文本分析與程式設計

Week 1

學習目標

- 了解文本分析的定義和應用。
- 了解什麼是對電腦來說是「字」、「字符」、「詞」。
- 學習使用 ArticutAPI 斷詞系統。

本課程使用的文本：

1. https://news-taiwan.xyz/uncategorized/39053.html
2. https://www.ctwant.com/article/111388 (有經過編輯)

什麼是「文本分析」？

什麼是語言？

- 我們需要先知道語言和文字的不同。
- 語言 = 語音 + 意義 + 結構。
- 一般而言，語言可以理解成我們所說出的話，所以有包括我們說話所發聲的聲音，這一串聲音是由句法結構的，說出的聲音也組建成不同的意思。

什麼是文本？

- 文字 = 語言的記號
- 文本是指所有以「文字」為記號來呈現「語言」的內容，例如我們用電腦打字打下來的內容。

什麼是文本？

- 文本分析的意思就是指我們使用程式語言來幫我們解構文本中的內容，方便我們做事後的「分析」。
- 分析的內容可以很廣泛，甚至可以說只要是文字相關都可以算是某種文字分析，從簡單的頻率計算到如何讓電腦自動幫我們分辨文本類別。
- 例如讓電腦從多篇新聞中去學會分類某個文本是屬於運動新聞還是政治新聞，就是文本分類的其中一個例子。

文本分析常見應用

- 文本分析是屬於自然語言處理 (Natural Language Processing, NLP) 的一部分，指處理文字方面的相關工作。
  - 自然語言處理是人工智慧底下的一個子領域。
- 黃色框框的是常見的文本分析任務

文本分析常見應用解析

- 資訊檢索：從大量之中找到關鍵詞或關鍵事實。
- 文本探勘：處理大量資料以發現文本中的趨勢或是隱藏的訊息。
- 語意分析：讓電腦整理出段落或是文章大意。
- 意圖分析：讓電腦整理出作者可能背後意圖。
- 機器翻譯 ：由機器翻譯不同語言的內容。
- 文本分類：利用電腦將大量資料依照文本內容分類。

文本分析常見應用解析

- 文本生成：給機器一部分資料之後，可以生成更多文字，有點像是電腦自己寫文章。
- 情緒分析：將不同的文章、語句或是字詞依照不同情緒分類。
- 自動語音辨識：電腦讀取文字之後可以轉成語音，或是收到的語音轉成文字。
- 自動語音合成：用人工方式合成語音。

文本分析第一步：斷詞

- 文本分析的第一步是要先讓電腦可以先知道什麼是字。
- 唯有如此，這樣電腦才能先知道怎麼樣讀懂句子和文章。

什麼是字？

- 對電腦來說，怎麼樣是一個「字」呢？
- 如果電腦得到以下字串：「啊我朋友就認識一個家裡養了綿羊的小朋友 」，電腦會將這些字詞做以下分類：
  - 字 (word): 句子裡的獨立意義段落，例如「朋友」。
  - 字符 (character): 字碼表裡的獨立符號，例如「朋」這個字。
  - 詞 (phrase): 字+詞綴(構詞/句法) ，例如「養了」。
  - 符記 (token): 自定切分規格後的結果。
  - 詞彙 (lexicon): 字典中列出的獨立項目。
  - 詞條 (entry): 資料庫中列出的獨立項目。

如果想要了解更多關於NLP 或文字分析可以閱讀這篇文章： http://bit.ly/nlp-sinica。

課間練習1

- 請問下面這段文字有幾個字？幾個字符？幾個詞？

「這個星期日本想往後山藥師佛寺去世人罕至處想一想自己的人生」

認識斷詞系統

ArticutAPI 套件介紹

- Articut 是一套「純台灣製造」的中文 NLP 系統。
- 它能同時處理中文斷詞、詞性標記以及命名實體標記的套件
- 相較於 Jieba 分詞、Stanford CoreNLP 以及中研院的 CKIP 或是其它基於簡體字的語料訓練出來的 HanLP、哈工大 LTP…等等方案，Articut 具有功能完備、應用靈活及對新詞的接受度高且更新迅速的特點。

ArticutAPI 套件介紹

- Articut 需在 Python3.6 以上的環境中運作！
- 本週課程的完整程式碼可在以下網址取得：https://github.com/Droidtown/NLP_Training/blob/main/Unit01/
- 如果想看 jupyter notebook 的範例，請至以下網址：https://plusdscp.csie.ntnu.edu.tw/index.php/demonstration/material/

安裝 ArticutAPI

- ArticutAPI 可透過 Python 的套件系統 (pip) 進行安裝並操作 Articut的 API 介面。
- 只要在電腦裡下以下指令擇一使用，即可安裝完成

```
pip3 install ArticutAPI
```

```
python3 -m pip install ArticutAPI
```

安裝 ArticutAPI

- 在電腦下指令有兩種方法
  - 一種是可以開 CMD (命令提示字元，windows) ，或是在 linux/mac 的 terminal 內輸入。
  - 如果是使用 Anaconda 內的 jupyter notebook，可以在 jupyter notebook 中直接執行 !pip3 install ArticutAPI 或是開啟anaconda prompt 中開啟使用。

註冊並取得教學用 API 金鑰

卓騰語言科技免費提供一個月無字數限制之 Articut 教學用金鑰給中華民國教育部承認之教學單位授課使用。請先至 https://api.droidtown.co 完成註冊，並來信 info@droidtown.co 載明：

- 授課教師姓名
- 授課教師 email (必需和前述註冊帳號一致)
- 授課大綱
- 經查證後，即可取得「一個月」無字數限制之 Articut 教學用金鑰。 再登入 https://api.droidtown.co 後，即能在以下畫面取得 Articut NLP 系統的操作金鑰。
- 金鑰可分給課堂學生使用。

基本操作範例：輸入-語法講解

```
from ArticutAPI import Articut #呼叫ArticutAPI套件
from pprint import pprint      #載入 pprint相關套件


if __name__ == "__main__":
    username = "" #這裡您註冊之droidtown email
    apikey   = "" #這裡您註冊之droidtown 後所得到的 api key
    articut = Articut(username, apikey)

    inputSTR = "會被大家盯上，才證明你有實力" #輸入中文字串
    resultDICT = articut.parse(inputSTR) #將結果存在 resultDICT 的變數中
    pprint(resultDICT)
```

基本操作範例：輸出

執行結果

```
{ 'document': 'https://api.droidtown.co/document/',
  'exec_time': 0.09083318710327148, #將剛剛輸入之中文斷詞需要之時間
  'level': 'lv2', #使用哪一種程度的斷詞
  'msg': 'Success!',
  'product': 'https://api.droidtown.co/product/',
  'result_obj': [[{'pos': 'MODAL', 'text': '會'},
                  {'pos': 'ACTION_lightVerb', 'text': '被'},
                  {'pos': 'ENTITY_nouny', 'text': '大家'},
                  {'pos': 'ACTION_verb', 'text': '盯'},
                  {'pos': 'RANGE_locality', 'text': '上'}],
                 [{'pos': 'PUNCTUATION', 'text': '、'}],
                 [{'pos': 'MODAL', 'text': '才'},
                  {'pos': 'ACTION_verb', 'text': '證明'},
                  {'pos': 'ENTITY_pronoun', 'text': '你'},
                  {'pos': 'ACTION_verb', 'text': '有'},
                  {'pos': 'ENTITY_noun', 'text': '實力'}]],
  'result'['<MODAL>會</MODAL><ACTION_lightVerb>被</ACTION_lightVerb><ENTITY_nouny>大家</ENTITY_nouny>
                   <ACTION_verb>盯</ACTION_verb><RANGE_locality>上</RANGE_locality>',
                  '、',
                   <MODAL>才</MODAL><ACTION_verb>證明</ACTION_verb><ENTITY_pronoun>你</ENTITY_pronoun><ACTION_verb>有
                   </ACTION_verb><ENTITY_noun>實力</ENTITY_noun>'],
   'result_segmentation': '會/被/大家/盯/上/、/才/證明/你/有/實力',        #僅顯示斷詞結果
   'status': True,
   'version': 'v235',
   'word_count_balance': 263}
```

#斷詞後是屬於哪種詞性(pos) 以及對應的字詞(text)

#將詞性及字詞放在同一句表示

進階用法之一：lv1 和 lv2 輸入

- Articut 可以針對不同的需求，來自由選擇斷詞的細緻程度。
- Articut Level 意指斷詞的深度。數字愈小，切得愈細。lv1 比較細一些。
- 使用的方法就是在呼叫 `articut.parse()` 不只放入想斷詞的中文字串，也要設定 `level` 參數。

比如說「小紅帽」

- 在 lv1 的設定下，會回傳為 [小/紅/帽]。
- 在 lv2 的設定下，則會回傳 [小紅帽]。

```
from ArticutAPI import Articut

if __name__ == "__main__":
    username = ""
    apikey   = ""
    articut = Articut(username, apikey)

    inputSTR = "小紅帽"
    resultDICT = articut.parse(inputSTR, level='lv1')
    print("\n lv1 的設定下，處理結果：\n", resultDICT)

    resultDICT = articut.parse(inputSTR, level='lv2')
    print("\n lv1 的設定下，處理結果：\n", resultDICT)
```

進階用法之一：lv1 和 lv2 輸入

執行結果

lv1 的設定下，處理結果：


```
{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.03942060470581055,
 'level': 'lv1',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'MODIFIER', 'text': '小'},
                 {'pos': 'MODIFIER_color', 'text': '紅'},
                 {'pos': 'ENTITY_nounHead', 'text': '帽'}]],
 'result_pos': ['<MODIFIER>小</MODIFIER><MODIFIER_color>紅</MODIFIER_color><ENTITY_nounHead>帽</ENTITY_nounHead>'],
 'result_segmentation': 小/紅/帽',
 'status': True,
 'version': 'v236',
 'word_count_balance': 1933}
```


lv2 的設定下，處理結果：


```
{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.04227423667907715,
 'level': 'lv2',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'ENTITY_nouny', 'text': '小紅帽'}]],
 'result_pos': ['<ENTITY_nouny>小紅帽</ENTITY_nouny>'],
 'result_segmentation': 小紅帽',
 'status': True,
 'version': 'v236',
 'word_count_balance': 1930}
```

進階用法之一：lv1 和 lv2

- 同理，在 lv1 下會把動詞和時態標記分開，因此「創造了」會被切分成「創造/了」；
  但在 lv2 的設定下，則會把動詞和時態標記結合在一起，因此「創造了」將在 lv2 處理為「創造了」。

進階用法之一：lv1 和 lv2

- 這是因為 Articut 將時態標記「了」視為像是英文裡的 -ed。因此，在 lv1 時，採取將之處理為 "create/-ed" 分開的兩個元素，但在 lv2 的設定下，則是以 "created" 這種「詞 + 時態標記」的形式輸出。

執行結果

lv1 的設定下，處理結果：


```
{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.03736257553100586,
 'level': 'lv1',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'ACTION_verb', 'text': '創造'},
                 {'pos': 'ASPECT', 'text': '了'}]],
 'result_pos': ['<ACTION_verb>創造</ACTION_verb><ASPECT>了</ASPECT>'],
 'result_segmentation': '創造/了',
 'status': True,
 'version': 'v236',
 'word_count_balance': 345}
```


lv2 的設定下，處理結果：

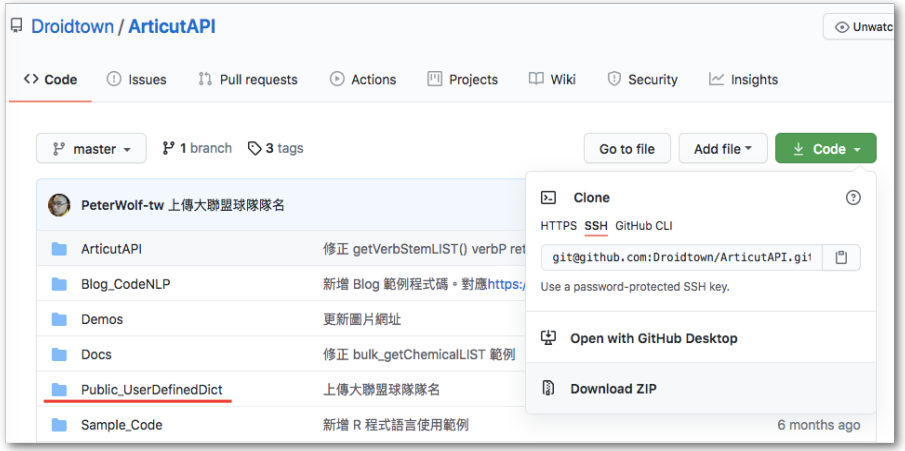

```
{'exec_time': 0.08351516723632812, 'result_pos': ['<MODAL>會</MODAL><ACTION_lightVerb>被</ACTION_lightVerb><ENTITY_noun>大家</ENTITY_noun><ACTION_verb>盯
</ACTION_verb><RANGE_locality>上</RANGE_locality>', '，', '<MODAL>才</MODAL><ACTION_verb>證明</ACTION_verb><ENTITY_pronoun>你</ENTITY_pronoun><ACTION_verb>有
</ACTION_verb><ENTITY_noun>實力</ENTITY_noun>'], 'result_segmentation': '會/被/大家/盯/上/，/才/證明/你/有/實力', 'result_obj': [[{'text': '會', 'pos': 'MODAL'},
{'text': '被', 'pos': 'ACTION_lightVerb'}, {'text': '大家', 'pos': 'ENTITY_noun'}, {'text': '盯', 'pos': 'ACTION_verb'}, {'text': '上', 'pos':
'RANGE_locality'}], [{'text': '，', 'pos': 'PUNCTUATION'}], [{'text': '才', 'pos': 'MODAL'}, {'text': '證明', 'pos': 'ACTION_verb'}, {'text': '你', 'pos':
'ENTITY_pronoun'}, {'text': '有', 'pos': 'ACTION_verb'}, {'text': '實力', 'pos': 'ENTITY_noun'}]], 'level': 'lv2', 'version': 'v236', 'status': True, 'msg':
'Success!', 'word_count_balance': 354, 'product': 'https://api.droidtown.co/product/', 'document': 'https://api.droidtown.co/document/'}
{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.04187440872192383,
 'level': 'lv2',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'VerbP', 'text': '創造了'}]],
 'result_pos': ['<VerbP>創造了</VerbP>'],
 'result_segmentation': '創造了',
 'status': True,
 'version': 'v236',
 'word_count_balance': 342}
```

課間練習2

- 仿照前例，用 Python3 設計一段程式，分別用 "lv1" 和 "lv2" 輸入「閱讀創造了奇蹟」，觀察 lv1 和 lv2 的設定下，回傳的結果有何差別。

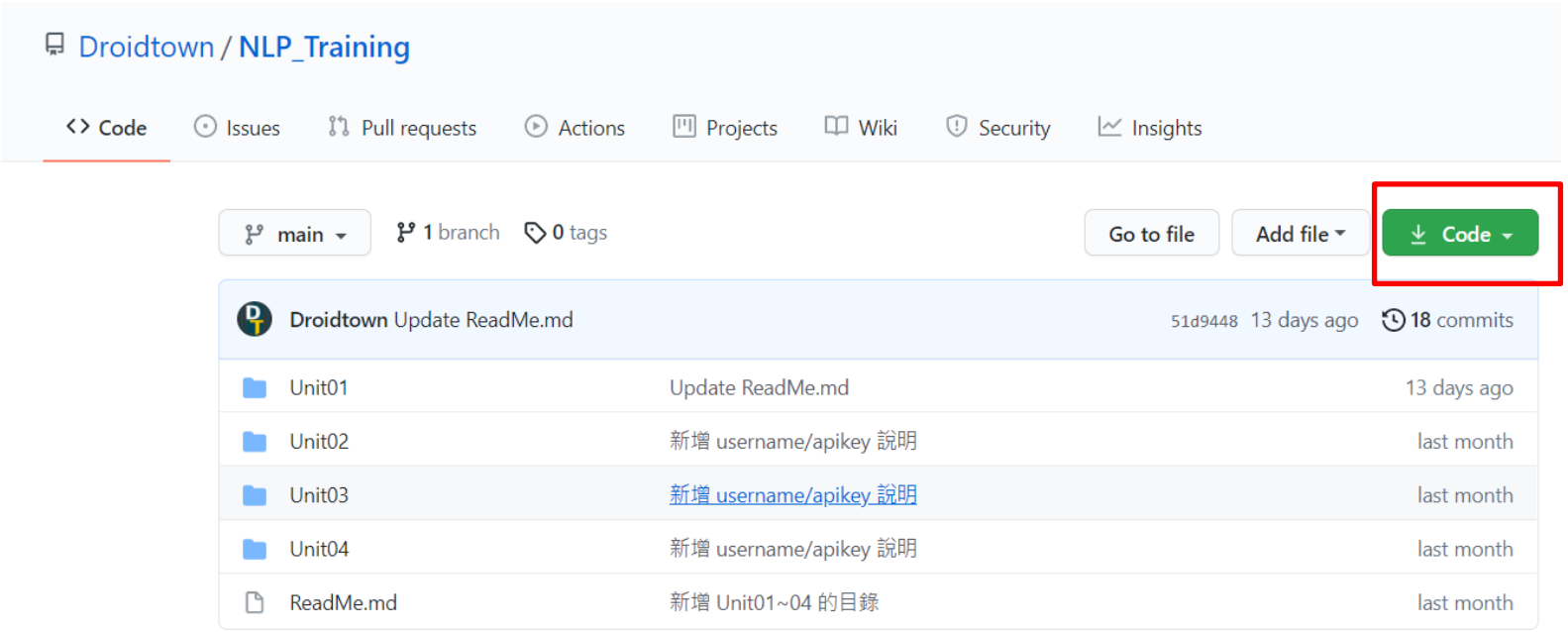
進階用法之二：載入自訂字典

- 處理不同領域的文本時，我們可以載入相應的領域字典以便增加處理結果的正確率。以下示範載入MLB 和 NBA 兩種字典來處理棒球和籃球的語料結果。
- 字典檔的存放位置在 ArticutAPI 的 Github.com 專案內 https://github.com/Droidtown/ArticutAPI



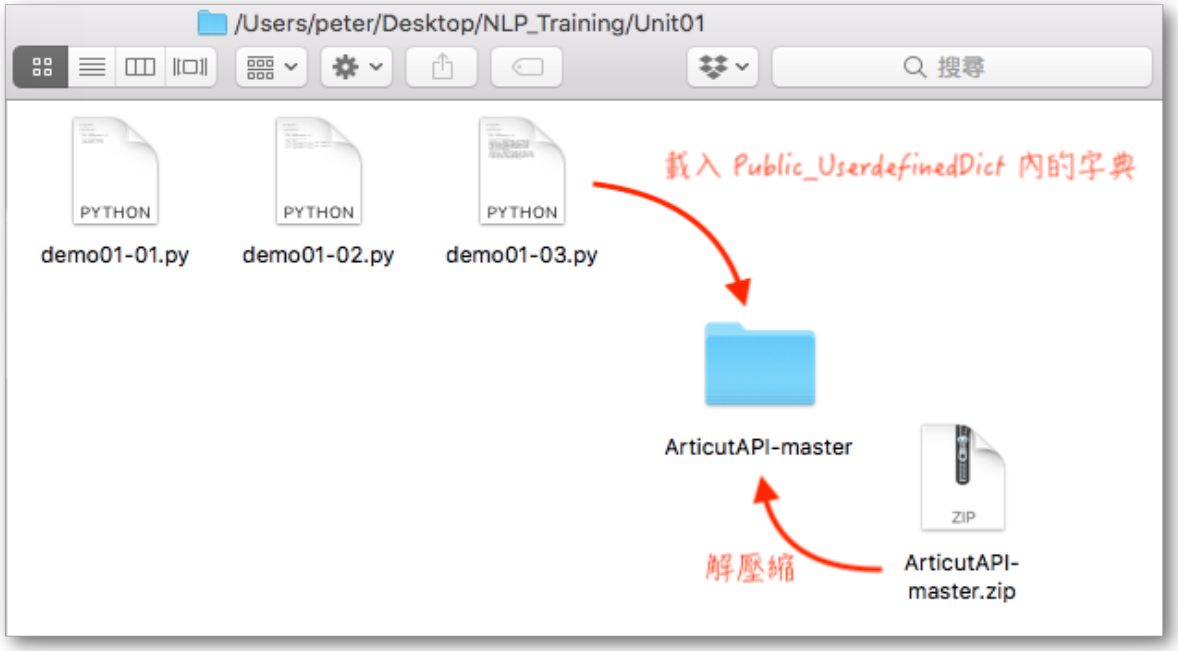
進階用法之二：載入自訂字典

- 點擊圖中畫面右方的 [Code] 找到 [Download ZIP] 的選項，下載壓縮檔，此次下載會將所有 NLP_Training 內容下載下來，然後將它解壓縮。



進階用法之二：載入自訂字典

- 最後請把 ArticutAPI-master 放在您使用的範例檔案旁邊，這樣後面示範才讀的到檔案。
  (如圖，如果demo01-03.py 為範例檔，那 ArticutAPI-master 要放在它旁邊)



圖一

進階用法之二：載入自訂字典

- 在之後的課程裡，我們將以「體育運動」類文章作範例。我們會使用 ArticutAPI-master 的 Public_UserdefinedDict 內幾個球隊名稱的領域字典。
- 我們將這幾個領域字典讀取出來，並結合成單一一個字典檔，和要處理的文本一起傳給 Articut 以便增加結果的正確性。

進階用法之二：載入自訂字典

- 以下為範例語料 (可以複製使用)

棒球語料：

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，此時輪到康福托上場打擊。在2好1壞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。

籃球語料：

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16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。

課間練習3

- 仿照後一頁的例子，結合兩個字典檔成為一個 mixedDICT.json 檔以後，分別將 basebalSTR 和 basketballSTR 輸入 Articut 進行處理並取得回傳結果。

課間練習3

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
from ArticutAPI import Articut
import json

if __name__ == "__main__":
    username = ""
    apikey   = ""
    articut = Articut(username, apikey)

    baseballSTR = ...                    #參考第 31 頁語料
    basketballSTR = ...                  #參考第 31 頁語料
```

課間練習3

```
#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)
```

課間練習3

```
# 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(baseballSTR, userDefinedDictFILE="./mixedDICT.json")

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

作業: 從字頻來分析文本

任務1:

- 請用 ArticurtAPI 將以下兩個文本 medicalSTR 以及 weatherSTR 用 lv2 進行斷詞，並將斷詞的結果依據頻率進行排序，取前 20 名。

medicalSTR

- 指揮中心今天公布新增135例武漢肺炎確定病例，其中132 例為本土個案，另有3例境外移入；確診個案中新增8例死亡。指揮官陳時中說，病例及死亡數減少是好現象。中央流行疫情指揮中心指揮官陳時中下午在記者會中說明，新增132例武漢肺炎（2019冠狀病毒疾病，COVID-19）本土病例，為62例男性、70例女性，年齡介於未滿5歲至80多歲，發病日介於6月1日至6月14日。陳時中說，個案分布以新北市65例最多，其次為台北市26例，苗栗縣18例，桃園市12例，基隆市3例，台南市、台中市及花蓮縣各2例，嘉義縣及彰化縣各1例。其中雙北地區以外縣市41例中，33例為已知感染源、6例關聯不明、2例調查中；相關疫情調查持續進行中。...

資料來源：
https://www.cna.com.tw/news/firstnews/202106150042.aspx

weatherSTR

- 今天是一年一度的端午佳節，台語有句俗話說：「未食端午粽，破裘不甘放」，形容過了端午之後，天氣才會穩定炎熱，終於可以把冬衣給收起來了，實際上今天(14日)開始台灣附近西南風逐漸增強，未來這一週的確都是暖熱且潮濕的西南風影響，溫度普遍都偏高，尤其是位在西南風背風面的北台灣，受到西南風過山後沉降增溫作用的加持，接下來幾天都很可能出現超過36度以上的高溫，其他像是中部地區、花東地區高溫也都有33-35度，只有直接面迎西南風的南部地區因為雲量較多又有機會下雨，高溫相對較低只有30-32度，這樣的狀況預期會持續一週左右，提醒大家要留意高溫，外出要多喝水預防中暑，同時做好防曬。...

資料來源：
https://tw.news.yahoo.com/%E5%88%86%E9%90%98%E5%A0%B1%E5%A4%A9%E6%B0%A3-%E9%80%B1%E4%BA%8C-06-15%E6%97%A5-%E8%A5%BF%E5%8D%97%E9%A2%A8%E6%BC%B8%E5%A2%9E%E5%BC%B7-091945627.html

作業: 從字頻來分析文本

任務1所需能力

- 解讀 ArticutAPI 斷詞後的字典檔:
  - 因為回傳是字典檔，我們可以運用以下方式取得斷詞結果 (回傳的會是一個字串)

```
resultDICT['result_segmentation']
```

作業: 從字頻來分析文本

任務1所需能力

- 將字串進行分割:
  - 因為回傳結果是整個字串，如

    \n/指揮中心/今天/公布/新增/135例/武漢/肺炎/確定/病例/

    所以我們必須將字串以 '/' 進行分割。

```
inputSTR = "\n/指揮中心/今天/公布/新增/135例"
inputSTR.split("/") #請電腦依據 "/" 切開字串
```

['\n', '指揮中心', '今天', '公布', '新增', '135例']

作業: 從字頻來分析文本

任務1所需能力

- 根據切割好的字串列表去統計每個字出現頻率，並轉成 Dataframe。

```
import pandas as pd

#將我們做好的list存成dictionary然後轉成dataframe
wordDF = pd.DataFrame({"WORD":["apple","apple","guava"]})

# value_counts()會自己去數設定欄位中字詞現的頻率
wordDF = pd.value_counts(wordDF.WORD).to_frame().reset_index()

#幫新做好的dataframe命名
wordDF.columns = ['WORD','FREQ’]
```

	WORD	FREQ
0	apple	2
1	guava	1

作業: 從字頻來分析文本

任務2:

- 經過斷詞還有頻率的分析，我們大概可以知道每篇文章的代表詞彙，但是在這些文章中的前20名裡頭，我們發現有一些標點符號 (、，。) 以及換行符號 (\n) 也被包含到分析內。
- 這些標點符號對於目前的文本貢獻性不大，因此如果可能的話我們會盡量將其移除。

作業: 從字頻來分析文本

任務2所需能力

- 將字串特定內容進行取代: 使用 re.sub()。

```
import re

testSTR = "我~~喜歡文本分析~~~~!!!!!!!"

#re.sub(要取代的東西,取代成什麼樣子,要被處理的字串)
re.sub("[~!]","",testSTR)
```

我喜歡文本分析

作業: 從字頻來分析文本

任務3: 反思

- 觀察斷詞後並濾掉標點符號後的表格。
  - 這樣的結果其實並不盡人意，有一些意義不大的功能詞，例如“的”。
- 請想想有什麼可能的方法可以讓分析出來的結果更能表達文本的差異以及內容呢?

文本分析與程式設計

Week 2

學習目標

- 了解什麼是「特徵詞」。
- 如何使用不同的工具來輔助你找到特徵詞。

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 (有經過編輯)

文本分析任務

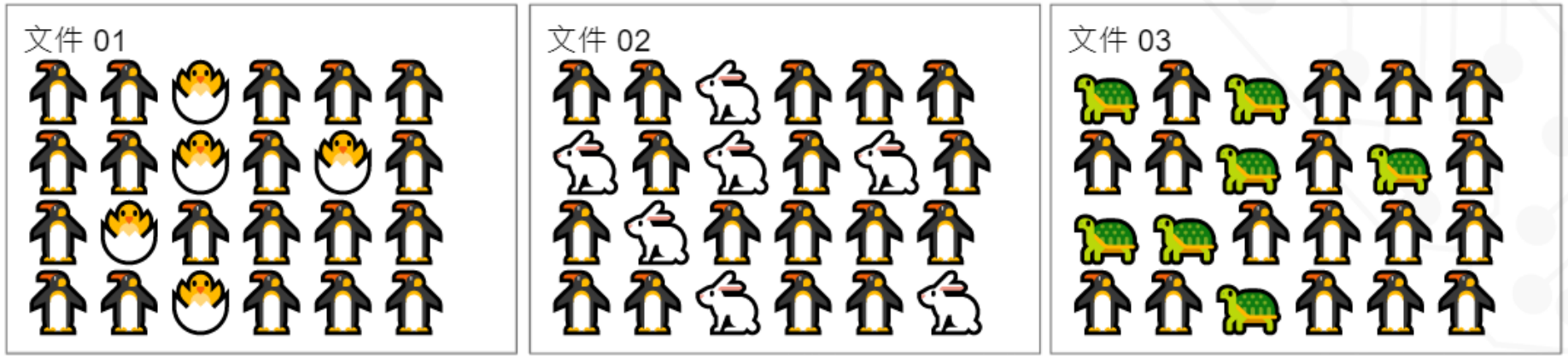
- 文本分析是常見的自然語言處理 (Natural Language Processing, NLP) 任務。
- 有三種常見的取特徵詞手法：
  - 一是取 TF-IDF 特徵詞。
  - 二是依詞性取特徵詞。
  - 三是依「人、事、時、地、物」取特徵詞。
- 本週課程會以 TF-IDF 和詞性為主
- 依「人、事、時、地、物」取特徵詞會是下週內容，不過這週會先開始介紹一點。

什麼是特徵詞

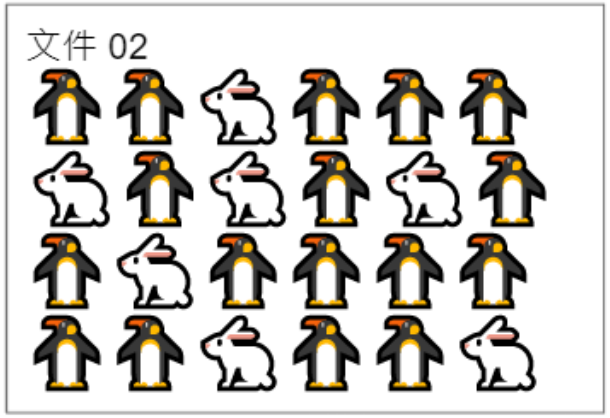
- 特徵詞可以做為代表文本內容的一種參考維度。
  - 一篇文本之中，那些字詞可以拿來區辨不同的文件。

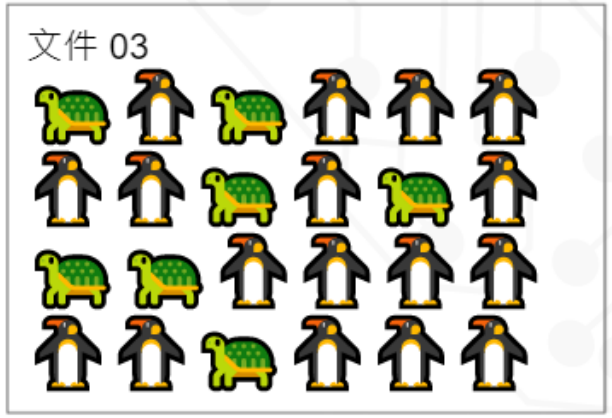
什麼是特徵詞

- 以下面這三個 emoji 組成的文件來舉例，請問什麼圖案可以拿來區分這三個文件呢？
- 也就是說，哪一個「詞」是以下文件的「特徵詞」呢？



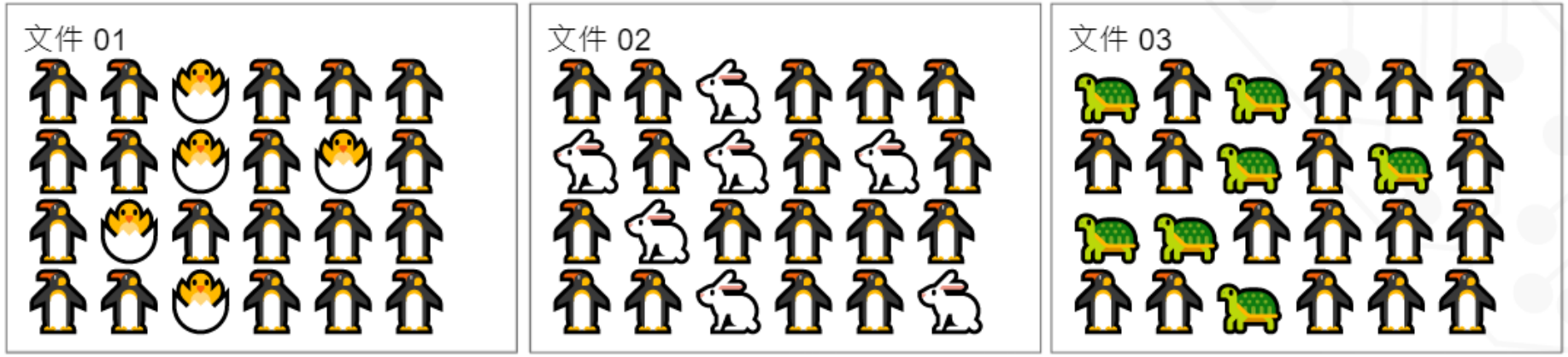
什麼是特徵詞





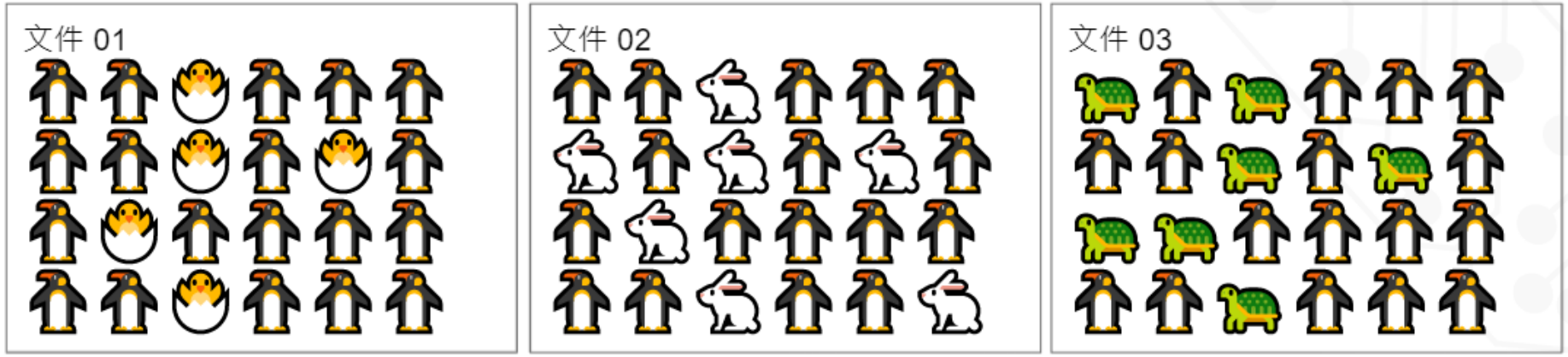
- 我們會說：
- 第一個文件的特徵詞是 
- 第二個文件的特徵詞是 
- 第三個文件的特徵詞是 

什麼是特徵詞



-  是特徵詞，因為他們是分別這三個文件中比較特別的存在。

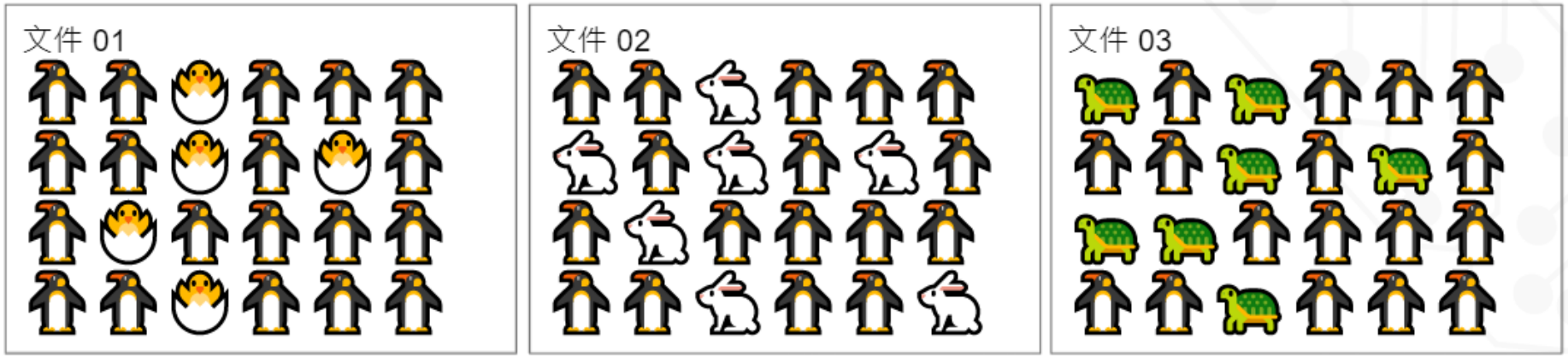
什麼是特徵詞



- 特別的是，當人類在做判斷的時候，我們不會去拿每個文件中數量最多的  來當作特徵。

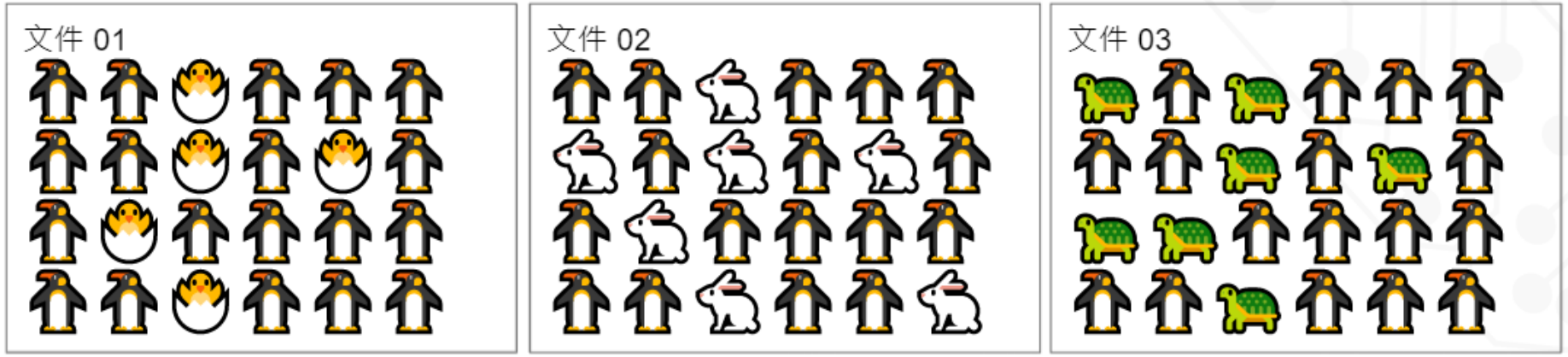
- 反而我們會用    這三個數量比較少的當作該文件的特徵，因為我們覺得這是這些文件「特別」的地方。

什麼是特徵詞



- 所以從以上的例子，我們可以知道一個詞出現的「頻率最高」不能當作判斷特徵詞的唯一指標。
- 那麼，該怎麼找到屬於人類判斷的「特徵詞」呢？

特徵詞取得方法



- 我們有三種方法
  - 取 TF-IDF
  - 看詞性
  - 看「人、事、時、地、物」

用TF-IDF來取得特徵詞

什麼是TF-IDF

- TF-IDF 是一種統計方法，用在計算某一詞在一篇文章中的重要性。
- 而TF 和 IDF 各有自己的意思。

什麼是TF-IDF

- TF 是 Term Frequency 的縮寫，意思是某個字在本文件裡出現的比率。
- 計算方法可以從計算每一個字的數量開始。
- 我們會以這個emoji 文件為例：

什麼是TF-IDF

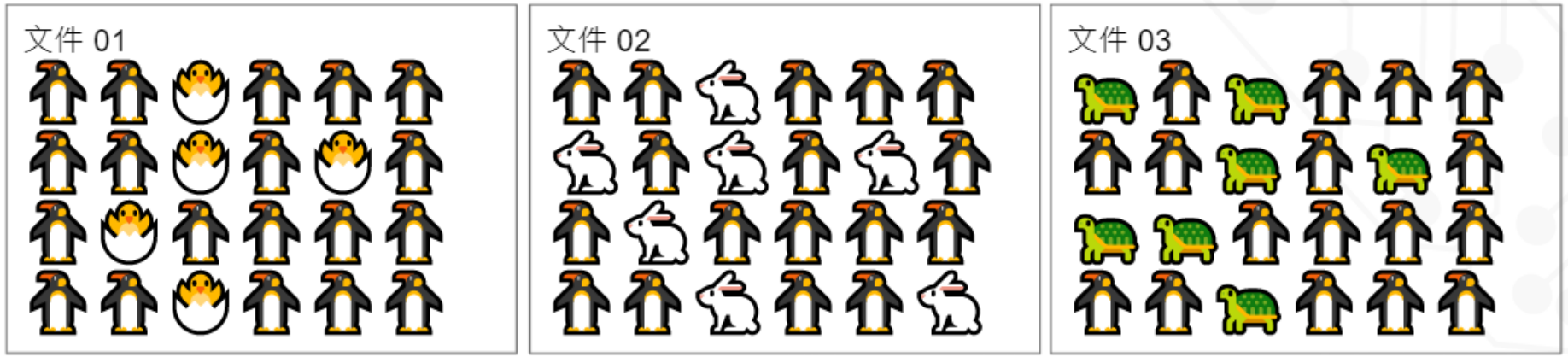
- TF 的計算方法如下，以右邊的 emoji 文件為例
  - 計算每個詞在本文件中的頻率
    例如：
     有19個
     有5個
  - 再來以下面算式計算他們的比重
    這個比重就是 TF
  - 例如  的TF 是 19/(19+5) 大約是 0.792
     的TF 是5/(19+5) 大約是 0.2

什麼是TF-IDF

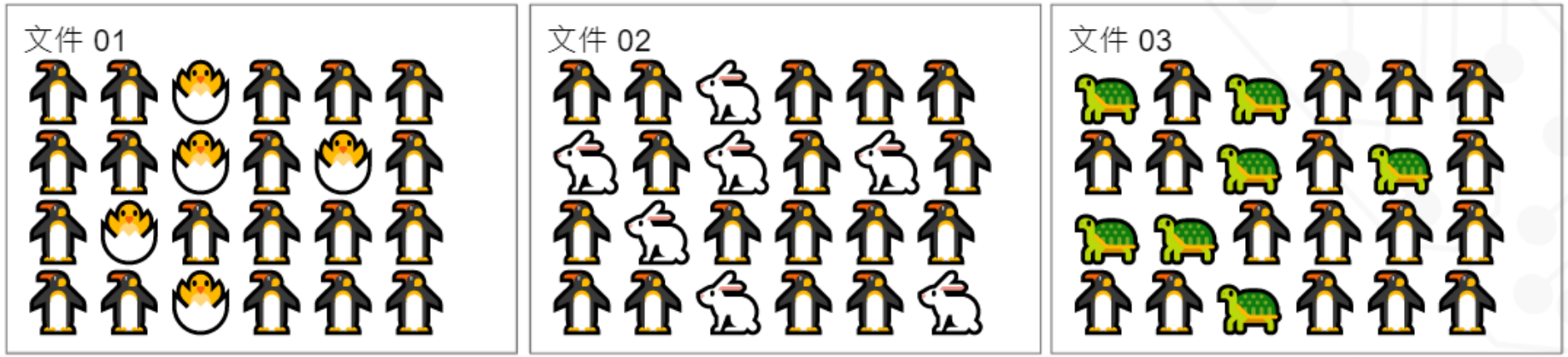
- 從上頁的例子，我們可以知道企鵝的 TF 比較大，所以從 TF 的角度來說，企鵝比較重要。

什麼是TF-IDF

- IDF 是 Inverse Document Frequency 的縮寫，也就是某個字在所有的文件中出現的頻率。
- 我們再來看看這些 emoji 文件：

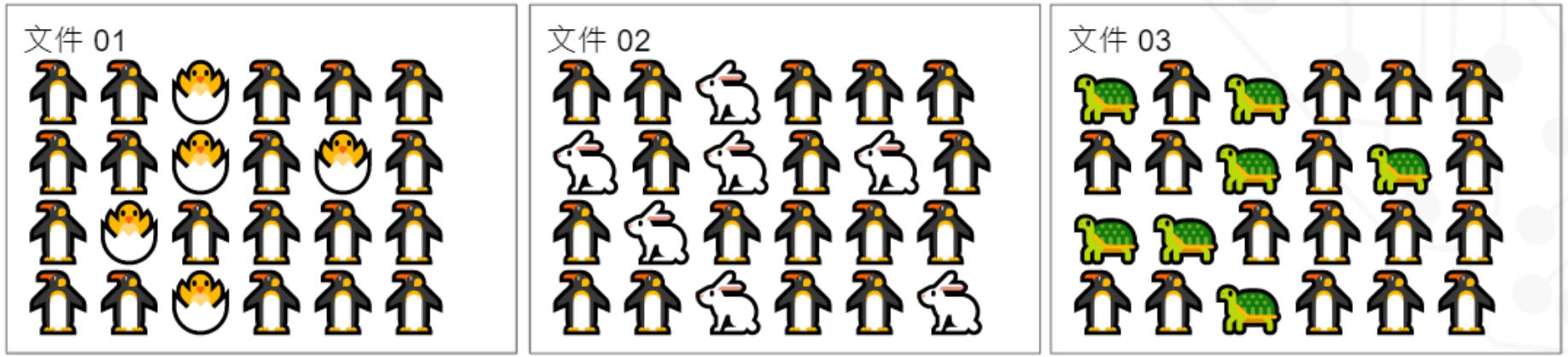


什麼是TF-IDF



- 計算IDF時，我們要先計算這些內容
  - 我們需要知道我們有多少文件 ➔ 目前我們有三個文件
  - 某個字出現在文件中的次數
    例如
    ➔ 🐧 出現在這三個文件中
    ➔ 🐣🐰🐢 各出現在一個文件中 (我們用🐣當例子)

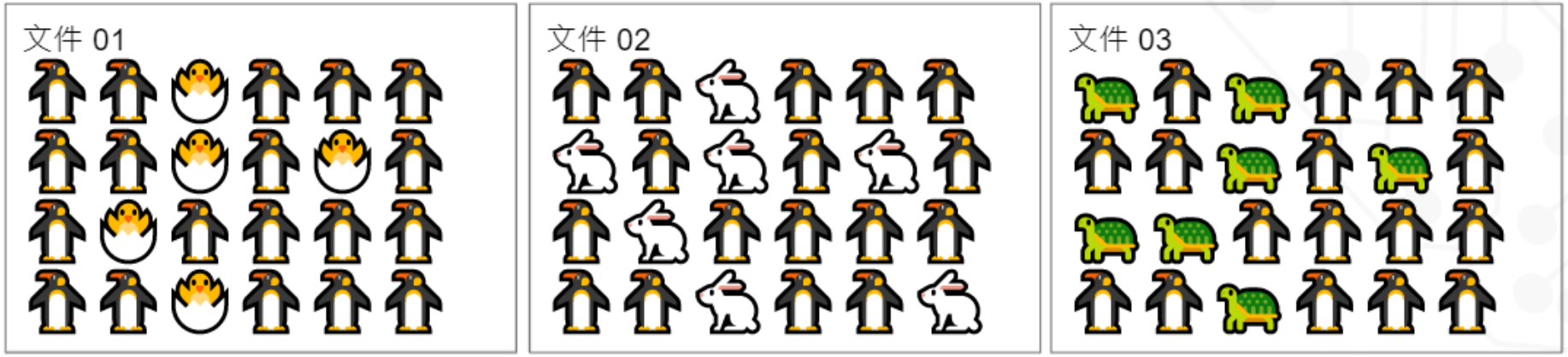
什麼是TF-IDF



- 計算IDF時，我們要先計算這些內容
  - 按照這個算式：log(文件數/該單詞出現在幾個文件)

➔ 所以 🐧 是log(3/3) = 0
➔ 🐣 是 log (3/1) 大約等於 0.477

什麼是TF-IDF

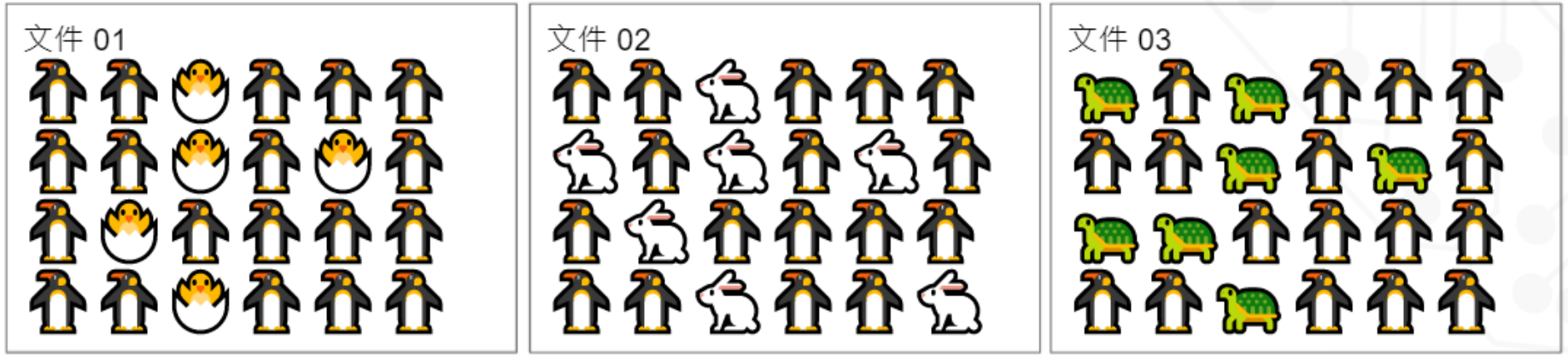


- 計算之後我們發現🐣的算出來比較大，然後🐧比較小
- 這個代表什麼呢？
- 這是代表🐣在IDF的角度中，比較重要

課間練習1

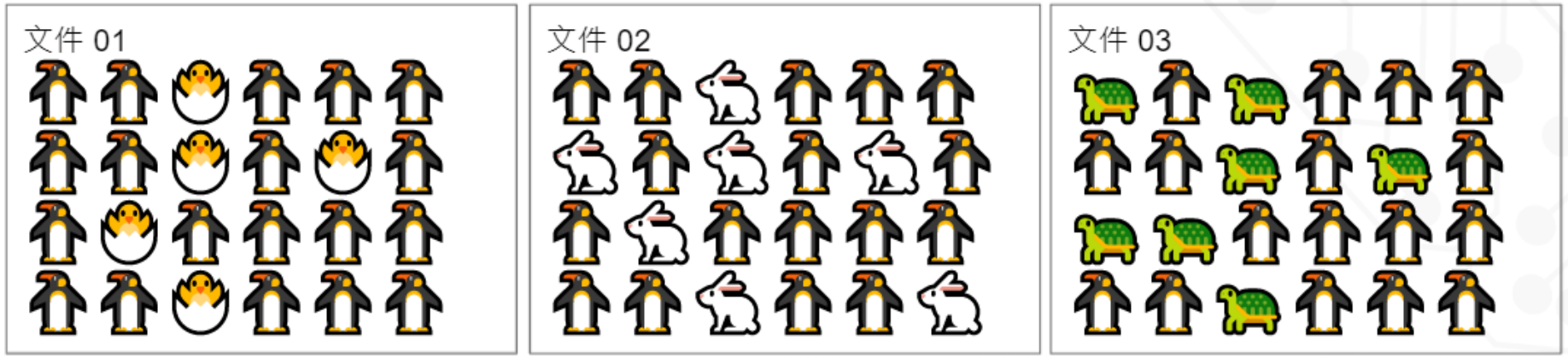
- 想想看，你是怎麼判斷一個文章的大意呢？
- 想想看特徵詞的定義，你覺得特徵詞等同於文章大意嗎？
- 看完 TF 和 IDF 的內容，你覺得 TF 和 IDF 哪個比較接近我們人類思考之中的特徵詞呢？為什麼？
- 想一想 TF 和 IDF 是如何計算的，你覺得和人類判斷語意一樣嗎？為什麼？

什麼是TF-IDF



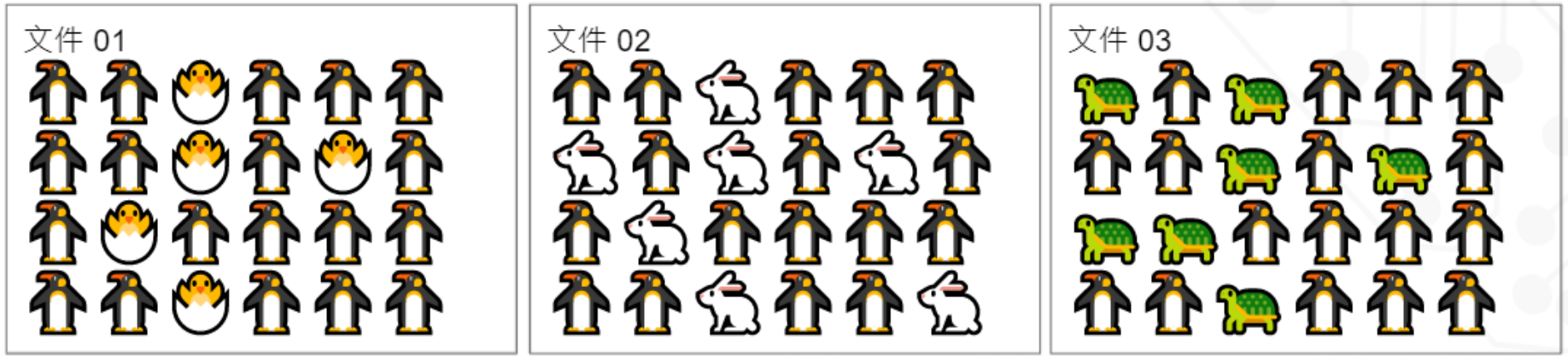
- 最後，我們可以將我們計算出來的TF 和 IDF 相乘
- 例如 🐧 是0.792 * 0 = 0
- 而 🐣 是 0.2 * 0.477 = 0.0954 ➔ 比較重要
- 0 和 0.0954 就是所謂的「 TF-IDF 權重值」

TF-IDF 可以告訴我們什麼？



- 現在我們知道TF 和 IDF 背後的原理，那 TF-IDF 可以告訴我們什麼呢？

TF-IDF 可以告訴我們什麼？



- TF-IDF 可以告訴我們文件數量和詞的頻率之間的關係
- 所以我們可以發現TF-IDF 可以幫助我們過濾常見的詞。常見字詞為什麼要過濾呢？我們先來做課間練習2 來想想看。

課間練習2

- 請看下一頁的新聞，並利用前一堂課學習到的 Articut 斷詞方法斷詞，並計算頻率。
- 什麼字詞頻率最高？高頻率的那些字是可以代表那篇新聞的字詞嗎？

課間練習2

- （中央社記者葉冠吟台北27日電）入圍坎城影展、改編自村上春樹短篇小說的日本電影Drive My Car，由西島秀俊、岡田將生主演，台灣演員袁子芸也參演，更透露西島秀俊本人相當親切，還有許多暖男舉動。袁子芸曾參演戲劇「泡沫之夏」、「未來媽媽」，Drive My Car是她的首部日本電影，飾演一名母語是中英雙語的女演員，在角色背景設定上與她自身有許多相似之處。袁子芸表示，開拍第一天自己很緊張，而且第一場戲就與岡田將生有近距離接觸。不過當時午餐放飯時間延遲，拍戲時大家都有點餓，沒想到在最安靜的瞬間，岡田將生的肚子竟然咕嚕叫了一聲。袁子芸回憶，當下兩人都很專業的繼續演出，等導演一喊卡之後，才忍不住大笑出來，連岡田將生都笑著表示不好意思。不過也因為這個意外插曲，沖淡第一天拍攝的緊張和陌生感，瞬間拉近演員之間的距離。關於男主角西島秀俊，袁子芸看過許多他的作品，原以為他是比較有距離感的大明星，沒想到西島秀俊在讀本過程就相當親切，還有許多暖男舉動。袁子芸分享，有天海外演員殺青後，相約要去慶祝，但因西島秀俊隔天一早還有拍攝，無法同行，他便私下交代製片想請大家吃飯，麻煩製片先代為招待。近日電影傳出入圍坎城影展好消息，袁子芸也相當開心，她表示劇組群組簡直要暴動。雖然她很希望能和劇組一起慶祝，卻因大家都分散不同地方，又正值疫情期間，只能透過通訊軟體互相道賀，期待Drive My Car 能在坎城影展奪下好成績。由柏林影展銀熊獎得主、導演濱口龍介執導的Drive My Car，描述一名想尋找亡妻生前外遇對象的舞台劇演員，藉由與沉默寡言女司機的對話，描繪出各自的故事與情感，電影8月20日將於日本上映。（編輯：張雅淨）1100627

來源:https://www.cna.com.tw/news/amov/202106270091.aspx

TF-IDF 可以告訴我們什麼？

- 通過剛剛的課堂練習2，還有以上我們很熟悉 emoji 文件，以及我們前面討論的討論詞，我們會發現反而最常出現在文件中的詞，通常都並不是最可以突出文件特點的詞，反而只有出現在某些文件的才是特徵詞
- 如果還想更多了解TF-IDF 可以參考以下內容：https://bit.ly/tfidf_explain

TF-IDF 限制

- 從以上的操作，我們會發現 TF-IDF，也就是單頻文章字詞的頻率來判斷，其實解讀內容有限。

為什麼呢？

- TF 中所得到的高頻詞，有可能無法解讀
  - 從以上的練習題，我們會發現，一篇文章如果我們直接只看高頻詞那我們可能會完全不了解這篇文章在講什麼，因為這些高頻詞，很多都是「功能詞」，也就是語法架構的一部分，並非「實詞」(content word)，也就是具備比較多「語意」的詞彙。

實詞 vs. 功能詞

- Content words，或是又稱「實詞」，指的是句子中有重要意義的意思。
- 例如以下句子 (取次於 Snow White 的維基條目)

"Snow White" is a 19th-century German fairy tale that is today known widely across the Western world.

  - 其中實詞就包括 Snow White, German, fairy tale
  - 而相對於實詞，如果只有文法上功能的就是虛詞，也就是功能詞。

https://en.wikipedia.org/wiki/Snow_White

實詞 vs. 功能詞

- 而相對於實詞，如果只有文法上功能的就是虛詞
- 例如在剛剛的句子中

"Snow White" is a 19th-century German fairy tale that is today known widely across the Western world.

  - a, that, the 就是虛詞

課間練習3

- 可能你會說，我們可以把功能詞濾掉啊？然後再做TF-IDF分析，這樣 TF-IDF 會不會比較有意義呢？請和同學討論看看。

TF-IDF 限制

- 如果濾掉功能詞，的確從 TF-IDF 或許可以解構出更多的內容。不過光是從「頻率」中，其實不一定可以看出所有端倪，甚至頻率也有可能誤導，為什麼呢？
  - 因為當電腦在計算TF-IDF 時其實並沒有注意語句的順序。
- 如果說有兩篇文章

1. 小明向小美說我愛你，你是我的最愛。
2. 小明向小美說你愛我，我是你的最愛。

  - 根據上述TF-IDF 的算法，如果我們想算出第一篇和第二篇文章「你」和「我」的差異，因為字數完全一樣，所以從TF-IDF 來看，這是兩篇特徵一樣的文章。不過我們因為有上下文，也就是語境(context)所以我們知道不一樣

TF-IDF 限制

- 或許很多人覺得 IDF 已經可以抓到整篇文章的精華，因為 IDF 所計算出來的結果是某篇文章獨有的內容。不過「某篇文章獨有內容」可以代表整篇文章想要講的概念嗎？

請看以下比喻：

- 便當街裡，每家店都有排骨飯，但只有「好好吃便當店」的排骨飯有加辣椒。
- 從這個敘述中，「辣椒」是一個特徵，這個好比 IDF 計算出來的結果。
- 不過我們會把「好好吃便當店」的這份有加辣椒的飯的排骨飯，叫做「辣椒飯」嗎？其實並不會。

TF-IDF 限制

- 同理，如果我們有兩篇文章，當中 IDF 所計算出來的，其實也頂多算是我們辨別某些文章的特徵，但是這個特徵一定等於整篇文章的核心嗎？並不一定。
- 所以當我們在理解 TF-IDF 的內容時，我們需要清楚 TF-IDF的原理，以及分清楚我們想要透過文本分析想要分析什麼內容。
- 我們需要清楚理解 TF-IDF 僅是工具，TF-IDF 結果不能直接等同於文本解釋。

延伸資料

- 如果還想更多了解TF-IDF 可以參考以下內容：https://www.cc.ntu.edu.tw/chinese/epaper/0031/20141220_3103.html

- 以下這個連結是有關 Jieba 以及 Articut 的 TF-IDF 運算以及優缺點比較: https://blog.droidtown.co/post/186883773617/tf-idf

如何利用 TF-IDF 取出特徵詞

- 使用 Articut 內建的 analyse 工具包裡的 `extract_tags()` 函式來取得經 TF-IDF 計算的特徵詞
- 延續上週課程中的棒球與籃球的例子

```
baseball_TFIDF = articut.analyse.extract_tags(baseballResultDICT)
print(baseball_TFIDF)

basketball_TFIDF = articut.analyse.extract_tags(baseballResultDICT)
print(baseball_TFIDF)
```

回憶一下，`baseballResultDICT` 是 `articut.parse()` 回傳的結果。

如何利用 TF-IDF 取出特徵詞

- 棒球語料的 TF-IDF 存在 baseball_TFIDF 裡，印出內容如下：

執行結果

```
['\n', '大都會', '馬林魚', '投手', '敲出', '安打', '2', '1', '巴斯', '局面', '康福托', '比賽', '被', '週三', '紐約', '此戰', '使用', '車輪戰', '4名', '輪番', '上陣', '壓制', '打線']
```

- 籃球語料的 TF-IDF 存在 baseball_TFIDF 裡，印出內容如下：

執行結果

```
['\n', '兩', '米契爾', '康利', '罰', '勝利', '太陽', '攻勢', '爵士', '籃板', '3', '助攻', '11分', '投', '一', '昨晚', '紐約', '西區', '霸王']
```

TF-IDF 的計算結果：觀察

- 從兩篇文章的 TF-IDF 特徵詞裡，如果沒有相關的背景知識知道「大都會」、「馬林魚」是棒球大聯盟的球隊，而「爵士」是 NBA 的球隊名稱的話，其實很難確認究竟哪些詞彙是可以做為「棒球類」或是「籃球類」的文本分類特徵詞。

TF-IDF 的計算結果：觀察

- 但我們可以確認的是「投手」、「安打」、「打線」這三個名詞，應該是只有棒球類的文本才會用到。而「籃板」這個名詞，則是籃球類的文本才會用到。
- 同理，我們也能觀察到「敲出」、「保送」、「打擊」這幾個動詞同時出現的文章，有很大的機率是在描寫棒球類的文本。而「助攻」、「上籃」則常見於描寫籃球比賽過程的動詞。

TF-IDF 的計算結果：觀察

- 這個觀察最重要的是讓我們發現「詞性」(前文提到的「名詞」、「動詞」) 是可以做為抽取特徵詞的一種方法。接下來，我們利用 Articut 的詞性篩選工具來取出「名詞」和「動詞」。

用詞性來取得特徵詞

名詞比較-輸入

- 我們使用 Articut 內建的 getNounStemLIST() 函式來取得文本中各句的名詞。

```
baseballNounLIST = articut.getNounStemLIST(baseballResultDICT)
print(wordExtractor(baseballNounLIST))

basketballNounLIST = articut.getNounStemLIST(basketballResultDICT)
print(wordExtractor(basketballNounLIST))
```

名詞比較-輸出

- 棒球報導文本的名詞

執行結果

['主審', '優勢', '分', '勝利', '安打', '局面', '意識地', '手', '打線', '打者', '投手', '此戰', '比賽', '滿壘', '球', '球隊', '登板', '眼', '角滑球', '觸身球', '身肘', '身體', '車輪戰', '追平比數', '這球', '陽春砲', '領先']

- 籃球報導文本的名詞：

執行結果

['中', '人', '勝利', '單場', '場', '布克', '延長賽', '戰', '攻勢', '機會', '次', '波格丹諾維奇', '籃板', '階段', '霸王']

- 從名詞裡看出來，有些詞彙 (例如「勝利」) 是重覆的，要得到更好的效果的話，還可以把重覆的詞彙去除以便讓不同的類別都具有各自獨立的特徵詞表。

名詞比較-輸出

- 棒球報導文本的名詞

執行結果

['主審', '優勢', '分', '勝利', '安打', '局面', '意識地', '手', '打線', '打者', '投手', '此戰', '比賽', '滿壘', '球', '球隊', '登板', '眼', '角滑球', '觸身球', '身肘', '身體', '車輪戰', '追平比數', '這球', '陽春砲', '領先']

- 籃球報導文本的名詞：

執行結果

['中', '人', '勝利', '單場', '場', '布克', '延長賽', '戰', '攻勢', '機會', '次', '波格丹諾維奇', '籃板', '階段', '霸王']

- 就觀察名詞的情形，黃色部分的標記有助於我們理解文本所討論的主題是棒球類，相較於籃球綠色標記的部分有助於我們理解文本主題為籃球。

動詞比較-輸入

- 我們使用使用 Articut 內建的 getVerbStemLIST() 函式來取得文本中各句的動詞。

```
baseballVerbLIST = articut.getVerbStemLIST(baseballResultDICT)
print(wordExtractor(baseballVerbLIST))

basketballVerbLIST = articut.getVerbStemLIST(basketballResultDICT)
print(wordExtractor(basketballVerbLIST))
```

動詞比較-輸出

- 棒球報導文本的動詞

執行結果

['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']

- 籃球報導文本的動詞：

執行結果

['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']

- 從動詞裡看出來，有些詞彙 (例如「打出」、「投」、「隨」) 是重覆的，要得到更好的效果的話，還可以把重覆的詞彙去除以便讓不同的類別都具有各自獨立的特徵詞表。

動詞比較-輸出

- 棒球報導文本的動詞

執行結果

['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']

- 籃球報導文本的動詞：

執行結果

['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']

- 針對動詞的部分，則是在以上標記的部分顯示出文本類別的線索，要特別注意的是，區分文本時需要多多考慮不同的詞性(例如:名詞與動詞)，如此一來才能正確找出文本的區分關鍵。

用「人、事、時、地、物」取特徵詞

取得人事時地物的工具

- 如果你想知道的特徵詞可以以「人事時地物」來分類，那可以嘗試使用以下的工具來幫助你
- 「人」、「事」、「地」、「時」和「物」因為通常都是名詞，所以可以使用下面的工具
  - 使用剛剛取名詞的 `getNounStemLIST()`。
  - 使用 `articut.getContentWordLIST()`取得 content word。
  - 下週會再介紹更詳細的內容。

取得人事時地物的工具

- 如果你想知道的特徵詞可以以「人事時地物」來分類，那可以嘗試使用以下的工具來幫助你
- 「地」
  - Articut 也可以直接擷取地方的名稱。
  - 使用 `articut.getLocationStemLIST()`。

課間練習4

- 請使用籃球和棒球的文章來練習。
  - 直接使用之前的 basketball 和 baseball 的 resultDICT。
- `articut.getLocationStemLIST()` 和用 `articut.getContentWordLIST()` 和前面取名詞和動詞的方法是一樣的，請嘗試自己摸索使用看看。
- 觀察看看 content word (內容詞) 和 location word (地方名稱) 是否具有比「動詞」或「名詞」更好/更差的特徵表現能力。說說看你的觀察。

作業: 遠距教學新聞的特徵詞抽取

- 作業敘述:

  2021年由於疫情關係遠距教學十分盛行，很多學者對遠距教學裡頭的學生端以及教師端進行不少研究，而在這個任務中我們將從另一個新的角度來探討遠距教學對社會的影響，我們蒐集了10篇從聯合新聞網的新聞，利用關鍵字遠距教學進行檢索，並取出前十篇進行分析。

作業: 遠距教學新聞的特徵詞抽取

- 資料概述

	標題	類別	內容
0	仁寶Q3營收 可望成長	股市	代工廠大仁寶（2324）今年來持續受惠居家工作、遠距教學等宅經濟熱潮，推升筆電銷售暢旺，惟第...
1	元山 下半年出貨 看增	股市	隨著歐美疫苗施打率日益提升，車市有望在下半年回溫，國內車用電子與生活科技系統廠元山（6275...
2	宅經濟退燒 Chromebook、平板看淡	產經	新冠肺炎疫情引發居家工作、遠距教學熱潮，帶來的Chromebook、平板等宅經濟硬體需求熱...
3	暑假將至...兒盟籲政府助弱勢兒少課輔轉型線上陪伴	文教	全台學校自5月下旬停止到校上課，採遠距學習，兒福聯盟今舉行「弱勢兒少數位學習困境與對策」線上...
4	竹市議員發起愛心筆電助學計畫 支援弱勢學子暑假學習	文教	三級警戒以來全國已停課快兩個月，所有學生需依賴3C設備線上上課，弱勢家庭數位落差大，缺乏手機...
5	手機終有訊號了 雙溪區魚行里基地台啟用	地方	雙溪區魚行里的坤溪、公館、八股、丁子蘭等聚落，一直以來都是區內通訊的隱蔽帶，造成居民手機訊號...
6	避免「線上中輟」！中山工商遠距教學按表操課點名課輔	文教	疫情讓各級學校停課」，採遠距教學已超過一個月，有學校出現「線上中輟」現象，令家長憂心忡忡。高...

新聞來源: 聯合新聞網

作業: 遠距教學新聞的特徵詞抽取

- **任務1: 找出文教類的特徵詞**

  請運用 Articut 的 extract_tags() 函式找出文教類的前 20 名特徵詞。

- **任務2: 找出股市類的特徵詞**

  請運用 Articut 的 extract_tags()函式找出股市類的前20名特徵詞。

- **任務3: 比較兩類特徵詞**

  請你比較兩類特徵詞，請問你覺得有那些字詞能代表文教類，而那些字詞能代表股市類？而那些具代表性的字詞反映出遠距教學對這兩方面的影響？

作業: 遠距教學新聞的特徵詞抽取

- **任務4: 根據特徵詞將文本進行歸類**

  假如今天沒有產經這一個新聞類別，你覺得那些新聞應該要歸到哪文教類還是股市類呢？請試著用特徵詞的方式進行比較。

作業: 遠距教學新聞的特徵詞抽取

▶任務5: 運用名詞比較來理解文本差異

從上面的特徵詞來看感覺名詞較多，因此我們希望更進一步從名詞下手，希望你能將文教類以及股市類裡的名詞用 getNounStemLIST() 這個方法找出名詞，然後用我們之前在之前學到的計算字詞頻率的方式，做出他的頻率表，之後將頻率表視覺化成文字雲。如右所示:

5.1 名詞頻率分析表格(文教類為例) 製作方法詳見上週作業。

	Noun	FREQ
0	學校	17
1	學生	17
2	孩子	12
3	老師	10
4	設備	9
5	家長	7
6	網路	7
7	教師	7
8	資源	7
9	弱勢兒	7
10	兒盟	7
11	疫情	7

作業: 遠距教學新聞的特徵詞抽取

- **任務5: 運用名詞比較來理解文本差異**

  5.2 文字雲分析(文教類為例)

  所需工具解釋請見下一頁

作業:遠距教學新聞的特徵詞抽取

```
wordcloud = WordCloud(width = 800, height = 800, #設定畫布大小
                      background_color ='white', #背景顏色
                      min_font_size = 10, #最小字體
                      font_path="TaipeiSansTCBeta-Regular.ttf")

#畫中文的文字雲一定要設定字體路徑(你可以直接把字體放在存這份檔案的同一層)
wordcloud.generate_from_frequencies(frequencies=data)
#繪製文字雲時有很多方法，其中一個方法就是將檔案轉為 dictionary 的格式
plt.figure(figsize = (8, 8)) #顯示的圖大小
plt.imshow(wordcloud, interpolation="bilinear") #這部分是有關顏色顯示的方式，參考
plt.axis("off") #不要有x,y軸的座標
plt.show() #顯示圖片
```

文本分析與程式設計

Week03

取特徵詞

- 特徵詞可以做為代表文本內容的一種參考維度。換句話說，可以透過特徵詞來理解文本想傳達的主要概念。
- 上週的課程提到取特徵詞的方法有三種：
  - TF-IDF
  - 詞性 (名詞 /動詞)
  - 「人、事、時、地、物」取特徵詞
- 這週會講解關於如何以「人、事、時、地、物」來取特徵詞。

取特徵詞

- 為什麼要使用「人、事、時、地、物」來取特徵詞呢？
  - 如果有多篇文章中提及的「人」有大量的重覆，那麼我們大概可以推測它大概是在討論類似的主題。同樣的道理，同樣的「事、時、地、物」也可以做為分類文本時的依據。

取特徵詞

- 前次課程中已經使用過 articut.getLocationStemLIST() 來幫助我們取出地點。
- 本週課程中，會介紹如何取出人、事、時和物：

類別	範例語法
人	articut.getPersonLIST(baseballResultDICT)
事	articut.parse(baseballSTR, userDefinedDictFILE="./mixedDICT.json", level = "lv3")["event"]
時	articut.parse(baseballSTR, userDefinedDictFILE="./mixedDICT.json", level = "lv3")["time"])
地	articut.getLocationStemLIST(baseballResultDICT)
物	articut.getNounStemLIST(baseballResultDICT)

課程目標

- 知道為什麼要取出人事時地物的特徵詞。
- 學習如何取出人事時地物的內容。

以下的練習，我們會繼續使用棒球和籃球的兩篇新聞。

1. https://news-taiwan.xyz/uncategorized/39053.html
2. https://www.ctwant.com/article/111388 (有經過編輯)

從文本中抽詞

從文本中抽詞 -- 「人」

- 使用 articut.getPersonLIST()

```
baseballPeopleLIST = articut.getPersonLIST(baseballResultDICT)
print(wordExtractor(baseballPeopleLIST))

basketballPeopleLIST = articut.getPersonLIST(basketballResultDICT)
print(wordExtractor(basketballPeopleLIST))
```

棒球執行結果
['他', '巴斯', '庫爾帕', '康福托', '自己', '麥尼爾']

籃球執行結果
['保羅', '克拉克森', '康利', '戈貝爾', '米契爾']

課間練習1

- 以下有一篇娛樂新聞，請你利用 `articut.getPersonLIST()` 來找這篇新聞的人物。請問裡面有哪些人物呢？

（中央社記者王心妤台北2日電）公視「勇者動畫系列」改編自台灣漫畫家黃色書刊作品，為公視動畫元年打頭陣，9月1日將上架Netflix，超過190個國家能看見，藝人黃子佼、吳慷仁、劉冠廷、孫可芳都獻聲。「勇者動畫系列」改編自以諷刺漫畫聞名的漫畫家黃色書刊2016年推出的網路連載漫畫「勇者系列」，目前已累積逾960話，不只有龐大世界觀，也用角色的刻板印象反諷社會。公視今天舉辦線上記者會，漫畫家黃色書刊、製作人王尉修、導演楊子霆、配樂師張衛帆，以及藝人黃子佼、旺福小民與Mami、美秀集團鍵盤手冠佑、鼓鼓呂思緯、小魏魏嘉瑩及黃奕儒都分享心得。對於是否擔心觀眾對漫畫變動畫的迴響，黃色書刊表示，因劇情仍照著漫畫進行，觀眾還是能夠有思考的空間。他也指出，漫畫被改編像是夢想實現，「這是很感動的，當動畫團隊來找我，我馬上就答應了。」他也特別創作全新角色「忠誠勇者」，將會起到貫穿全作的效果。黃子佼表示，當時僅跟團隊聊了幾分鐘就決定加入，為了配合角色「老魔王」的個性，特別壓低聲音並放慢語速。已經看過全作的他表示，「沒有人是絕對的好人，或絕對的壞人，不能用種族或者長相的分類去判斷，很呼應現實世界。」為「勇者動畫系列」創作歌曲「今世世界紀錄」的旺福樂團成員小民表示，腦袋在創作過程都是動畫情節，用熱血沸騰的心情完成創作，負責演唱的Mami則說，歌曲裡的龐克精神，能夠讓人找回初心。美秀集團鍵盤手冠佑的插曲「不能重生的冒險」，則抱持「雖然做了不一定成功，但是不做一定不會成功」的心態。公視「勇者動畫系列」將於4日起，每週日晚上10時在公視、公視+播出；8月1日則在myVideo；9月1日LINE TV、Netflix上架。（編輯：陳政偉）1100702

本文取自於 https://www.cna.com.tw/news/amov/202107020302.aspx

課間練習2

- 「事」，在這裡指的是「事件」。請和同學討論看看，一個事件應該要包含哪些內容呢？

從文本中抽詞 -- 「事」

- 人、時、地和物由 articut 都可以直接取出相關的特徵詞。
- 「事」，在這裡指的是「事件」。一個「事件」由「涉及的人/物」加上「動作」構成。

從文本中抽詞 -- 「事」

- 可以使用 `articut.parse()` lv3 的「語意分析」能力來取得事件。
- 之前操作 lv1/lv2 一樣，透過 `parse()` 函式，傳入 *mixedDICT.json* 的字典檔，這次設定為 "lv3"。
- 並將最後計算後的結果取出 `["event"]` 的值

```
baseballEventLIST =
    articut.parse(baseballSTR, userDefinedDictFILE="./mixedDICT.json",
                  level = "lv3")["event"]

basketballEventLIST =
    articut.parse(basketballSTR, userDefinedDictFILE="./mixedDICT.json",
                  level = "lv3")["event"]
```

從文本中抽詞 --「事」

- 棒球類文本輸出的結果如下

```
[[], ['此戰', '使用'], ['上陣', '打線'], ['壓制', '打線'], '\n', [], ['讓', '球隊'], ['帶
著', '球隊'], ['進入', '優勢'], ['推出', '登板'], ['關門', '登板'], '\n', [], [], ['讓', '
追平比數'], [], [], [], '\n', ['形成', '局面'], ['福', '輪到'], ['上場', '康福托'], ['打擊
', '康福托'], [], ['投了', '角滑球'], ['眼', '看'], '\n', ['靠近', '自己'], ['伸進', '手'],
['碰觸', '他'], ['他', '到'], ['隨', '庫爾帕'], ['判定', '庫爾帕'], '\n', [], ['讓', '分'],
['拿下', '分'], ['再見', '分'], ['贏得', '勝利']]
```

- 從棒球比賽的文本裡可以看到有「帶著 - 球隊」、「進入 - 優勢」...等等的事件發生。甚至可以看到最後是「贏得 - 勝利」而可以推測該句的主角在最後應該是贏了比賽。

從文本中抽詞 --「事」

- 籃球類文本輸出的結果如下

```
[[], ['錯失', '勝利'], ['開始', '攻勢'], ['打出', '攻勢'], ['比', '攻勢'], '\n', ['力圖',
'康利'], ['追分', '康利'], [], ['讀秒', '階段'], ['上籃', '階段'], ['得手', '階段'], ['罰',
'中'], ['留給', '機會'], ['追平', '機會'], '\n', ['造成', '米契爾'], ['犯規', '米契爾'], ['
罰', '中'], ['隨', '勝利'], ['用', '勝利'], ['罰', '勝利'], ['鎖定', '勝利'], ['狂轟', '米
契爾'], ['助攻', '米契爾'], ['得分', '次'], [], '\n', [], [], [], ['助攻', '康利'], [], ['
人', '合計'], [], '\n', [], []]
```

- 從籃球比賽的文本裡，則可以看到「錯失-勝利」、「狂轟-米契爾」、「助攻-米契爾」...等等事件。

課程練習3

- 請使用課程練習1 提及的那則娛樂新聞來分析有哪些事件。

從文本中抽詞 -- 「時」

- 「時間」資訊也是屬於 lv3 語意分析的範疇。
- 因此操作上和前述的「事件」一樣，只在最後取出的是 ["time"] 而不是 ["event"] 而已。

```
baseballTimeLIST =
    articut.parse(baseballSTR, userDefinedDictFILE="./mixedDICT.json",
                  level = "lv3")["time"]
print(baseballTimeLIST)

basketballTimeLIST =
    articut.parse(basketballSTR, userDefinedDictFILE="./mixedDICT.json",
                  level = "lv3")["time"]
print(basketballTimeLIST)
```

從文本中抽詞 -- 「時」

- 籃球類文本輸出的結果如下

```
[[{'absolute': False, 'datetime': '2021-07-04 22:00:00', 'text': '昨晚', 'time_span':
{'year': [2021, 2021], 'month': [7, 7], 'weekday': [7, 7], 'day': [4, 4], 'hour': [22,
2], 'minute': [0, 59], 'second': [0, 59], 'time_period': 'night'}}], [], [], '\n', [],
[], [], [], [], '\n', [], [], [], [], [], [], '\n', [], [], [], [], [], [], '\n', [],
[]]
```

- 另一篇籃球比賽的文本裡，「昨晚的紐約西區霸王之戰中…」中，有「昨晚」這一個表示時間的詞彙。
  - Articut lv3 的 `["time"]` 取出了「昨晚」並加以計算出它的時間在 `"2021-05-06 22:00:00"`。

課間練習4

- 請使用課程練習1提及的那則娛樂新聞來分析裡面提及那些時間點。

從文本中抽詞 -「地」一般地點

- Articut 的 getLocationStemLIST() 函式可以像前述取得人名列表一樣地操作，將文本中指涉「某種地方」或「位置」的詞彙抽出。

```
baseballlocLIST = articut.getLocationStemLIST(baseballResultDICT)
print(baseballlocLIST)

basketballlocLIST = articut.getLocationStemLIST(basketballResultDICT)
print(basketballlocLIST)
```

棒球執行結果
['球帶內', '紐約']

籃球執行結果
['紐約', '西區']

從文本中抽詞 - 「地」

```
inputSTR =
    '''澎湖由本島以及周邊離島組成，以旅遊路線劃分本島，可以分成四大區域：馬公市區、北環、南環、以及澎湖機場以東的東環。至於澎湖離島分為四大海域：東海（鳥嶼、員貝、澎澎灘）、北海（吉貝、目斗嶼、險礁）、南海（七美、望安、虎井、桶盤）及南方四島國家公園（東嶼坪、西嶼坪、東吉嶼及西吉嶼）。熱門的七美、望安，是屬於南海四島，不要跟南方四島搞混囉！◈ 馬公市區：遊客住宿大多會選在馬公市區，尤其是中央老街周邊，美食及住宿選擇很多，離南海遊客中心也近。◈ 北環：有許多澎湖必去景點，包括跨海大橋、鯨魚洞、柱狀玄武岩、燈塔...等，通常會安排一整天的一日遊。◈ 南環：以沙灘及海邊景觀為主，山水沙灘傍晚極美。◈ 東環：最著名的景點是奎壁山摩西分海，還可以在隘門沙灘上喝咖啡享受悠閒夏日。★ 我會建議澎湖旅遊行程天數至少安排 4 天，其中 3 天在本島，跑北環線加上東環、南環景點，以及馬公市區的吃喝逛街。（下面會有更詳細的景點介紹）★ 行程中可以穿插 1 - 2 天跳島，視個人喜好安排離島海域。★ 先確認要去的離島是在哪一個海域，再分別到南海、北海、東海遊客中心搭船（三個海域的遊客中心地址都不同）。船班時間一天只能安排一個海域，下午可以回馬公市區吃喝。➤ 旅遊季節澎湖 4 – 9 月為旅遊旺季，冬天東北季風強勁，較不推薦冬天時前往旅行。個人推薦 4、5 月最適合，開始進入夏季，太陽也不會過於炙熱，還可以避開暑假的恐怖人潮。馬公市區◈ 市區景點以天后宮及周邊的中央老街為主，有許多美食及住宿選擇。一級古蹟澎湖天后宮是台灣歷史最悠久的媽祖廟。中央老街是澎湖最早發展的街道，古色古香的街道建築很適合漫步。四眼井旁的乾益堂中藥行已開業超過百年，來到這可以品嘗看看他們的藥膳蛋和豆干。北環北環線是澎湖經典旅遊路線，從馬公市區一路到最遠的西嶼燈塔（漁翁島燈塔），沿途有許多知名必去景點。下面依照從市區出發的經過順序介紹：◈ 後寮天堂路白沙鄉後寮村的天堂路，這幾年是越來越熱門的澎湖秘境景點。'''
```

本文取自於 https://yencheng0817.pixnet.net/blog/post/326445630

從文本中抽詞 -「地」景點

- Articut 也可以取得景點，使用 articut.parse()。

```
penghuLIST = articut.parse(inputSTR, openDataPlaceAccessBOOL = True)
```

執行結果
[“貝吉”,”目斗嶼”,...]

課間練習5

- 請使用課程練習1提及的那則娛樂新聞來分析裡面有哪些地點，也利用取得景點的語法，來看看裡面是否有特殊的景點。

從文本中抽詞 -「物」

- 從文本中抽取其意義指「物品」的詞彙組，這個功能其實和前一週提到的「抽出名詞」是一樣的。

```
baseballNounLIST = articut.getNounStemLIST(baseballResultDICT)
print(baseballNounLIST)

basketballNounLIST = articut.getNounStemLIST(basketballResultDICT)
print(basketballNounLIST)
```

棒球執行結果

```
['主審', '優勢', '分', '勝利', '安打', '局面', '意識地', '手', '打線', '打者', '投手', '此戰', '比賽', '滿壘', '球', '球隊', '登板', '眼', '角滑球', '觸身球', '身肘', '身體', '車輪戰', '追平比數', '這球', '陽春砲', '領先']
```

籃球執行結果

```
['中', '人', '勝利', '單場', '場', '布克', '延長賽', '戰', '攻勢', '機會', '次', '波格丹諾維奇', '籃板', '階段', '霸王']
```

課間練習6

- 請使用課程練習1提及的那則娛樂新聞來分析裡面有哪些「物」。

討論

- 剛剛在不同的練習中練習如何取得人事時地物，練習到現在，大家覺得電腦幫你挑出人事時地物，這些特徵詞對分類文本有幫助嗎？會怎麼幫助你呢？

人事時地物特徵詞應用 brainstorm

- 從一篇關於「洗錢」的文章中可以直接取得人名，接下來可以分析誰是犯人，誰是檢察官，取得的人名可以繼續做後續分析。
- 從一篇長篇幅的故事，例如水滸傳，可以取得每個章節的事件，藉此可以用最短時間整理每個章節的大意。

(…… 歡迎分享你的想法)

作業: 股市文本分析

- 作業敘述:

在處理股市文本任務中，我們傾向將文本分成2種:

1. 描述跌的文本
2. 描述漲的文本

而這次的任務裡，我們從聯合新聞網找了近期的股市新聞 (20篇)，希望可以透過我們學到的抽取特徵詞的技術，幫我們順利辨認出有關於敘述股市漲的文本是那些，讓我們直接開始吧!

作業: 股市文本分析

任務一: 思考時間

- 從文本中我們可以觀察出以下幾個關鍵：
  - 首先我們會關心這是在說哪一支股票。
  - 第二部分我們會好奇這支股票是漲還是跌。
- 那我們是怎麼判斷文章是再說漲還是跌呢?
  - 很明顯這時候我們無法依賴單純斷詞後的詞頻。
  - 我們先用語感來觀察資料，你覺得那些句子透露出一篇文章的漲跌線索呢?

(文本在下一頁)

任務1文本

元太電子紙商機起飛 股價挑戰歷史新高

電子紙大廠元太（8069）因市場看好電子紙閱讀器與筆記本，以及電子紙標籤（ESL）商機，近期股價持續走揚，今（6）日盤中高點來到84.5元，上漲7元、漲幅達9%，近期有望挑戰歷史最高價85.6元。元太今年前5月累計合併營收達71億元，年增26.5%，是自2017年轉型100%電子紙製造商後的同期新高。元太指出，疫情加速電子貨架標籤裝機潮，公眾顯示器及物流應用同步升溫，都讓元太今年客戶端的需求大幅成長。時序進入下半年，第3季歐美主要市場迎來解封開學季，元太也擴增全新彩色電子紙技術產能，電子紙閱讀器和電子紙筆記本終端產品尺寸放大、功能更強，並有手寫功能成為標配，可望在供需兩端正面影響下帶起一波換機潮；第4季還有黑色星期五及聖誕假期購物旺季，電子紙產品終端銷售暢旺，將使元太下半年業績逐步升溫。

文本來源：
https://udn.com/news/story/7253/5581639

作業: 股市文本分析

任務二之一: 抽取需要的特徵詞

- 在這個任務中，請你根據所學過的抽取特徵詞的方法，包含運用
  - TFIDF值
  - 抽取名詞
  - 抽取動詞
  - 抽取事件

這四種技術，分別以第一則新聞去嘗試，之後綜合比較，你覺得要用哪個方法來判斷一篇股市新聞的漲跌比較好呢？

作業: 股市文本分析

任務二之二: 根據詞性自定義程式

- 在這個任務中，我們希望自己透過對詞性標記的觀察，留下標記為 UserDefined, ACTION, ENTITY, VerbP 的詞，並且只留下句子 (個人定義句子為有 Entity/UserDefined跟Action)。
- 希望透過這樣子的方式我們可以過濾目前不需要的元素，但是又保持句子的概念，如此一來，當我們發現"股價"跟"走揚"出現在同一個句子時，我們就可以比較放心認定他是一個有關股市上漲的句子了。

作業: 股市文本分析

任務二之二: 根據詞性自定義程式產出範例

```
[['市場', '看好', '電子紙', '閱讀器', '筆記本'],
 ['股價', '持續', '走揚'],
 ['日盤', '高點', '來到'],
 ['漲幅', '達', '9%'],
 ['挑戰', '歷史'],
 ['元太', '累計', '合併', '營收', '達'],
 ['增', '26.5%'],
 ['轉型', '100%', '電子紙', '製造', '商', '同期'],
 ['元太', '指出'],
 ['疫情', '加速', '電子貨', '架標籤', '裝', '機潮'],
 ['公眾', '顯示器', '物流', '應用', '同步', '升溫'],
 ['讓', '元太', '客戶端', '需求', '成長'],
 ['時序', '進入'],
 ['市場', '迎來', '解封', '開學'],
 ['元太', '擴增', '全新', '彩色', '電子', '紙技術', '產'],
```

作業: 股市文本分析

任務二之三: 探索 `articut.getContentWordLIST()`

- 這是一個超便利小工具，如果前面對你而言有點太難，在 articut 裡面也有一個類似的小工具，我們來嘗試看看吧!!
- `articut.getContentWordLIST()` 回傳結果範例:

```
[[(14, 17, '電子紙'), (46, 48, '大廠'), (76, 78, '元太')],
 [],
 [],
 [],
 [(43, 45, '市場'),
  (76, 78, '看好'),
  (106, 109, '電子紙'),
  (138, 141, '閱讀器'),
  (208, 211, '筆記本')],
```

作業: 股市文本分析

任務二之四: 思考時間

- 請回想一下之前的思考活動，我們認定有一些句子有助於我們理解一篇文章是描述漲還是跌。
- 試著觀察是那些字詞讓我們有這樣的感受呢?
  - 例如在第一句我們發現是「股價」和「走揚」讓我們有這樣的認知，那我們是不是能夠建立一個列表，把有關股價上漲的字詞都放進去。
  - 如果列表中的詞出現了，我們就可以暫時認定文章跟這方面有關係；如果這樣的句子越多，我們就越能肯定這個文章跟股市上漲有關。

作業: 股市文本分析

任務三: 開始判斷文本

- 在這個任務中，我們開始要將我們在思考時間所累積的想法實踐出來，我們首先要先藉由人工觀察，建出一個有關於描述股市漲的詞列表，之後將它和新聞比對以得到結果。

作業: 股市文本分析

- 目標: 我們希望給每個文本一個分數，分數越高表示它越可能跟股票上漲有關。
- 設計原理: 我們先找句子，它有包含跟上漲相關的詞，我們就加上一分。
- 最後我們以句子為單位計分，用**句子數量**進行平均。

作業: 股市文本分析

假設有個文本是:元太電子紙商機起飛股價挑戰歷史新高

- 斷詞之後，我們發現「商機」、「股價」、「起飛」、「新高」跟股價上漲比較有關係。
- 動詞比較有明顯股價上漲的感覺，因此先加上一分給動詞（起飛）。
- 接著再看看名詞，有三個所以加上三分。
- 最後除以句子數量(1句)，所以分數為(1+3)/1=4。

作業: 股市文本分析

- 範例結果

Title	Content	go_up_score
元太電子紙商機起飛 股價挑戰歷史新高	電子紙大廠元太（8069）因市場看好電子紙閱讀器與筆記本，以及電子紙標籤（ESL）商機，近期...	0.256
台股再登新高後壓回 航海王權證最熱門	台股今（6）日早盤在傳產與金融族群支撐盤勢，多頭指標—海運股買盤持續湧入下，指數一度登上「...	0.087
聯詠6月、Q2營收同創新高 上半年應可賺回兩個股本	驅動晶片廠聯詠（3034）今（6）日公告6月合併營收續升至115.8億元，較上月微增1.2%...	0.469
台股衝關萬八終場收跌6.26點 三大法人賣超33.09億	台股今（6）日開高後一度衝上18,008.37點，再創歷史新高，首度越過18,000點大關...	0.207
高端起落坑殺人 立委籲加嚴炒股查緝	「股票炒半天，大起大落、暴漲暴跌，被有心人賺走錢，散戶什麼都不剩！」台灣民眾黨立委張其祿說...	0.023
除權息遞延衝擊！0050將除息0.3元、殖利率0.2%	「國民ETF」元大台灣50（0050）今年下半年的配息出爐，元大投信公告，0050每單位擬發...	0.114
陽明5月獲利240.35億元 EPS再創新高達3.15元	陽明海運（2609）今（6日）公布5月獲利自結獲利，再度寫下新高記錄，一個月輕鬆賺進百億元。...	0.188
國巨第二季營收季增逾16% 並較去年成長逾一倍	被動元件龍頭國巨（2327）今（6）日公布6月自結合併營收為95.05億元，單月營收較上月增...	0.333

作業: 股市文本分析

- 小結:

從以上的任務，雖然只是一個小嘗試，但我們可以發現，分數較低的文本的確就是跟股票漲價比較不相關。如下所示，一則是有關法條，一則是有關個人投資的失敗，一則是有關於外商投資策略，

Title	Content	go_up_score
高端起落坑殺人 立委籲加嚴炒股查緝	「股票炒半天，大起大落、暴漲暴跌，被有心人賺走錢，散戶什麼都不剩！」台灣民眾黨立委張其祿說...	0.023
他當沖狂賺幾萬塊「想玩大一點」 交易金額提高300萬卻GG了	台股今（6）日早盤指數一度登上「萬八」大關，再創歷史新高，而近年台股創高也引起許多投資客走上...	0.037
台積長線股價 外資喊千元	外電報導，英特爾將包下台積電（2330）3奈米產能，外資摩根大通、華興資本及瑞士信貸昨（6）...	0.044

作業: 股市文本分析

- 結語:

當然這個方法還是有不少缺陷，或許我們可以設計另一個功能判斷文本是「跌」的分數，然後去和「漲」的分數進行比較。

又或是有更好的計算分數的方式。

那這部分就交給有興趣延伸的人更進一步探討囉! 加油!

文本分析與程式設計

Week04

學習目標

- 學習使用取用特徵詞的工具以分類文本。

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(有經過編輯)

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，我們知道那些特徵詞可以區分棒球和籃球。
- 我們使用過的特徵包含:

1. 將所有的詞進行 TF-IDF 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:

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。

來源:https://newtalk.tw/news/view/2021-05-06/570369

- 我們希望電腦回答我們，這篇文本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」)

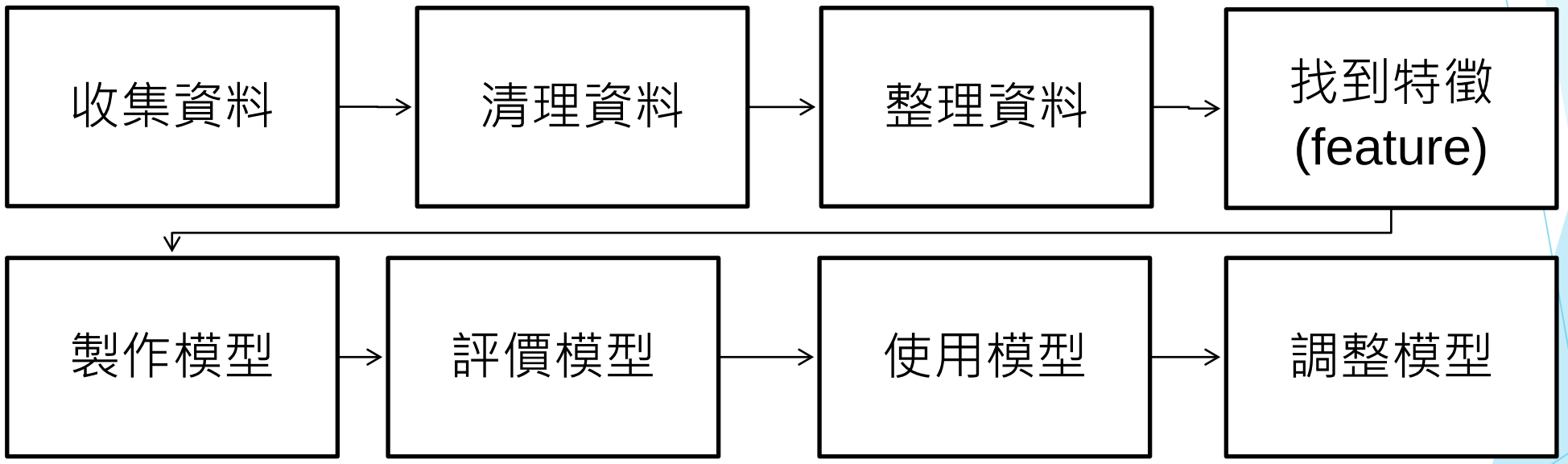
課間練習2

- 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

- 做文本分析時，我們通常依下列流程。



這些步驟是什麼意思呢？

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

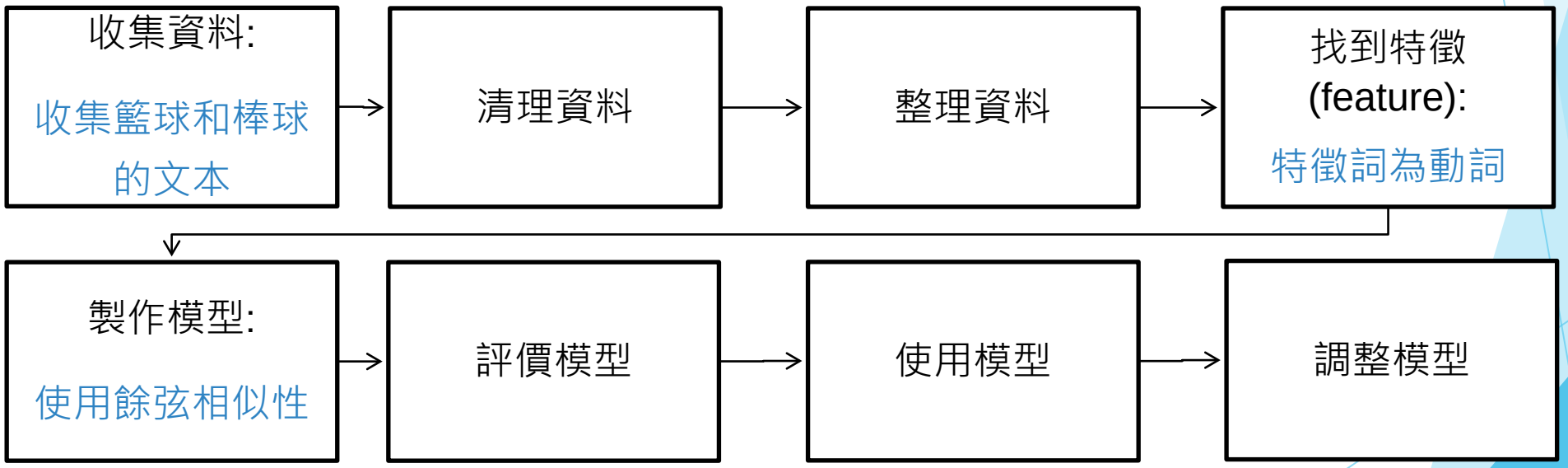
文本分析流程

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

- 我們將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，我們經常會把文字轉換成數字，以利計算。
- 當我們要把文字轉換成數字向量，就可以利用數學公式來計算他們「相不相似」。
- 「餘弦相似性」(cosine similarity) 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

- 在「解讀」餘弦相似性時需要特別注意：

1. 文字到底是怎麼轉成數字？

- 很多時候這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. 拿什麼文本去訓練？

- 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，來檢視兩個文本是否相似。

實作範例 – 動詞為特徵詞

- 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 != []:
            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

```
# 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

- 我們將一篇「不知其類別」的文本作為「測試文本」，也用一樣的步驟取出它的「動詞列表」。

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

- 目前流程

  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：我們分成基準資料 (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：基準資料和測試資料都用 `getVerbStemLIST()` 找到動詞特徵詞。

實作範例 – 動詞為特徵詞

- 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 需要從 collections 套件匯入。
  - counterCosinSimilarity() 是我們自己寫出來的函式。

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)
```

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
    '''

    #將兩個dictionary的keys合併出來成set的格式
    terms = set(counter01).union(counter02)

    #將兩個文章傳換成向量，數值表示不同字的頻率，然後做點積
    dotprod = sum(counter01.get(k, 0) * counter02.get(k, 0) for k in terms)
    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

- 我們以 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。

```
# 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。

未知文本

```
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。
"""
```

課間練習3

- 這個結果和你們自己討論出來的結果一致嗎？
- 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 某種程度便算是在評價模型。得到結果之後，我們檢視這個結果是不是對的。
- 目前我們文本量非常的少，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地正不正確。有一個簡單的方式稱為混淆矩陣(confusion matrix)。

Confusion matrix 混淆矩陣

- 混淆矩證可用來比較人類和電腦的判斷結果。
- 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：

	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

- 我們以人類判斷為正確判斷的話，以下兩種代表電腦也判斷對了。
  - True Positive (TP): 人類和電腦都覺得是。
  - True Negative (TN): 人類和電腦都覺得不是

	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

- 以下兩種情況就是電腦搞錯了
  - False Positive (FP): 人類判斷不是但是電腦判斷是
  - False Negative (FN): 人類判斷是，但電腦判斷不是
- 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

- 常見指標：
  - 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) ：這個模型有多少判斷（包含「是」「否」）是正確的
  - 算法 (TP + TN)/全部
- 精確率 (precision)：電腦說「是」，有多少比率真的「是」
  - 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)：「是」的樣本中有多少比率電腦說「是」
  - 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

- recall 和 precision 看事情的角度不太一樣。
- recall 和 precision 的比率會受到「電腦判斷錯誤數量」的影響。
  - Precision = TP / (TP + FP)
  - Recall = TP / (TP + FN)
- recall 和 precision 只是看到其中一個角度，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 (TP)	False Positive (FP)
電腦判斷 不是	False Negative (FN)	True Negative (TN)

Confusion matrix 混淆矩陣延伸閱讀

如果想要更了解以上內容可以讀

- 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
- 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

- 人類自己判斷的結果和電腦計算後的結果來做比較，這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滿足期待，那麼就需要作調整模型和更新模型。

- 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

- 請依照以下步驟，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。

- 透過上面步驟，未知文本是哪一種文本呢？
- 你覺得可以透過「名詞」和「動詞」來分類文本嗎？

課間練習5

- 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。
- 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

- 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。
- 試著解釋你的分類依據。
- 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:

斷詞工具:

- `resultDICT = articut.parse(inputSTR)`
  - 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:

目的	函式
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)`
找出地點	`articut.getLocationStemLIST(resultDICT)`
找出人名	`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，我們有多種的分析方式，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:
  - 選用中文文本
  - 請蒐集起碼兩種類的文本各 10 篇以上，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。
  - 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!