刀工備料:
CoreNLP與自動詞意標記

曾昱翔

2020/12/12 @ HOCOR 2020

大綱

- 準備
- 資料準備
  - 資料擷取、文字編碼、檔案儲存
- 資料前處理
  - 資料前處理(全形半形、標點符號等)、資料結構
- 自然語言處理
  - Deep-learning-based
  - CoreNLP 斷詞、詞類標記、詞意消岐
  - 中文詞彙網路

刀工備料

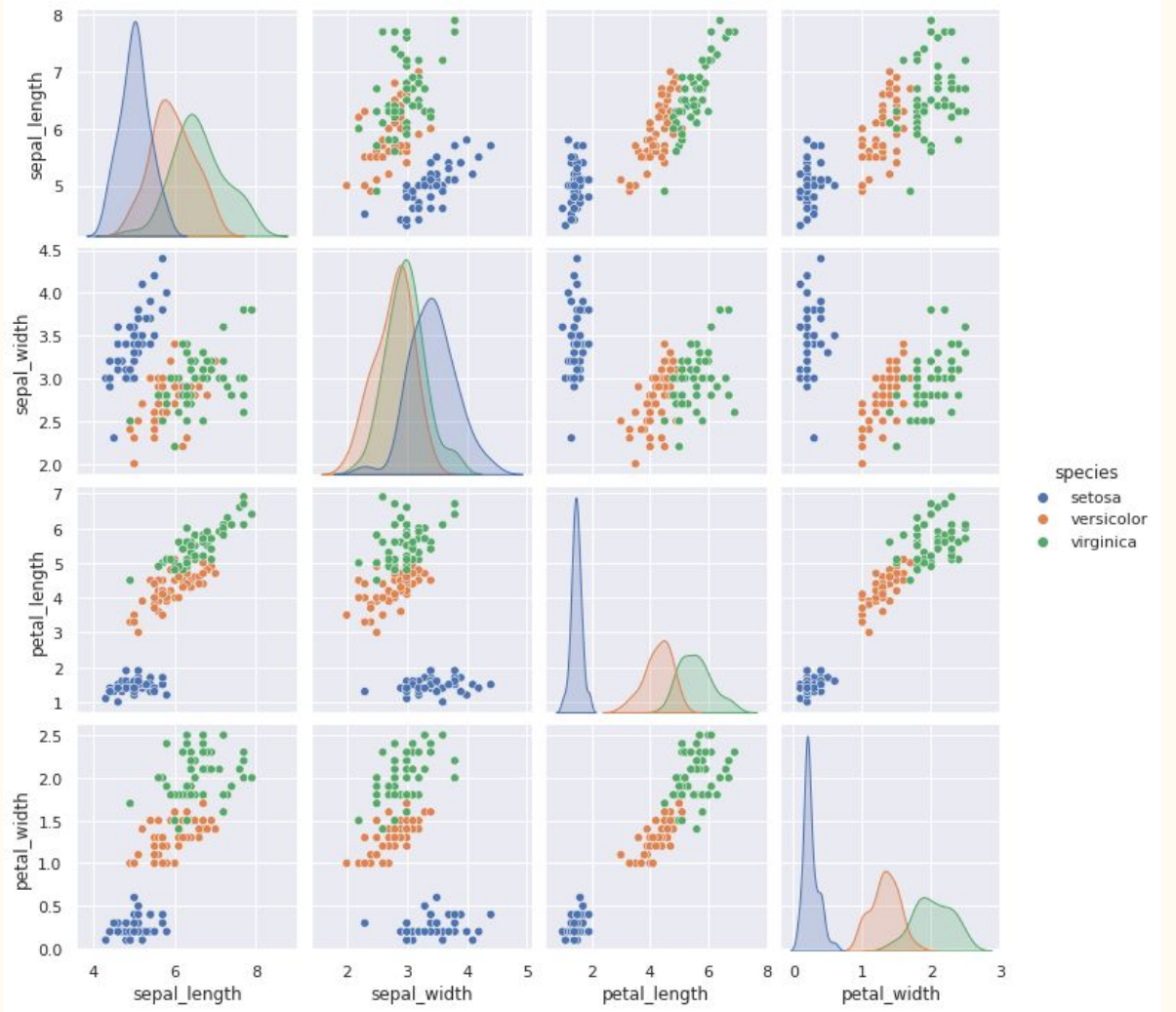
刀工備料

- 語言是超級複雜的非結構化資料。
- 就跟料理一樣，所有食材、調味料需要若合符節地搭配在一起，我們才會覺得那是道菜。
- 同樣地，我們怎麼切語料，怎麼搭配不同訊息，決定我們從語料中可以看到什麼。那是分析和整合的過程。

（韭菜）切得愈細，他釋放出來的味道愈多，好比如果你像切蔥花一樣，切得很細很細，...他韭菜的味道就很濃郁。《型男大主廚》

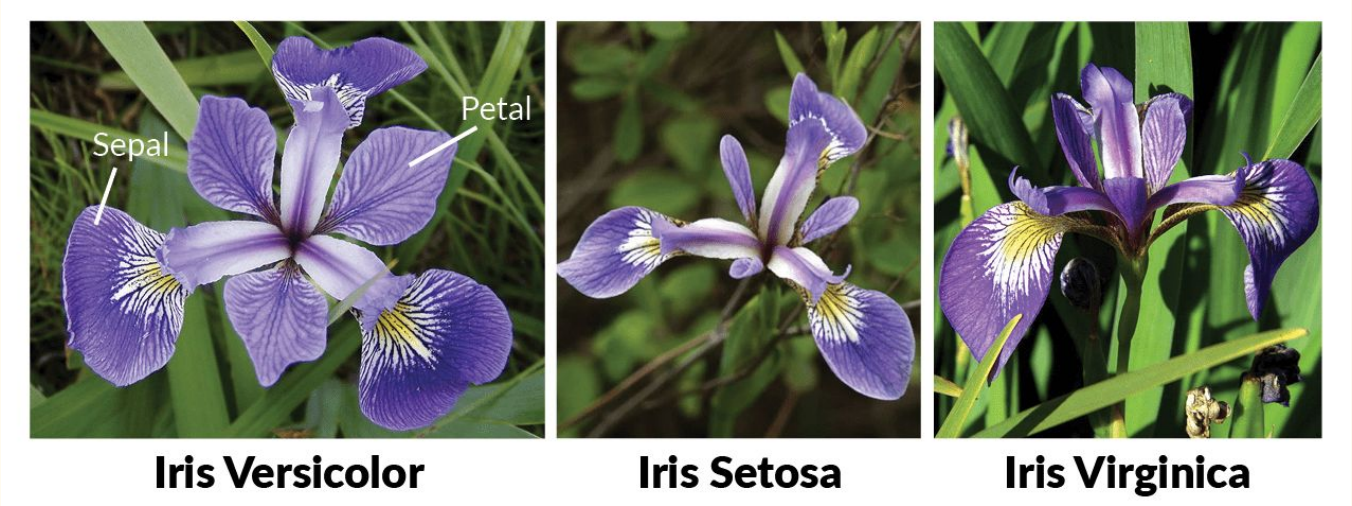
語料處理

- 語料處理，或文字處理，本身是一個困難的工作。
- 語料本身不是結構性資料。
- 結構性資料已經有變項，思考脈絡清楚很多。



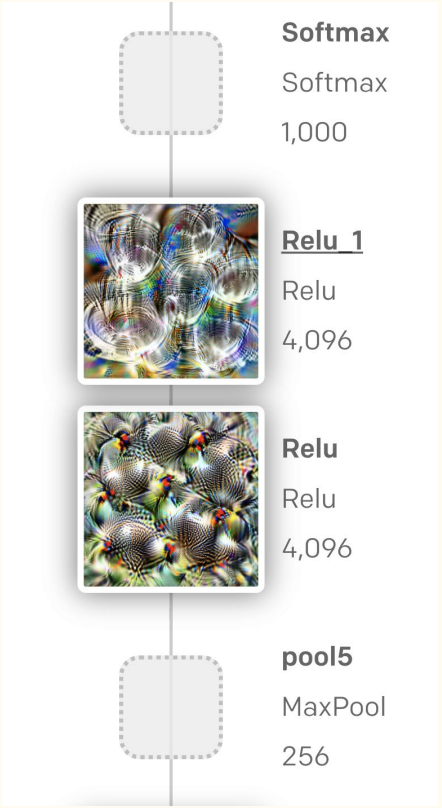
語料處理

- 如果iris資料是非結構性的，意思就是我們要從以下圖片中，找到「有意義」的特徵，讓我們描述三種不同的花。



結構化

- 人們習慣用符號來思考。
- 電腦不需要結構化，人需要。(c.f. 電腦視覺，甚至 BERT)。我們永遠會好奇，文本裡發生什麼事。
- 結構化的好處，是分類問題會變得非常簡單，甚至線性，(SVM on iris: ~.95 Acc)。
- 更大的好處是，模型變得很透明，因為模型就是用人類可以理解的符號做判斷。



語料處理

- 語言本身就是複雜非結構化的, 所以任何試圖把文字結構化(或向量化)的方法, 都只是一種「表達方式」。
- 語料處理, 就是過去傳承下來如何把語言結構化的方法。對比於2015年後, 深度學習對語言有非常不一樣的洞見。
- 語料處理可至少分為幾個步驟:
  - 資料取得:從各種來源取得文本
  - 資料前處理:將文本「清乾淨」
  - 語言前處理:將語言材料自動標記上「基本訊息」

刀工備料：Hands-on

Colab notebook

資料取得

文本資料來源

- 文本資料在現代當然唾手可得, 但有些文字內容還是很難取得
    - 網路語料(討論區、新聞網站、Wikipedia等)
    - 文件檔案(Word檔、PDF檔)
    - 掃描、圖檔(以前的剪報等)
    - 其他:訪談逐字稿、手寫資料(??)
- 拜現在技術之賜, 愈來愈多工具可以自動化。
- 但是, 重要概念還是, 擷取後的資料是給電腦的, 人看得懂外, 電腦也要看得懂！

網路語料

- 最難的地方不是技術，而是怎麼知道我們拿的材料是我們應拿的。
- 官方提供的API
  - 較少見，但較「安全」
  - 在台灣的網路環境中還是比較少看到。
  - Wikipedia就有提供API
- 寫爬蟲
  - 請觀察目標網站到底已經多不希望有爬蟲。
    - robots.txt，使用者協議，或甚至CAPTCHA
  - 寫個有禮貌的爬蟲，請爬自己需要的東西，且有效率、溫和地做這件事。
    - 限制速度，快取已經抓到的東

文件檔案

- 許多以前的語料大多整理成Word或PDF, 尤其在資料科學未興起前, 很少資料會存成csv、xml、json等方便格式。
- 許多政府或研究單位的資料常是如此。
- 在簡單的狀況下, Python有pyPDF2和python-docx可以幫忙。
- 但大部分狀況下, Word由於裡面訊息太過混雜:尤其是表格、雙欄排版等。PDF檔也是, 而且PDF的內容有可能只是內嵌一張圖。
- 手動複製貼上(或工人智慧), 可能反而是最乾淨、快速、準確的作法。

掃描圖檔

- 很多文件是透過掃描成圖檔保存下來的。
- 以前遇到這種檔案，唯一方法就是請人繕打成文字檔。
- 但，如果圖檔很乾淨清晰，排版也簡單，那現在的電腦視覺或許可以幫忙。（例如，Google Cloud Vision）
- 如果圖檔模糊，或排版複雜（如報紙版面），請人打還是最快的方法。

檔案格式

- 但不管是現成爬回來的網路資料, 或者是用人工/工人智慧轉寫出來的語料, 這些語料都要先存成一些方便後續處理的檔案。
- 資料科學/語料處理中, 通常不會用文件格式 (docx/xlsx/pdf), 原因是
  - 太複雜, 甚至不透明:直接用記事本打開是無意義的。
  - 裡面記載許多電腦用不到的訊息:如排版、字型、字體大小等
- 適合語料處理的格式, 最好是簡單, 讀寫的時候不需要想太多:
  - 純文字格式:.txt
  - (逗號)分隔格式:.csv
  - 結構化格式:.json

純文字檔

- 可以用記事本打開, 不會有任何亂碼、直接可理解內容的檔案。
- 純文字檔在深度學習中有很多應用場景:
  - 例如, hugging face transformers中預設的輸入資料, 就是要求純文字檔
  - Gensim的語料庫格式, 也是要求純文字格式。檔案中, 一行就是一篇「文章」(document)
- 但在實際的使用過程中, 尤其是Windows使用者, 很容易遇到.txt打開是亂碼。
- 什麼是亂碼？

文字編碼

- 文字(符號)對電腦是非常奇怪的概念。電腦需要用處理數字的方式, 來處理文字。
- 基本上電腦需要有一個對應表, 把文字對應到數字(反之亦然)。
- 在英文裡, 這個對應相當清楚:
  - A: 65(0x41), a: 97(0x61)
- 在中文裡, 問題變得非常複雜。即便只在台灣, 就有兩種編碼:
  - Big5 (較早期的編碼)- 語: 0xBB79
  - UTF-8 (Unicode)- 語: 0xE8AA9E
- 現在大都建議用UTF-8, 他是目前大多數環境預設的文字編碼。
  - 😂: 0xF09F9882

文字編碼

- 亂碼是電腦用錯的編碼讀檔案中的中文, 例如用UTF-8讀Big5或用Big5讀UTF-8
- 避免亂碼方式:能選UTF-8就選UTF-8
- 最大的問題是繁體中文Windows:
  - 記事本或甚至Python都預設Big5編碼
  - Excel檔讀或存CSV時也預設Big5
- 在Python中, 記得在需要的地方加入encoding選項:
  - open("filename", "r", encoding="UTF-8")
  - pd.write_csv("filename.csv", encoding="UTF-8")

CSV/JSON 格式

- CSV和JSON都是較多格式訊息的純文字檔。
- CSV適合扁平的, 表格化的結構資料
- JSON適合表格的一個細格(cell)內有多個數值, 甚至又是另一個表格的巢套資料(nested)。
- CSV和JSON會受到資料/語料處理的歡迎, 因為處理相當直覺簡單, 在Python中:
  - CSV: csv (built-in), pandas
  - json: json (built-in)

我需不需要用到資料庫？

- 常見的資料庫包含MS Access, MySQL, SQLite, MongoDB等。
- 就檔案格式的角度而言, 資料庫是一種非常不透明的檔案格式。
- 資料處理初期, 也不建議先把東西丟到資料庫
  - 初期都是純文字檔。文字檔沒有結構, 資料庫也很難幫上忙。
  - 資料庫本身的複雜度, 反而增加資料處理過程中的困難。
  - 資料庫檔案很難轉移。如果開發環境不只一個, 要怎麼同步不同環境的資料庫, 或怎麼把資料庫設置在一個共用伺服器上, 都會需要額外的設定。

什麼時候資料庫很有用？

- 文件除了文本以外，有複雜的欄位資訊（作者、來源、媒體、頻道、瀏覽人數等）
- 文件很多，單純用檔案/目錄結構已經很難管理：＞10K個檔案
- 全部語料的量已經超出電腦記憶體負擔：1~2GB
- 語料庫需要多人共享，或到最後需要有語料庫查詢介面的時候

資料取得：Hands-on

Colab notebook

資料前處理

資料前處理

- 前處理的目的包含：哪些部分真的是要處理的語料？以及從檔案格式變成「適合處理」的資料格式
- 哪些部分真的是語料？
  - 空白、tab要怎麼處理？換行真的是換行嗎？
  - 文字表情符號（emoticon, e.g. XD, ^^||||）怎麼辦？
  - 表情符號（emoji, e.g. 😂😅😳😭😛）算嗎？
  - 英數符號呢？或者網址呢？
  - 還是, 最保守的, 只有「中文字」才算？

資料清洗

- 開始切語料的第一步。
- 通常這一步會用到正規表達式(Regex)來做資料前處理,
  - 例如濾掉所有網址
  - 移除所有非中文、標點訊息等
- Python的zhon套件, 提供很多常用的前處理正規表達式
  - zhon.hanzi.characters
  - zhon.hanzi.punctuation

語料格式

- 在Python裡, 通常建議把語料當成是一個list, 語料中的每篇文章是一個dict。
- 這樣做的好處是我們可以任意紀錄文章的各種資訊。同時, 紀錄各種語言自動標記資訊。


```
{
  "title": "#問 Dr. Martens 1460 鞋墊",
  "commentCount": 14,
  "likeCount": 3,
  "forumName": "穿搭",
  "gender": 0,
  "rawtext": "雖然知道可能會被..."
}
```

資料前處理：Hands-on

Colab notebook

語言處理管線

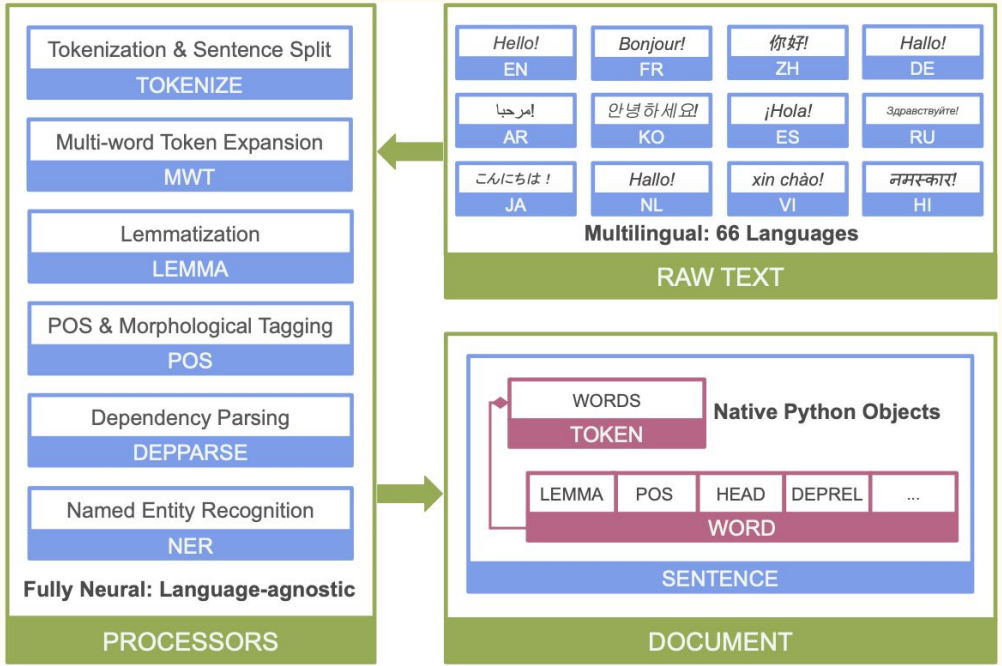
語言處理管線

- 準備好語料的資料格式後，我們要自動賦予語料一些基本的語言訊息。
- 中文裡最常用的訊息包含斷詞、詞類標記等。
- 視實際需求，往下也可以繼續做詞意標記。


```
{
  "title": "#問 Dr. Martens 1460 鞋墊",
  "commentCount": 14,
  "likeCount": 3,
  "forumName": "穿搭",
  "gender": 0,
  "rawtext": "雖然知道可能會被..."
  "text": [[
     {"word": "雖然", "pos": "Cbb"},
     {"word": "知道", "pos": "VK"},
     {"word": "可能", "pos": "D"},
     {"word": "會", "pos": "D"},
     {"word": "被", "pos": "P"}]]
}
```

語言處理管線

- 語言處理管線(Pipeline) 泛稱一系列從原始文字字串, 開始賦予語言結構的自動化工具。
- 中文的處理管線最重要的第一步是斷詞, 他完全決定了在Pipeline下游可以觀察到什麼語言現象。



中文斷詞的彈性

- 中文斷詞最大的問題不僅是斷詞錯誤或模糊:
  - 我住家裡:我/住家/裡
  - 我回家教學生:我/回/家教/學生 vs. 我/回家/教/學生
- 更大的問題是「詞」層次代表語言表達的概念, 而概念是很容易變動的
  - 專有名詞:腎/上/腺/皮質/素、鬼/滅/之/刃、后/翼/棄/兵
  - 新詞(用法):咩/噗、像/極/了/愛情、芒果/乾
  - 詞彙變異:合/太/醬料、種花/電信
- 現在的常見做法, 是把這些詞當作領域知識, 用詞表加入斷詞。
- 但這件事永遠是困難的:如果我就是不知道有哪些新詞, 我怎麼知道要加入詞表？我加入詞表了, 他怎麼還是新詞？

語料探勘

- 如果真的有新詞, 而新詞真的很重要很常出現, 那在語料中應該會有一點端倪。
- 最簡單的方法, 是找ngram。
- Ngram無法取代斷詞, 但可以在我們拿刀切下去之前, 讓我們看看資料本來的樣子。
- Ngram本質有點暴力, 所以如果語料小, 或許我們可以看看 4-gram, 萬一語料比較大, 至少我們可以看看2-gram
- Ngram的結果不會是詞, 但如果在語料中有常見新詞, 那2-gram很可能就會發現, 例如「萊克多巴胺」

簡單文本處理

- 如果我們有很完整的詞表, 斷詞的問題相對簡單。
  - 最大匹配法:「中文/自然語言/處理」
  - 在全部是已知詞的狀況下, 最大匹配法就已經有大於9成正確率。
  - Jieba的機率模型可以斷得更好
- 但完整詞表通常很難取得。所以如果開發環境以及計算資源允許, 我們通常可以選擇其他工具來做斷詞
  - 中研院CoreNLP
  - LOPE DistilTag
  - 其他還有spaCy和Stanford stanza

詞類標記

- 當語料被分成一個個詞彙之後，我們接著想知道每個詞彙在句子中所扮演的語法角色。
- 詞類訊息可幫助我們初步了解句子結構，也開始能區辨詞意。
- 中文有哪些詞類本身是一個研究問題，不同自動標記工具也會標出不同詞類。

我	打	打	電話
代名詞	動詞	動詞	名詞
我	一	打	鉛筆
代名詞	數詞	量詞	名詞

詞類標記集

- 中研院詞類(CKIP, DistilTag)

標籤	詞類	標籤	詞類
Na	一般名詞(如歌迷、舞台)	VA	動作不及物動詞(他在「唱歌」)
Nb	專有名稱(如人名)	VC	動作及物動詞(「表達」敬意)
Nep	指代定詞(如這、那)	VH	狀態不及物動詞(品質「優良」)
Neu	數詞定詞(如「二」個)	A	非謂形容詞(「共同」市場)
Nf	量詞(如一「條」)	Caa	對等連接詞(如與、和、及)
Nv	名物化動詞(如「主辦」單位)	Dfa	動詞前程度副詞(「最」近、很)

詞類標記集

- Penn Treebank (Stanza)

標籤	詞類	標籤	詞類
NN	一般名詞（如歌迷、舞台）	VV	一般動詞（他在「唱歌」）
NR	專有名稱（如人名）	VA	形容詞述語（品質「優良」）
DT	指代定詞（如這、那）	JJ	一般形容詞（「共同」市場）
CD	數詞定詞（如「二」個）	CC	對等連接詞（如與、和、及）
M	量詞（如一「條」）	AD	動詞前程度副詞（「最」近、很）

詞類標記工具

- Jieba只能提供粗略的詞類訊息
- 做語料分析時, 大多還是使用較有規模的自動標記工具。
- CKIPTagger和DistilTag都可提供中研院詞類標記集的詞類訊息。

命名實體辨識

- 除了詞類以外，很多應用真正關心的是這個句子裡面在「講什麼」。
- 命名實體辨識（Named Entity Recognition）可找出句子中的人名、地名、機構名或時間。
- CKIPTagger有提供此功能。

蔡英文	去年	12月	出席	台積電	尾牙	時	指出
Nb	Nd	Nd	VC	Nb	Nd	Ng	VE
專有名詞	時間詞	時間詞	動詞	專有名詞	時間詞	後置詞	句賓動詞
PERSON	DATE			ORG			

自動詞意標記

- 有了斷詞、詞類標記、和命名實體，可以解決很多問題
- 但更有趣的問題，是同樣的詞，同樣的詞類，他們的意思一樣嗎？
  - 如「吃」饅頭，「吃」敗仗
- 詞意消岐可以自動找出句子中的詞是對應到哪個詞意。

他	Nh	代指自己和對方以外的第三人。	吃	VC	比喻經歷後述負面事件。
吃	VC	使物體經過口中吞入體內。	了	Di	表示變化或出現新的情況，
饅頭	Na		敗仗	Na	
後	Ng	晚於特定參考時間點或事件。			

中文詞彙網絡

- 那「吃」有幾個詞意？(在CWN中，28個)

中文詞彙網路

- 中文詞彙網路(Chinese WordNet)是耗費將近20年的時間, 透過人力編輯、區辨、並建立詞意關係的語意知識資源。
- 中文詞彙網路在今年釋出程式介面(CwnGraph), 讓使用者可方便地用Python取得詞意資料。
- 搭配自動詞意標記工具, 讓詞意更容易進入一般語意處理流程。
- CWN的基本結構:
  - 詞(word)
  - 詞條(lemma)
  - 詞意(sense)
  - 詞意關係(semantic relations)

語言處理管線：Hands-on

Colab notebook

Wrap-up

- 語料處理的事前準備工作, 包括資料取得、前處理、語言訊息自動標記(處理管線)。
- 這些準備工作都是讓我們把非結構的複雜的文字材料, 切分成更細緻的語言結構, 讓語料不只是一串文字序列。
- 前處理後, 接著我們就能對語料進行索引(Session 2)。
- 透過索引後的語料, 迅速找到有趣的目標文本, 並對之進行語料查詢、分析(Session 4、5)。