分析，節省許多觀察、手寫和人工計算的時間，我們也學到如何使用程式來解決原本需要人工耗時計算的問題，這些重複進行的大量作業，可以用更快捷的工具簡

化負擔，讓研究者將心力投注於資料歸納後的分析上。

五、 工作分配

設計程式：吳沛馨、翁珮瑜

錄製影片：吳沛馨

報告撰寫：翁珮瑜