文本分析與程式設計
Week04

本課程由卓騰語言科技贊助

學習目標

- 可以使用取用特徵詞的工具來分類文本

- 本課程使用的文本來源:
- 1. https://news-taiwan.xyz/uncategorized/39053.html
- 2. https://www.ctwant.com/article/111388

課間練習1：複習

- 請問我們前三週用來取得特徵詞(籃球或棒球類文本)的工具有哪些呢？

實作 -球類競賽報導新聞分類

- 在前三次課程中，我們知道利用棒球和籃球文章的特徵詞，所以知道那些特徵可以分開棒球和籃球。
- 我們使用過的特徵包含:

1. 將所有的詞進行TFIDF的加權運算
2. 抽取名詞進行分析
3. 抽取動詞進行分析
4. 抽取事件進行分析

而在第三周的功課中我們還學到可以透過:

5. 抽取實詞(content words)進行分析

實作目標

- 在這次課程中，我們的目標是可以讓電腦主動分辨哪些文本是屬於籃球，那些是棒球。

- 這部分跟上周的作業有所連結，換句話說，我們把判斷的工作交給電腦，而非人工檢查判斷。例如:上週作業裡，我們請電腦幫我們判斷: 這篇是不是有關股票上漲?

今天就讓我們更深一步來討論這個議題吧!!

實作目標

- 例如我們有以下一篇需要被分類的文本:

- 我們希望電腦回答我們:
- 請問這篇是屬於哪一種球類新聞呢？(以下這篇我們給他的代稱為「未知文本」)
- *金鶯隊左投 John Means 今天在面對水手隊比賽中，完成一項大紀錄，那就是以 27 個出局數，在沒有保送、觸身球、失誤的狀況下完成無安打比賽，而 John Means 差一點就有完全比賽，主要是 3 局下對 Sam Haggerty 投出不死三振，差點就可以完成「完全比賽」，金鶯最終以 6:0 贏球。根據紀錄，金鶯隊上次左投投出無安打比賽已經是 1969 年，也是大聯盟本季第三場無安打比賽，球隊史上第 10 位投出無安打比賽的投手，而他也是第一位在沒有投出保送、安打、失誤，卻投出無安打比賽的投手。*
- 本篇取自於 NewTalk 新聞 https://newtalk.tw/news/view/2021-05-06/570369

課間練習2

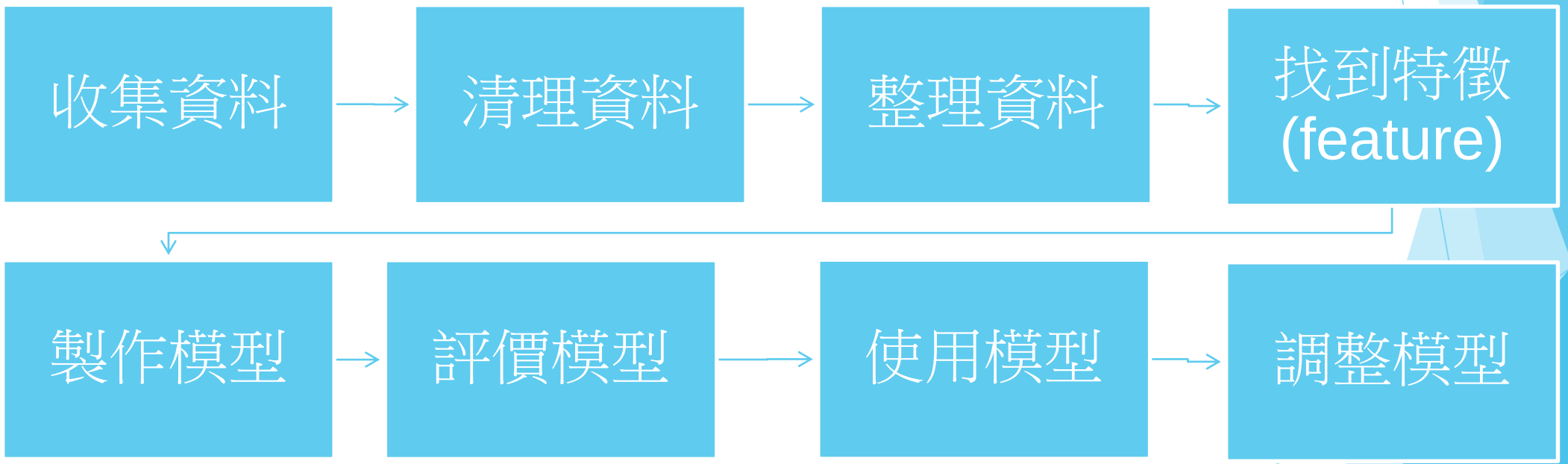
- 那在設計電腦的判斷邏輯之前，我們先看看人是怎麼判斷的。

- 請看剛剛的未知文本，請和同學討論看看這是屬於棒球還是籃球呢？你是怎麼知道的呢？

文本分析流程

文本分析流程

- 通常做文本分析時，我們會有以下的流程
- 這些是什麼意思呢？



文本分析流程

流程	內容	參考工具
收集資料	把資料收集回來	如果要大量收集，關鍵字可以搜尋「網路爬蟲」
整理資料	資料收集回來之後，需要想一下要整理成什麼格式，另外也會分成訓練用和測試用	可以參考python 中有哪些資料格式可以幫助你 ，還有參考將資料隨機分類成訓練用和測試用
清理資料	有些字詞可能對於你的分類沒有助益，例如網址，或是某些標點符號	可以參考 re (regular expression) 的用法
找到特徵	透過不同的工具或是可以透過自己的觀察如何分類	TF-IDF /名詞、動詞/人事時地物

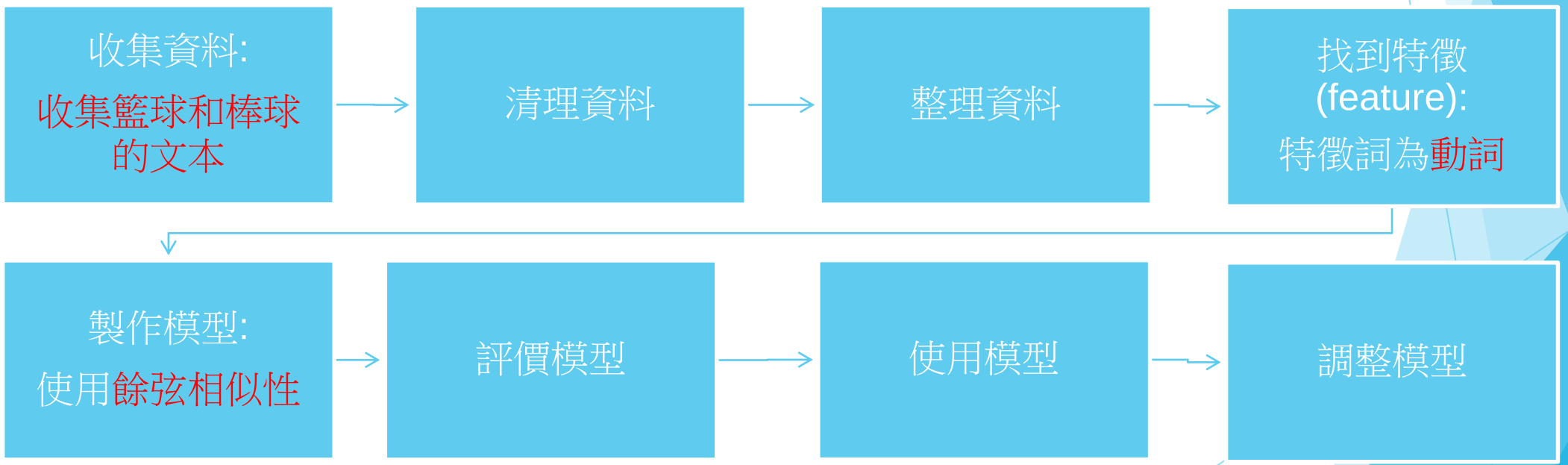
文本分析流程

流程	內容	參考工具
製作模型	就是把模型做出來	模型是一個可以做到我們目的的一套系統
評價模型	可以透過不同的統計模型來檢視自己的模型正確率	
使用模型	接著可以真的使用自己的模型看看，然後解釋模型分類結果	
更新模型	最後如果結果不盡如意，或是未來有新的資料，就需要更新模型	

實作範例

實作範例 - 動詞為特徵詞

- 我們可以透過計算文本中動詞的餘弦相似性，來看看它比較像哪一種文章。
- 比對之前的流程



什麼是餘弦相似性

- 在文本分析中，很常我們會把文字轉換成數字，因為只有數字可以計算
- 所以如果當我們要把文字轉換成數字時，方法就是把文字想個方法變成向量。
- 如此一來就可以利用數學公式來計算他們「相不相似」
- 所以「餘弦相似性」(cosine similarity) 就是計算向量的cosine夾角，夾角越大代表越不像，夾角越小表示越像

- 請參考以下網站： https://clay-atlas.com/blog/2020/03/26/cosine-similarity-text-count/

餘弦相似性限制

- 不過在「解讀」餘弦相似性時也需要特別注意
- 1. 文字到底是怎麼轉成數字？
  - 其實很多時候這麼部分並沒有太多語言上面的解讀，比較多是數學統計後的結果，不過這個統計結果產生出來的「趨勢」以及語言是否都是這樣使用還是有一段差距，需要特別注意。
- 2. 是拿什麼文本去訓練，所以電腦知道什麼字詞會轉呈什麼數字？
  - 文字的確是有規律性，不過要注意不同領域的文章也有屬於那個領域的「規律」，所以資料收集是不是比較「偏頗」，你收集的量大不大也是需要關注的焦點

實作範例 - 動詞為特徵詞

- 接下來將會帶著大家一個步驟一個步驟來操作，並解釋如何利用動詞為特徵詞，來看看兩個文本是否相似

實作範例 - 動詞為特徵詞

- 1.我們利用Week01 時做過的步驟，取出兩篇做為比較基準的「棒球類文本」和「籃球類文本」中的「動詞列表」。
- 這個步驟的重點在於如何取得特徵詞

實作範例 - 動詞為特徵詞

```
username = "" #這裡填入帳號 email
apikey   = "" #這裡填入api Key

articut = Articut(username, apikey)
#以下語料可以參考
baseballSTR = """本週三在紐約的比賽中，馬林魚此戰使用投手車輪戰，4名投手輪番上陣壓制大都會打線，前8局僅被敲出4支安打失1分，讓球隊能帶著2-1的領先優勢進入到9局下半。不過馬林魚推出巴斯登板關門，他面對首名打者麥尼爾，就被打出一發陽春砲，讓大都會追平比數，接下來又分別被敲出2支安打、投出保送，形成滿壘局面，此時輪到康福托上場打擊。在2好1壞的局面下，巴斯投了一顆內角滑球，康福托眼看這顆球越來越靠近自己的身體，似乎有下意識地將手伸進好球帶內，結果這球就直接碰觸到他的身肘，隨後主審庫爾帕判定這是一記觸身球，讓大都會兵不血刃拿下再見分，最終贏得比賽勝利。""".replace(" ", "")
basketballSTR = """昨晚的紐約西區霸王之戰中，錯失勝利的太陽沒有就此束手就擒，延長賽一開始就打出7比2攻勢，米契爾和康利雖然力圖追分，但太陽總能馬上回應。康利讀秒階段上籃得手，布克兩罰一中，再次留給爵士追平機會。米契爾造成犯規，可惜兩罰一中，保羅隨後用兩罰鎖定勝利。米契爾狂轟41分8籃板3助攻，本季單場得分次高；戈貝爾160分18籃板3抄截，波格丹諾維奇20分。康利拿到11分4助攻，克拉克森11分，兩人合計28投僅9中。爵士的三分攻勢難以有效施展，全場44投僅11中。""".replace(" ", "")
```

實作範例 - 動詞為特徵詞

```
#將 KNOWLEDGE_NBA_Teams.json 和 KNOWLEDGE_MLB_Teams.json 兩個體育欸字典讀取出來，
合併成mixDICT 以後，寫入 mixedXICT.json 檔
    with open("ArticutAPI-
master/Public_UserDefinedDict/KNOWLEDGE_NBA_Teams.json", encoding="utf-8")
as f:
        nbaDICT = json.loads(f.read())
    with open("ArticutAPI-
master/Public_UserDefinedDICT/KNOWLEDGE_MLB_Teams.json", encoding="utf-8")
as f:
        mlbDICT = json.loads(f.read())

    mixedDICT = {**nbaDICT, **mlbDICT}
    with open("mixedDICT.json", mode = "w", encoding = "utf-8") as f:
        json.dump(mixedDICT, f, ensure_ascii=False)
```

實作範例 - 動詞為特徵詞

```
## 只取出文字
 def wordExtractor(inputLIST, unify=True):
        '''
        配合 Articut() 的 .getNounStemLIST() 和 .getVerbStemLIST() …等功能，拋棄位置資訊，只抽出詞彙。
        '''
        resultLIST = []
        for i in inputLIST:
            if i == []:
                pass
            else:
                for e in i:
                    resultLIST.append(e[-1])
        if unify == True:
            return sorted(list(set(resultLIST)))
        else:
            return sorted(resultLIST)
```

實作範例 - 動詞為特徵詞

```
# 將baseballStR 和 basketballSTR 兩篇文本各自送入articut.parse() 裡，同時指定 userDefinedDictFILE 為剛才產生mixedDICT.json
baseballResultDICT = articut.parse(baseballSTR,
userDefinedDictFILE="./mixedDICT.json")
basketballResultDICT = articut.parse(basketballSTR,
userDefinedDictFILE="./mixedDICT.json")

print("\n棒球斷詞結果：\n")
pprint(baseballResultDICT)
print("\n籃球斷詞結果：\n")
pprint(basketballResultDICT)
```

斷詞結果範例

```
'result_segmentation': '本/週三/在/紐約/的/比賽/中/，/馬林魚/此戰/使用/投手/車輪戰/，/4名/投手/輪番/上陣/壓制/大都會/打線/，/\n'
                       '/前8局/僅/被/敲出/4支/安打/失/1分/，/讓/球隊/能/帶著/2/-/1/的/領先/優勢/進入/到/9局/下半/。/不過/馬林魚/推出/巴斯/登板/關門/，/\n'
                       '/他/面對/首名/打者/麥尼爾/，/就/被/打出/一發/陽春砲/，/讓/大都會/追平比數/，/接下來/又/分別/被/敲出/2支/安打/、/投出/保送/，/\n'
                       '/形成/滿壘/局面/，/此/時/輪到/康福托/上場/打擊/。/在/2/好/1/壞/的/局面/下/，/巴斯/投了/一顆/內/角滑球/，/康福托/眼/看/這顆/球/越來/越/\n'
                       '/靠近/自己/的/身體/，/似乎/有下/意識地/將/手/伸進/好/球帶內/，/結果/這球/就/直接/碰觸/到/他/的/身肘/，/隨/後/主審/庫爾帕/判定/這/是/\n'
```

實作範例 - 動詞為特徵詞

- 2.我們將一篇「不知其類別」的文本，也用一樣的步驟取出它的「動詞列表」
- 因為我們接下來希望可以使用餘弦相似性的內容，所以我們需要知道訓練用和測試用的文本的特徵詞，這樣才可以拿來比對

實作範例 - 動詞為特徵詞

```
# 取得「動詞」做為特徵列表
## 用動詞取出特徵詞

baseballVerbLIST = articut.getVerbStemLIST(baseballResultDICT)
print("\n getVerbStemLIST 棒球結果")
print(wordExtractor(baseballVerbLIST))

basketballVerbLIST = articut.getVerbStemLIST(basketballResultDICT)
print("\n getVerbStemLIST 籃球結果")
print(wordExtractor(basketballVerbLIST))
```

結果範例

```
 getVerbStemLIST 棒球結果
['上場', '上陣', '伸進', '使用', '保送', '再見', '判定', '到', '壓制', '失', '帶', '形成', '打出', '打擊', '投', '投出', '拿下', '接下來', '推出', '敲', '有下', '看', '碰觸', '讓', '贏得', '越來', '輪到', '進入', '關', '隨', '靠近', '面對']

 getVerbStemLIST 籃球結果
['上籃', '分', '力圖', '助攻', '合計', '回應', '得分', '得手', '打出', '抄截', '投', '拿到', '施展', '比', '犯規', '狂轟', '用', '留給', '罰', '讀秒', '追分', '追平', '造成', '錯失', '鎖定', '開始', '隨']
```

實作範例 - 動詞為特徵詞

```
## 計算未知文本的動詞
unkonwnSTR01 = """
金鶯隊左投 John Means 今天在面對水手隊比賽中，完成一項大紀錄，那就是以 27 個出局數，
在沒有保送、觸身球、失誤的狀況下完成無安打比賽，而 John Means 差一點就有完全比賽，
主要是 3 局下對 Sam Haggerty 投出不死三振，差點就可以完成「完全比賽」，金鶯最終以
6:0 贏球。根據紀錄，金鶯隊上次左投投出無安打比賽已經是 1969 年，也是大聯盟本季第三場
無安打比賽，球隊史上第 10 位投出無安打比賽的投手，而他也是第一位在沒有投出保送、安打、
失誤，卻投出無安打比賽的投手。
"""
unknownResultDICT =
articut.parse(unkonwnSTR01,userDefinedDictFILE="./mixedDICT.json")
unknownVerbLIST = articut.getVerbStemLIST(unknownResultDICT)

print("未知文本動詞:")
print(wordExtractor(unknownVerbLIST, unify = False))
print("\n")
```

結果範例

```
未知文本動詞:
[' John', ' John', ' Sam', 'Haggerty', 'Means', 'Means', '出局數', '史', '大紀錄', '大聯盟', '安打', '投手', '投手', '比賽', '比
賽', '比賽', '比賽', '比賽', '比賽', '比賽', '比賽', '無安打', '無安打', '無安打', '無安打', '無安打', '狀況', '球', '球隊', '第', '紀
錄', '觸身球']
```

實作範例 - 動詞為特徵詞

- 那我們對照一下前面兩個步驟和我們的流程
  - 1) 收集資料：收集了籃球和棒球的資料 (以此範例，我們找了三篇)
  - 2) 整理資料：我們分成訓練用資料(一篇籃球，一篇棒球)和測試用資料 (一篇不知道是哪一種的運動新聞)
  - 3) 清理資料： 我們只想取用動詞，所以其他字詞不考慮
  - 4) 找到特徵：從訓練用資料和測試用資料都用 `.getVerbStemLIST()` 找到動詞特徵詞

實作範例 - 動詞為特徵詞

- 3.接下來，利用 `Counter()` 模組將列表中的每個動詞出現的次數，各自累加起來。再用 `counterCosinSimilarity()` 函式計算 [棒球類文本 vs. 未知文本」的餘弦相似度，以及 [籃球類文本 vs. 未知文本] 的餘弦相似度。
- `Counter()`是需要`import` 的套件
- `counterCosinSimilarity()` 是我們自己寫的 function

實作範例 - 動詞為特徵詞

```
## 利用 Counter() 模組計算每個動詞出現的次數

from collections import Counter

baseballCOUNT = Counter(wordExtractor(baseballVerbLIST, unify=False))
basketballCOUNT = Counter(wordExtractor(basketballVerbLIST,
unify=False))
unknownCOUNT = Counter(wordExtractor(unknownVerbLIST, unify=False))

print("棒球動詞次數")
print(baseballCOUNT)
print("籃球動詞次數")
print(basketballCOUNT)
print("未知文本動詞次數")
print(unknownCOUNT)
```

實作範例 - 動詞為特徵詞

結果範例

棒球動詞次數
Counter({'安打': 2, '局面': 2, '投手': 2, '比賽': 2, '主審': 1, '優勢': 1, '分': 1, '勝利': 1, '意識地': 1, '手': 1, '打線': 1, '打者': 1, '此戰': 1, '滿壘': 1, '球': 1, '球隊': 1, '登板': 1, '眼': 1, '角滑球': 1, '觸身球': 1, '身肘': 1, '身體': 1, '車輪戰': 1, '追平比數': 1, '這球': 1, '陽春砲': 1, '領先': 1})
籃球動詞次數
Counter({'中': 2, '勝利': 2, '攻勢': 2, '籃板': 2, '人': 1, '單場': 1, '場': 1, '布克': 1, '延長賽': 1, '戰': 1, '機會': 1, '次': 1, '波格丹諾維奇': 1, '階段': 1, '霸王': 1})
未知文本動詞次數
Counter({'比賽': 8, '無安打': 5, ' John': 2, 'Means': 2, '投手': 2, ' Sam': 1, 'Haggerty': 1, '出局數': 1, '史': 1, '大紀錄': 1, '大聯盟': 1, '安打': 1, '狀況': 1, '球': 1, '球隊': 1, '第': 1, '紀錄': 1, '觸身球': 1})

實作範例 - 動詞為特徵詞

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

```
def counterCosineSimilarity(counter01, counter02):
    '''
    計算 counter01 和 counter02 兩者的餘弦相似度
    '''
    terms = set(counter01).union(counter02)
    #將兩個dictionary的keys合併出來成set的格式
    dotprod = sum(counter01.get(k, 0) * counter02.get(k, 0) for k in terms)
    #將兩個文章傳換成由數字表示的LIST，數值表示不同字的頻率，然後做點積
    magA = math.sqrt(sum(counter01.get(k, 0)**2 for k in terms)) #歐式距離
    magB = math.sqrt(sum(counter02.get(k, 0)**2 for k in terms)) #歐式距離
    return dotprod / (magA * magB)
```

實作範例 - 動詞為特徵詞

```
def lengthSimilarity(counter01, counter02):
    '''
    計算 counter01 和 counter02 兩者在長度上的相似度
    '''

    lenc1 = sum(iter(counter01.values()))
    lenc2 = sum(iter(counter02.values()))
    return min(lenc1, lenc2) / float(max(lenc1, lenc2))
```

越接近1表示長度越相似

實作範例 - 動詞為特徵詞

```
# 計算 [棒球文本 vs. 未知文本] 的餘弦相似度；計算 [籃球文本 vs. 未知文本] 的餘弦相似度；
baseball2unknownSIM = counterCosineSimilarity(baseballCOUNT, unknownCOUNT)
basketball2unknownSIM = counterCosineSimilarity(basketballCOUNT, unknownCOUNT)

print("[棒球文本 vs. 未知文本] 的動詞餘弦相似度:{}".format(baseball2unknownSIM))
print("[籃球文本 vs. 未知文本] 的動詞餘弦相似度:{}".format(basketball2unknownSIM))
```

```
[棒球文本 vs. 未知文本] 的動詞餘弦相似度:0.3749343922215396
[籃球文本 vs. 未知文本] 的動詞餘弦相似度:0.0
```

越接近1表示長度越相似

實作範例 - 動詞為特徵詞

- 這個方法就是要來使用我們的模型了。我們希望可以用Cosine Similarity 來比對兩個文本中動詞的次數出現是否相似
- 如果說兩個文本他們使用的動詞頻率很接近，例如一篇使用最多的動詞是「投出」，另外一篇使用最高的也是「投出」，那就代表他們可能很相似

實作範例 - 動詞為特徵詞

- 將步驟三比對我們的流程就是「製作模型 」這個步驟
- 我們這邊是直接選用餘弦相似性這個方法來幫助我們看看兩個文本是否相似

實作範例 - 動詞為特徵詞

- 得到的值如下：

```
[棒球文本 vs. 未知文本] 的動詞餘弦相似度:0.3749343922215396
[籃球文本 vs. 未知文本] 的動詞餘弦相似度:0.0
```

- 這表示 [棒球文本] 和 [未知文本] 之間的相似度，比 [籃球文本] 和 [未知文本] 之間的相似度來得高。

課間練習3

- 請問這個結果和你們自己討論出來的結果一致嗎？
- 請問你也是用動詞來評判一個這個新的文本是不是棒球或是籃球的文本嗎？

實作範例 - 動詞為特徵詞

- 課間練習3 的討論其實就是某種評價模型的開始
- 在得到結果之後，我們可以看看這個評價是不是對的
- 因為目前我們文本量非常的少，所以還可以用「肉眼」看是不是我們人類自己判斷，也會把他判斷為棒球文本
- 如果文本量很大，就需要使用不同的統計方式來驗證你的模型判斷的正不正確。有一個簡單的方式就是計算 混淆矩陣(confusion matrix)

Confusion matrix混淆矩陣

- 混淆矩證就是用來比較
  - 1. 人類自己的判斷結果
  - 2. 電腦的判斷結果
- 例如我們判斷未知文本是不是棒球文本
- 交叉比對人類和電腦的結果，會有以下四種可能

Confusion matrix混淆矩陣

- True Positive: 人類和電腦都覺得是。
- True Negative: 人類和電腦都覺得不是
- 我們以人類判斷為正確的判斷的話，這兩種可能性就代表電腦也判斷對了

	人類判斷是	人類判斷不是
電腦判斷是	True positive	False Positive
電腦判斷不是	False Negative	True Negative

Confusion matrix混淆矩陣

- False Positive: 人類判斷不是但是電腦判斷是
- False Negative: 人類判斷是，但電腦判斷不是
- 以上兩種情況就是電腦搞錯了
- 我們可以計算以上這四種情況的個數，來看看這個模型的預測的情況

	人類判斷是	人類判斷不是
電腦判斷是	True positive	False Positive
電腦判斷不是	False Negative	True Negative

Confusion matrix混淆矩陣

- 如果知道以上這四種數據的多寡，就可以知道我們的模型是否準確
- 所以我們就可以看：
  - 精準率 (accuracy)
  - 精準率 (precision)
  - 召回率 (recall)
  - F1-score

	人類判斷是	人類判斷不是
電腦判斷是	True positive	False Positive
電腦判斷不是	False Negative	True Negative

Confusion matrix混淆矩陣

- 精準率 (accuracy)
  - 這個模型可以預測多少東西正確
  - 算法 (TP + TN)/全部
- 精確率 (precision)
  - 在電腦判斷正確的情況下，判斷「是棒球文本」占多少比率
  - TP / (TP + FP)

	人類判斷是	人類判斷不是
電腦判斷是	True positive (TP)	False Positive (FP)
電腦判斷不是	False Negative (FN)	True Negative (TN)

Confusion matrix混淆矩陣

- 召回率 (recall)
  - 在人類判斷「是棒球文本」的數量中，那些是電腦判斷「是棒球文本」
  - TP / (TP + FN)
- F1-score
  - 是一個兼顧 recall 和 precision 的計算方法

	人類判斷是	人類判斷不是
電腦判斷是	True positive (TP)	False Positive (FP)
電腦判斷不是	False Negative (FN)	True Negative (TN)

Confusion matrix混淆矩陣

- 從上述解釋會發現recall 和 precision 看事情的角度不太一樣
- 所以recall 和 precision 的比率會受到「電腦判斷錯誤數量」的影響。所以當在看 recall 或是 precision 我們就會想說要不要降低FN 或是 FP 的量。

	人類判斷是	人類判斷不是
電腦判斷是	True positive (TP)	False Positive (FP)
電腦判斷不是	False Negative (FN)	True Negative (TN)

Confusion matrix混淆矩陣

- 那因為recall 和 precision 其實只是看到其中一個角度，如果使用F1-score 就可以有一個比較全觀的數值來觀察

	人類判斷是	人類判斷不是
電腦判斷是	True positive	False Positive
電腦判斷不是	False Negative	True Negative

Confusion matrix混淆矩陣延伸閱讀

- 如果想要更了解以上內容可以讀
- 1.如何辨別機器學習模型的好壞？秒懂Confusion Matrix
  https://www.ycc.idv.tw/confusion-matrix.html
- 2.心理學和機器學習中的 Accuracy、Precision、Recall Rate 和 Confusion Matrix
  https://chingtien.medium.com/%E5%BF%83%E7%90%86%E5%AD%B8%E5%92%8C%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E4%B8%AD%E7%9A%84-accuracy-precision-recall-rate-%E5%92%8C-confusion-matrix-529d18abc3a
- Day 11 - Confusion Matrix 混淆矩陣-模型的好壞 (1)
  https://ithelp.ithome.com.tw/articles/10254593
- Day 12 - Confusion Matrix 混淆矩陣-模型的好壞 (2)
  https://ithelp.ithome.com.tw/articles/10254671

實作範例 - 動詞為特徵詞

- 人類自己比較出來的結果和電腦計算後的結果來做比較，這個就是評價我們的模型
- 如果你多重複幾次看看不同的文本然後發現結果沒有達到你的理想，那麼這樣就需要做到流程圖所說的調整模型和更新模型

實作範例 - 動詞為特徵詞

- 因為我們是使用「動詞」來計算的，因此我們可以將這次的分類結果解釋為：「未知文本中，描述發生什麼事件使用的動詞，和棒球文本相比，較為相似。」

實作範例 - 名詞為特徵詞

- 同樣的步驟，除了在「動詞」上操作以外，我們也能在「名詞」上依樣畫葫蘆。

課間練習4

- 1. 請依照以下步驟，來以名詞為特徵來看看棒球和籃球文本的相似程度
  - 1) 取出做為基準文本的「棒球類文本」和「籃球類文本」的「名詞列表」
  - 2) 用一樣的方法取出「未知文本」的名詞列表
  - 3)利用 Counter() 模組將列表中的每個名詞出現的次數，各自累加起來。再用 counterCosinSimilarity() 函式計算[棒球類文本 vs. 未知文本」的名詞餘弦相似度，以及[籃球類文本 vs. 未知文本] 的名詞餘弦相似度。

課間練習4

- 2. 請問透過上面步驟，未知文本是哪一種文本呢？
- 3. 你覺得透過「名詞」和「動詞」可以來分類文本嗎？

課間練習5

- 1. 請參考利用名詞和動詞來當作特徵的文本分類步驟，使用計算TF-IDF 為特徵，來比對未知文本和棒球還是籃球的餘弦相似性。
- 2. 請問利用TF-IDF 可以告訴你未知文本與哪一種文本比較相似嗎？

課間練習6

- 目前你已經有用「動詞」、「名詞」以及「TF-IDF」所得到的特徵詞和未知文本和棒球文本及籃球文本比對而得出的餘弦相似性。請問看到目前電腦給你的分析成果，哪一種你覺得比較好解釋「為什麼未知文本和棒球文本比較像」？

分析成果

- 我們可以比較我們會怎麼解釋從三種不同特徵詞得出的

用動詞當特徵	用名詞當特徵	用TF-IDF當特徵
未知文本中，描述發生什麼事件使用的**動詞**，和棒球文本相比，較為相似。	未知文本中，涉及的**物體或人物**，和棒球文本相比，較為相似。	文本特徵很像

分析成果

- 從上述的根據不同特徵詞解釋的比較，我們可以發現
- 1.因為**詞性有其解釋性**，我們知道「動詞」代表著涉及的事件、「名詞」代表著事件中的物體或人物。因此我們做出來的結果也具有解釋性。
  - 詞性有解釋性是因為我們知道「詞性」背後是代表什麼意思。例如動詞是一個描述動作的總類。而名詞大多是代表人物和物品。所以如果一篇文章中他們使用類似數量的動詞和名詞，應該就可以說這兩篇的本質比較相似

分析成果

- 從上述的根據不同特徵詞解釋的比較，我們可以發現
- 2.利用 TF-IDF 來做分類，一樣有效果。但是如果要解釋究竟「未知文本」和「棒球文本」之間「什麼東西很相似？」我們只能說「文本特徵很像」，而無法像前面的例子中所說明的「它們描述的事件很像」或是「它們涉及的物體/人名」很相似。
  - 因為TF-IDF 只是把每篇文章最特殊的地方選出來，但這個選出機制是頻率來計算，如同在第三週的討論中，我們會發現頻率並非完全是人類判斷文本種類的依準。

分析成果

- 從上述的根據不同特徵詞解釋的比較，我們可以發現
- 3.利用詞性 (動詞/名詞) 做出的抽詞技術，我們可以用很少量的資料就做出文本分類模型。
  - 其實一般在做文本分析訓練時，僅用一篇是不太夠的，因為資料量太少。所以通常都會用「大量」的文本。通常如果做一個學術研究用上幾百篇的新聞，可能都不太算大量的資料。可能要到幾千或是幾萬篇才可能明確的相似度比較。不過目前使用三篇就可以有這樣的成果。

課間練習7

- 1. 仿照前例，請在網路上找到十篇籃球比賽報導，十篇棒球比賽報導以及十篇「非」籃球亦「非」棒球的比賽報告，試試看透過 [名詞] 、[動詞] 或其它特徵詞抽取方式來分類。
- 2. 試著解釋你的分類依據。
- 3. 請思考，若做為分類基準的文本和測試的文本長度相差過大時，是否會造成分類效果的影響？該如何調整？

作業:個人Project設計

- 最後一週希望大家都能夠有所收穫，複習一下我們至今所學習的工具們:

斷詞工具:

- `resultDICT = articut.parse(inputSTR)`
- 在這個功能裏頭，我們可以調參數成`lv2` 或 `lv3`，來進行不同細緻程度的斷詞分析。
- 我們也可以在其中加入自定義的辭典來處理一些比較不好斷詞的專有名詞。

作業:個人Project設計

- Articut lv2 回傳的字典檔，我們可以做以下不同的分析:

找出每個字詞的值TFIDF

- `articut.analyse.extract_tags(ResultDICT)`

找出名詞

- `articut.getNounStemLIST(ResultDICT)`

找出動詞

- `articut.getNounStemLIST(ResultDICT)`

找出實詞

- `articut.getContentWordLIST(ResultDICT)`

作業:個人Project設計

- Articut lv2 回傳的字典檔，我們可以做以下不同的分析:

地點

- `articut.getLocationStemLIST(ResultDICT)`

人名

- `articut.getPersonLIST(ResultDICT)`

作業:個人Project設計

- 也可以透過lv3的回傳結果分析:

動詞事件

- articut.parse(baseballSTR, level = “lv3”)[“event”]

時間

- articut.parse(baseballSTR, level = “lv3”)[“time”])

作業:個人Project設計

- 而透過這些工具所得到的結果，我們很多不同的分析方式，像是之前學到的文字雲，功課中計算股市為漲的分數的方式，又或是今天所學算餘弦相似度，這些分析方法都只是冰山一角，而這些方法要有效的前提是要可靠的斷詞結果，還有充分的詞性知識，我們才能順利找出關鍵的切入點。

作業:個人Project設計

- 在這次的project中，希望同學們達到的條件:
- 1. 選用中文文本
- 2. 請蒐集起碼兩中種類的文本起碼10篇，選文本總數的80%作為你的訓練集，透過對文本的觀察，請你做出自己的分類器。
- 3. 將剩下的20%的文本當作測試集，看看你訓練出來的分類器能不能正確區分出這些文本，並計算f1 score與準確率。

加油!