

文本前處理：**Articut NLP** 系統與句法學

台師大通識教育課程

文本分析與程式設計

授課：卓騰語言科技 _ PeterWolf

N-Gram

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

uni-gram:

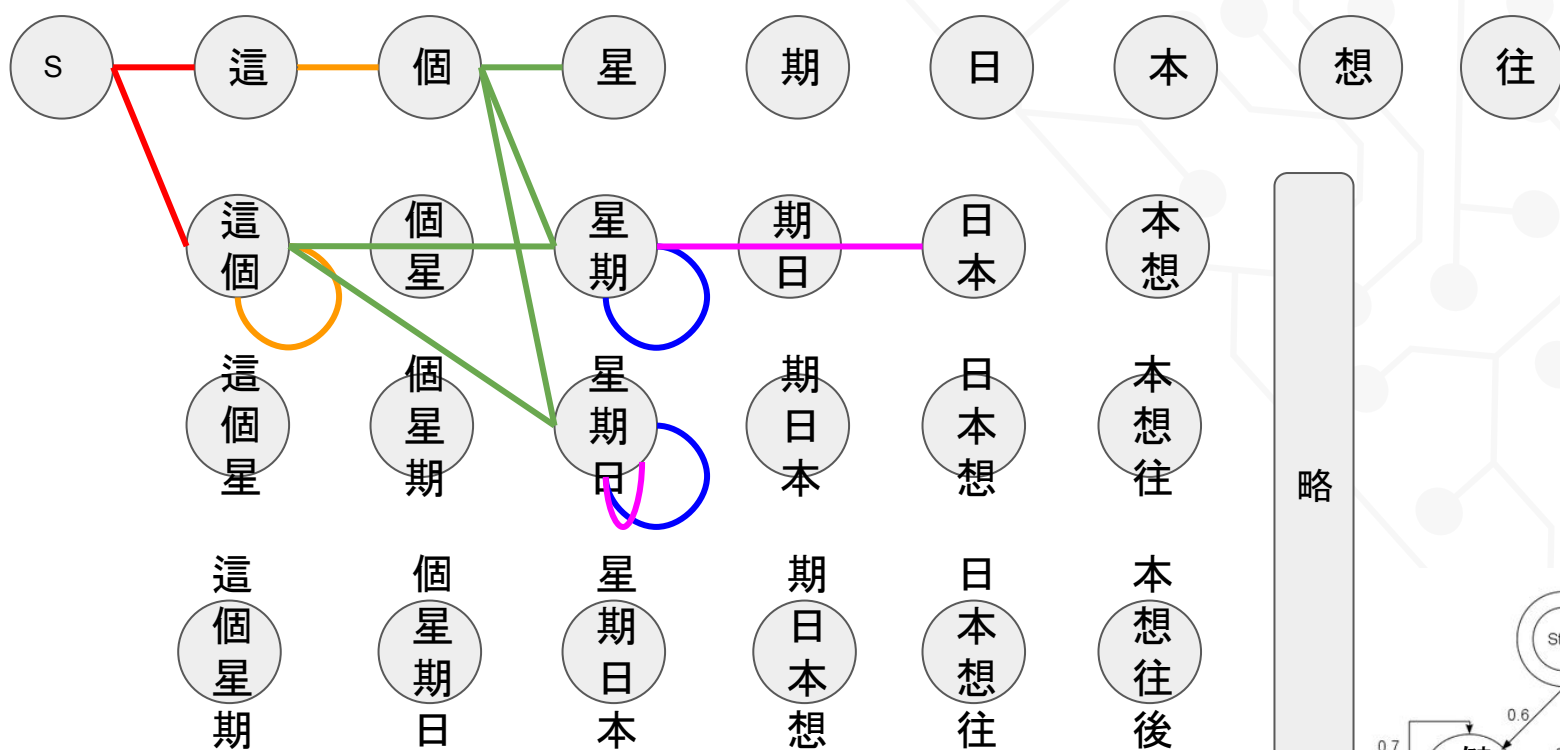
這/個/星/期/日/本/想/往/後/山/藥/師/佛/寺/去/世/人/罕/至/處/想/一/想/自/己/的/人/生

bi-gram:

這個/個星/星期/期日/日本/本想/想往/往後/後山/山藥/藥師/師佛/佛寺/寺
去/去世/世人/人罕/罕至/至處/處想/想一/一想/想自/自己/己的/的人/人生

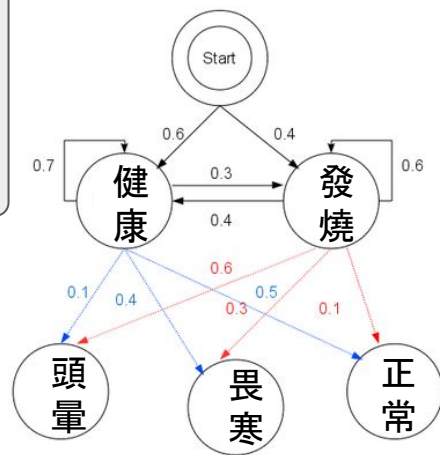
tri-gram:

...



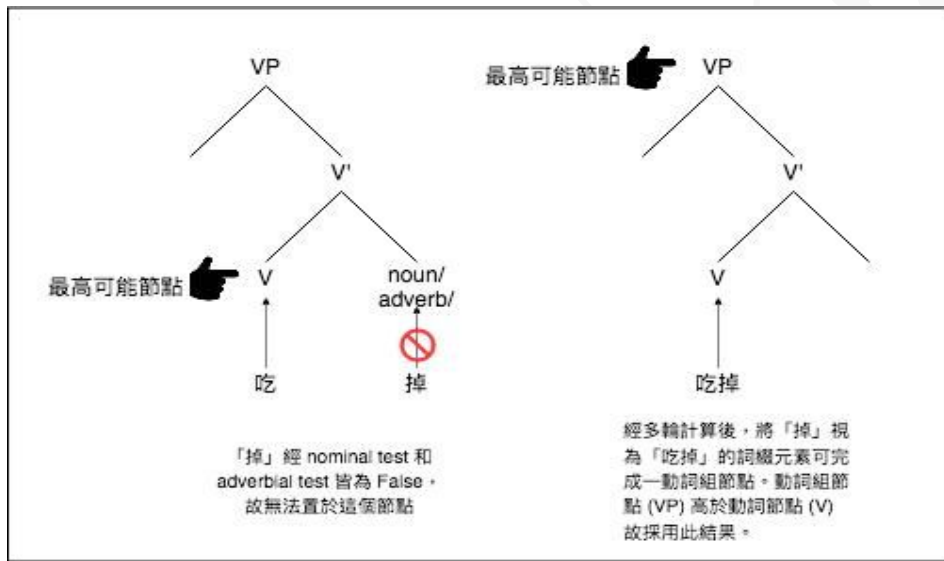
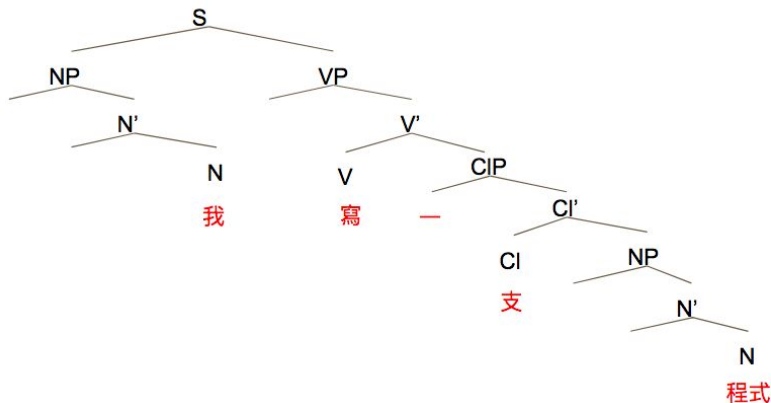
略

這/個/星...
 這/個/星期/日本...
 這個/星期/日本...
 這/個/星期日/本...
 這個/星期日/本...



依句法學原理設計的中文 NLP 系統

1. Articut 不靠資料訓練，而依句法結構計算斷詞、詞性及命名實體。因為不需資料訓練，對**隱私保護**具有先天的優勢。
2. CWS + POS + NER 同時完成降低因分段訓練造成的良率下滑問題，使最終應用產品良率基準提高。



Articut 運作機制：以動詞切分為例

ArticutAPI: All-in-one 的工具組

For Python3 Users: <https://github.com/Droidtown/ArticutAPI>

For other languages: POST API Document <https://api.droidtown.co/document/#articut-2>

❖ 安裝:

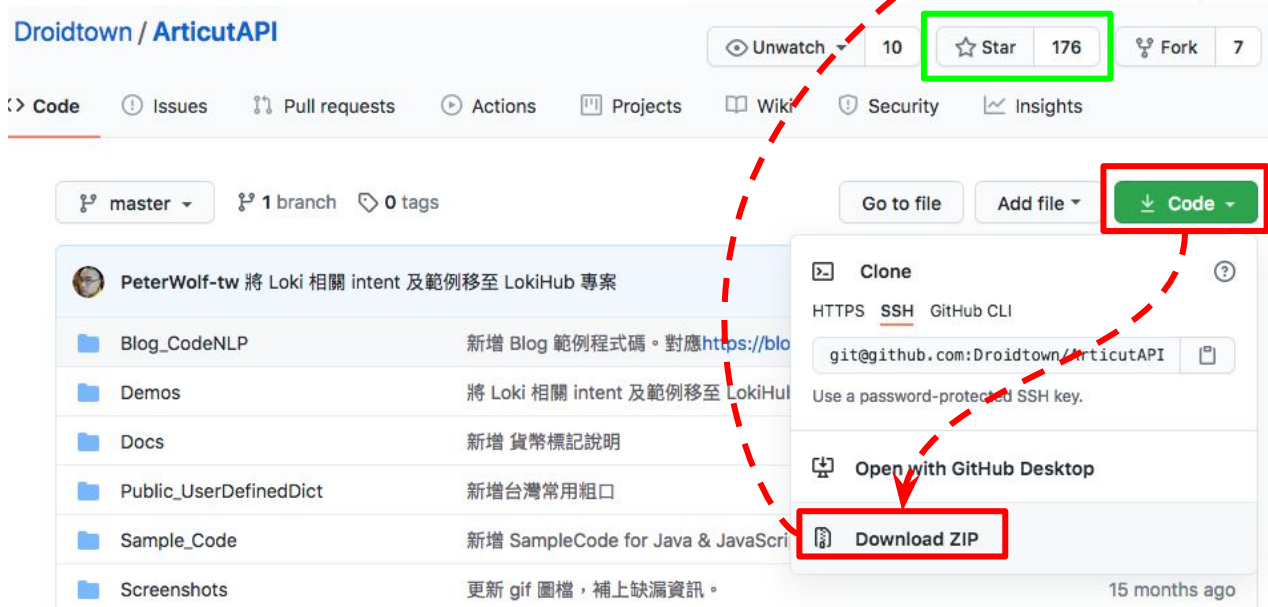
- 呃...不用安裝, 下載就好了 !:)

❖ 使用:

- lv1: 極致斷詞 (斷到「構詞內部」結構)
- lv2: 詞組斷詞 (斷到「詞組」結構)
- lv3: 各種「語意」相關資訊
- .getPersonLIST(): 取得全文中的「人名」
- .getVerbStemLIST(): 取得全文中的「動作」(動作即事件)
- .getNounStemLIST(): 取得全文中的「實體」(實體即參與者)
- ...族繁不備載

ArticutAPI 安裝

到這裡 : <https://github.com/Droidtown/ArticutAPI>
git clone <https://github.com/Droidtown/ArticutAPI>
或



Articut 基本操作

解壓縮，並做好更名了

從 ArticutAPI 目錄中，
載入 ArticutAPI 模組。

實體化 ArticutAPI 模組
中的 Articut() 工具，並
命名為 "articut"

準備要處理的文本字串

呼叫 articut.parse() 工
具來處理文本內容。

level 參數設定為 "lv2"
，處理完畢後結果存入
"resultDICT" 中。

處理結果

```
(vEnv_py36) peter@peter-G6:~/.../NTNU_TextProcessing/week09/example$ ls
ArticutAPI
(vEnv_py36) peter@peter-G6:~/.../NTNU_TextProcessing/week09/example$ ipython
Python 3.6.9 (default, Feb 12 2020, 21:15:06)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.3.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from ArticutAPI import ArticutAPI
No module named 'graphene'
Articut-graphQL requires 'graphene' module.
Please use pip/conda install graphene-python to install the module and reload Ar
ticutAPI.

In [2]: articut = ArticutAPI.Articut()

In [3]: inputSTR = "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"

In [4]: resultDICT = articut.parse(inputSTR, level="lv2")

In [5]: resultDICT
Out[5]:
{'exec_time': 0.029288053512573242,
 'result_pos': ['<ENTITY_DetPhrase>這個</ENTITY_DetPhrase><TIME_week>星期日</TIM
E_week><MODIFIER>本</MODIFIER><ACTION_verb>想</ACTION_verb><FUNC_inner>往</FUNC_
inner><RANGE_locality>後</RANGE_locality><ENTITY_oov>山</ENTITY_oov><ENTITY_noun
y>藥師佛寺</ENTITY_nouny><ACTION_verb>去</ACTION_verb><ENTITY_nouny>世人</ENTITY
_nouny><MODIFIER>罕</MODIFIER><FUNC_inner>至</FUNC_inner><ENTITY_nouny>處</ENTIT
Y_nouny><ACTION_quantifiedVerb>想一想</ACTION_quantifiedVerb><ENTITY_pronoun>自
己</ENTITY_pronoun><FUNC_inner>的</FUNC_inner><ENTITY_noun>人生</ENTITY_noun>'],
 'result_segmentation': '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',
 'this_sentence': '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'}
```


Articut 進階操作： 取得人名

呼叫 `articut.parse()` 工具來處理文本內容。

`level` 參數設定為 `"lv2"`，處理完畢後結果存入 `"resultDICT"` 中。

```
In [17]: inputSTR = """甲○於民國109年5月初某日，與少年彭倫（93年3月
...: 生，另由本院少年法庭審理）加入詐欺集團（下稱本件詐欺
...: 集團），彭倫隨後邀集丁○、高柏鈞、胡哲誠（前2人
...: 另經本院以109年度金訴字第141號判決處徒刑）等人加入
...: 該集團，由高柏鈞、胡哲誠擔任俗稱「車手」之提領詐欺
...: 贓款工作、丁○負責發放車手報酬予高柏鈞、胡哲誠，並
...: 透過通訊軟體Telegram（下稱Telegram）聯繫，甲○則依
...: 彭倫之指示，至胡哲誠、高柏鈞居住之位於桃園市楊梅區
...: 新農術6號之晚安旅館，喚醒並督促高柏鈞、胡哲誠準時前
...: 往指定地點即板橋火車站等候彭倫之指示，並於彭倫未
...: 能接聽Telegram之來電時，代為接聽電話。"""
```

準備好「待處理文本」

```
In [18]: inputSTR = inputSTR.replace("\n", "")
In [19]: resultDICT = articut.parse(inputSTR, level="lv2")
In [20]: nameLIST = articut.getPersonLIST(resultDICT)
```

文本前處理：
移除換行符

```
In [21]: nameLIST
Out[21]:
[[[],
[],
[],
[],
[(84, 87, '彭倫')],
[],
[],
[],
[],
[],
[],
[],
[],
[],
[],
[],
[],
[],
[(15, 18, '彭倫'), (137, 140, '丁○')],
[],
[(15, 18, '高柏鈞')],
[],
[(15, 18, '胡哲誠')],
[]]]
```

利用 `.getPersonLIST()`，將 `resultDICT` 交給它處理，取出所有的人名，並存入 `"nameLIST"` 中。

從原本的「被告：楊宇喬...被告：閻康合」，我們可延伸出其它的本案相關人士。

Articut 進階操作： 取得動作

呼叫 `articut.parse()` 工具來處理文本內容。

`level` 參數設定為 `"lv2"`，處理完畢後結果存入 `"resultDICT"` 中。

```
In [33]: inputSTR01 = " (中央社記者謝靜雯台北7日電) 開南盃大學棒球邀請賽冠軍戰，
...: 地主開南大學投手張鴻煜6局優質先發，團隊12安打線串連搶分，終場以7比1勝南
...: 華大學奪冠；季軍戰富邦公牛延長賽第10局靠再見保送以3比2擊敗台灣啤酒。"

In [34]: inputSTR02 = "Laird 和台灣好手潘政琮，都是並列63名，美國選手 Kirk也射下
...: 老鷹，不過，吞下4個Bogey，1個DoubleBogey，總桿數73，並列73名，墨西哥選
...: 手CarlosOrtiz，在最後一洞，有非常精彩的長推桿。"

In [35]: resultDICT01 = articut.parse(inputSTR01, level="lv2")
In [36]: resultDICT02 = articut.parse(inputSTR02, level="lv2")
In [37]: verbLIST01 = articut.getVerbStemLIST(resultDICT01)
In [38]: verbLIST02 = articut.getVerbStemLIST(resultDICT02)

In [39]: verbLIST01
Out[39]:
[[[],
 [],
 [],
 [(13, 14, '開'), (134, 136, '邀請')],
 [],
 [(121, 122, '投')],
 [(40, 41, '開'), (264, 265, '發')],
 [],
 [(126, 128, '串連'), (155, 157, '搶分')],
 [],
 [(102, 103, '比'), (213, 215, '奪冠')],
 [],
 [(105, 107, '延長'),
 (205, 206, '靠'),
 (233, 235, '再見'),
 (262, 264, '保送'),
 (343, 344, '比'),
 (397, 399, '擊敗')],
 []]

In [40]: verbLIST02
Out[40]:
[[[],
 [],
 [],
 [],
 [(114, 116, '射下')],
 [],
 [],
 [],
 [(13, 15, '吞下')],
 []]
```

準備好多篇「待處理文本」

利用 `.getVerbStemLIST()`，將 `resultDICT` 交給它處理，取出所有的動作，並存入 `"verbLIST"` 中。

假設有第三篇體育新聞出現，我們就能利用這些動詞來讓電腦判斷「這是哪一種運動」。

Articut 進階操作： 取得實體

準備好多篇「待處理文本」

呼叫 `articut.parse()` 工具來處理文本內容。

`level` 參數設定為 `"lv2"`，處理完畢後結果存入 `"resultDICT"` 中。

利用 `.getNounStemLIST()`，將 `resultDICT` 交給它處理，取出所有的**實體**，並存入 `"entityLIST"` 中。

假設有下一篇食記出現，我們就能利用這些**實體**來讓電腦判斷「這是哪一種風格的料理」。

```
In [3]: inputSTR01 = "古早味的粿條搭著一些油蔥與油菜，吃起來有夠過癮，另外它們的
...: 客家鹹湯圓也是經典，糯米香配上些許香菇、豬肉、芹菜，小編私心覺得好吃程度
...: 不在粿條之下，也非常推薦唷。"

In [4]: resultDICT01 = articut.parse(inputSTR01, level="lv2")

In [5]: entityLIST01 = articut.getNounStemLIST(resultDICT01)

In [6]: entityLIST01
Out[6]: [[(45, 46, '味'), (99, 101, '粿條'), (188, 190, '油蔥'), (257, 259, '油菜')],
[],
[],
[],
[(130, 132, '客家'), (160, 163, '鹹湯圓'), (226, 228, '經典')],
[],
[(12, 14, '糯米'), (131, 133, '香菇')],
[],
[(13, 15, '豬肉')],
[],
[(12, 14, '芹菜')],
[],
[(16, 16, '小編'), (123, 125, '程度'), (215, 217, '粿條')],
[],
[],
[]]
```

Articut 進階操作： 取得地點 Location

呼叫 `articut.parse()` 工具來處理文本內容。

`level` 參數設定為 `"lv2"`，處理完畢後結果存入 `"resultDICT"` 中。

利用 `.getLocationStemLIST()`，將 `resultDICT` 交給它處理，取出所有的地點，並存入 `"locLIST"` 中。

我們就能列出文本中所有的行政區名稱的列表。這麼一來，就能知道這篇文章在描述「什麼地方」

準備好「待處理文本」

```
In [33]: inputSTR
Out[33]: '嗨嗨!!這張地圖是不死兔這次到澎湖玩耍所記錄的景點與美食喔!\n上面的名稱都是直接用GOOGLE地圖搜尋名稱就可以找到的地點~\n看地圖知道位置之後~就可以更輕鬆的安排行程~不怕繞遠路啦!!\n歡迎有需要的朋友參考~~\n...\n如果要到市區玩~找不到停車位~其實可以先把車寄放在租車行這邊XD\n因為博明租車的附近超多景點~徒步就能到!!\n像是著名的老街天后宮與四眼井~只要一分鐘就能走到喔!\n四眼井就是有四個口的井~\n四眼井是台灣澎湖縣最老的古井，位於馬公市中央老街的最北端，建立於明代初期(西元1592年)，正式名稱為『四穴井』或『四孔井』。四眼井其實只是一個大井，出水量多，為避免民眾取水時不慎跌落入井中，早年以六塊花崗石條覆蓋，並用紅磚砌緣；但先前整修時已經改為
```

```
In [12]: resultDICT = articut.parse(inputSTR)
In [13]: locLIST = articut.getLocationStemLIST(resultDICT)
```

```
In [14]: locLIST
```

```
[[],
 [(10, 12, '這裡')],
 [],
 [(10, 12, '澎湖')],
 [],
 [(64, 66, '馬公')],
 [],
 [(39, 41, '臺灣'), (62, 65, '澎湖縣'), (86, 89, '馬公市')],
 [],
 [(36, 38, '澎湖')],
 [],
 [(44, 47, '澎湖縣')],
 [],
 [(117, 119, '西嶼')],
 [],
 [],
 [],
 [(10, 12, '西瀛')],
 []]
```


Articut 進階操作： 取得景點 Place

呼叫 `articut.parse()` 工具來處理文本內容。

別忘了加入 `openDataPlaceAccessBOOL` 的參數為 `True`。處理完畢後結果存入 `"resultDICT"` 中。

利用 `.getOpenDataPlaceLIST()`，將 `resultDICT` 交給它處理，取出所有的景點，並存入 `"openLIST"` 中。

準備好「待處理文本」

```
In [33]: inputSTR
Out[33]: '嗨嗨!!這張地圖是不死兔這次到澎湖玩耍所記錄的景點與美食喔!\n上面的名稱都是直接用GOOGLE地圖搜尋名稱就可以找到的地點~\n看地圖知道位置之後~就可以更輕鬆的安排行程~不怕繞遠路啦!!\n歡迎有需要的朋友參考~~\n...\n如果要到市區玩~找不到停車位~其實可以先把車寄放在租車行這邊XD\n因為博明租車的附近超多景點~徒步就能到!!\n像是著名的老街天后宮與四眼井~只要一分鐘就能走到喔!\n四眼井就是有四個口的井~\n四眼井是台灣澎湖縣最老的古井，位於馬公市中央老街的最北端，建立於明代初期(西元1592年)，正式名稱為『四穴井』或『四孔井』。四眼井其實只是一個大井，出水量多，為避免民眾取水時不慎跌落入井中，早年以六塊花崗石條覆蓋，並用紅磚砌緣；但先前整修時已經改為
```

```
In [24]: resultDICT = articut.parse(inputSTR, openDataPlaceAccessBOOL=True)
```

```
In [25]: openLIST = articut.getOpenDataPlaceLIST(resultDICT)
```

```
[],
[(17, 21, '跨海大橋'), (159, 166, '易家仙人掌冰店')],
[],
[],
[],
[],
[(34, 38, '跨海大橋')],
[],
[(70, 74, '二崁聚落')],
[]]
```

配合前一段的 `.getLocationStemLIST()`，我們就能讓電腦學會「什麼地方有什麼景點」了！

更多 Articut 進階操作 (就說族繁不備載咩)

調用 WikiData 資料庫

Articut 可取其條目 (Label) 的資訊，並標記為 <KNOWLEDGE_wikiData>。

```
#允許 Articut 調用 WikiData 字典，列出所有 WikiData 條目名稱的字串。
inputSTR = "我在瓶子里發現兩隻邊緣真亮羽水蚤。"
result = articut.parse(inputSTR, wikiDataBOOL=True)

wikiDataLIST = articut.getWikiDataLIST(result)
print("## WikiData:")
pprint(wikiDataLIST)
```

Local RE (台灣地址)

地址，是一串的字符，內含國家、省份、城市或鄉村、街道、門牌號碼、屋邨、大廈等建築物名稱，或者再加樓層數目、房間編號等。一個有效的地址應該是獨一無二的，才能讓郵差等物流從業員派送郵件，或者上門收件。

羅斯福路四段，指的是地理上的區域，而不是一個地址，因為光是看到「羅斯福路四段」並無法得知正確的所在位置。

X 台北市大安區羅斯福路四段

羅斯福路四段1號，指的是地理上的絕對位置，並可正確的知道所在位置。

O 台北市大安區羅斯福路四段1號

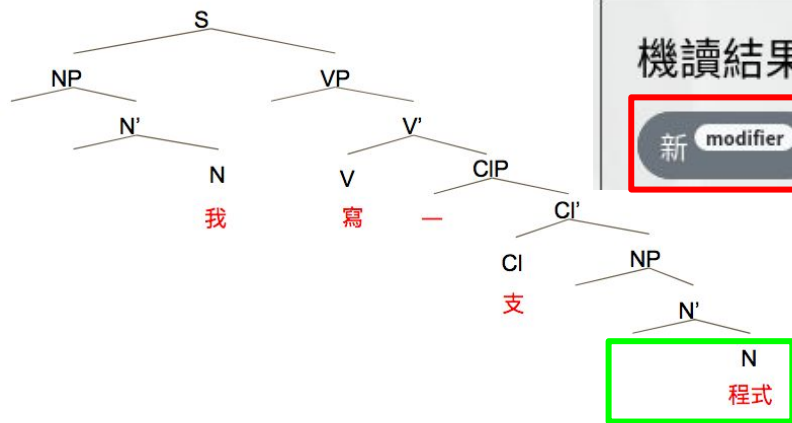
```
inputSTR = "台北市大安區羅斯福路四段1號"
result = articut.parse(inputSTR)
```

```
#列出所有的台灣地址
addTWLIST = articut.getAddTWLIST(result)
print("## Address:")
pprint(addTWLIST)
```

```
#使用 localRE 工具取得地址分段細節
countyResult = articut.localRE.getAddressCounty(result)
print("## localRE: 縣")
pprint(countyResult)
```

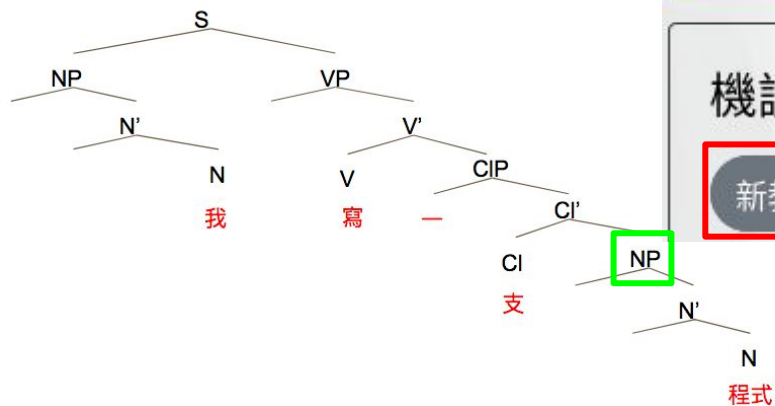
```
cityResult = articut.localRE.getAddressCity(result)
```

更多更多 **Articut** 進階操作 (lv1, lv2, lv3)



The screenshot shows the Articut web interface. At the top left is the Articut logo. To its right are links for "設定" (Settings), "lv1 極致斷詞" (lv1 Ultimate Word Segmentation), and "景點資料庫" (Landmark Database). Below these is a "免費字數: 1526" (Free characters: 1526) indicator. A red box highlights the "lv1 極致斷詞" button. Below the header is a text input field containing "新教練機預計明年進入量產" (New coach machine expected to enter mass production next year). Below the input field is a "機讀結果: 新/教練機...(more)" (Machine reading result: New/coach machine...(more)) section. A red box highlights the "新" (New) and "教練機" (Coach machine) buttons, which are labeled "modifier" and "nounhead" respectively. Other buttons in the row include "預計" (Expected), "明年" (Next year), "進入" (Enter), and "量產" (Mass production), each with a part-of-speech tag like "verb", "time", or "verb". A red arrow points from the "lv1 極致斷詞" button to the "新" and "教練機" buttons.


更多更多 **Articut** 進階操作 (lv1, lv2, lv3)



The screenshot shows the Articut app interface. At the top left is the Articut logo. To its right is a settings icon and a dropdown menu labeled "lv2 詞組斷詞" (lv2 phrase chunking), which is highlighted with a red box. Further right is a "景點資料庫" (Scenic spot database) section with a toggle switch and a "關閉" (Close) button. Below the settings is a text box showing "免費字數：1514" (Free characters: 1514) and "新教練機預計明年進入量產" (New coach machine expected to enter mass production next year). Below this is a section titled "機讀結果：新教練機/預計...(more)" (Machine reading result: New coach machine/expected...(more)). Below the title is a row of five buttons: "新教練機" (nouny), "預計" (verb), "明年" (time), "進入" (verb), and "量產" (verb). The "新教練機" button is highlighted with a red box, and a red arrow points from the "lv2 詞組斷詞" dropdown to it.

更多更多 **Articut** 進階操作 (lv1, lv2, lv3)

lv3 處理各種「整篇文本中，和語意相關的議題」(e.g., 人、事、時、地、物)



⚙ 設定： lv3 語意斷詞 ▾

🔤 A 拼音： 注音 ▾


📌 景點資料庫： × 關閉

🌐 WikiData： × 關閉

📖 自定義詞典 ▾

免費字數：1562

新教練機預計明年進入量產，2025年撥交部隊



時間 (Time)

新教練機預計 明年 進入量產， 2025年 撥交部隊

明年 → night → 2021 年 1 月 1 日 星期五 0 點 0 分 0 秒 ~ 2021 年 12 月 31 日 星期五 23 點 59 分 59 秒

2025年 → night → 2025 年 1 月 1 日 星期三 0 點 0 分 0 秒 ~ 2025 年 12 月 31 日 星期三 23 點 59 分 59 秒

<https://api.droidtown.co/graphQL/>

更多更多 Articut 進階操作 (Articut-GraphQL)

SpaCy: <https://spacy.io/>

- https://github.com/howl-anderson/Chinese_models_for_SpaCy
- <https://www.mdeditor.tw/pl/pzWI/zh-tw>

```
Edit the code & try spaCy spaCy v2.3.0 - Python 3 - via Binder

# pip install spacy
# python -m spacy download en_core_web_sm


import spacy

# Load English tokenizer, tagger, parser, NER and word vectors
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. "I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn't "
        "worth talking to," said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])
```

Articut-GraphQL: 所有 GraphQL 指令都「相容」SpaCy, 但 Articut-GraphQL 的「中文處理」能力更強!


 Articut-GraphQL

免費字數：1,707

GraphiQL Prettify History Explorer 教學模式 < Docs

```
1 # shift-option/alt-click on a query below to jump to
2 # option/alt-click on a field in the explorer to se
3
4 query ($text: String!, $model: String!) {
5   nlp(text: $text, model: $model) {
6     meta {
7       lang
8       description
9     }
10    doc {
11      text
12      tokens {
13        text
14        pos_
15        tag_
16        isStop
17        isEntity
18        isVerb
19        isTime
```

這裡可以瀏覽解析後的原始資訊!



QUERY VARIABLES

```
{
  {
```

```
{
  "data": {
    "nlp": {
      "meta": {
        "lang": "TW",
        "description": "Articut GraphQL Query Result."
      },
      "doc": {
        "text": "周天成的逆轉秀不僅讓外媒大呼不可思議，與香港
        伍家朗在冠軍戰的最後一球也被BWF選為單月最佳好球。而小天連續在印
        尼及泰國奪冠，積分擠下中國好手石宇奇，排名升至世界男單第二，創下
        生涯最佳，許多外媒也看好他挾著這股氣勢，能在接下來的世錦賽中打出
        佳績。",
        "tokens": [
          {
            "text": "周天成",
            "pos_": "PERSON",
            "tag_": "ENTITY_person",
            "isStop": false,
            "isEntity": true,
            "isVerb": false,
            "isTime": false,
            "isClause": false,
```

可以不要再給我「不分詞性」的文字雲了嗎？

普通文字雲：

[我們] 出現很多次，可見講者很重視「我們」
[台灣] 出現很多次，可見講者很重視「台灣」
[經濟] 出現很多次，可見講者想要拼經濟。

這些都是後見之明的
「過度腦補」而已。



Articut-GraphQL: 可依詞性 (POS) 分別繪製文字雲
(哦！這個功能嘛...SpaCy 沒有。)



動詞文字雲：

動詞描述的是行動，是事件。因此動詞文字雲可清楚看出文本中的實際 [行動] 是哪些。



形容詞文字雲：

形容詞描述的是主觀判斷的結果。因此形容詞文字雲可清楚看出文本中的 [判斷結果] 是什麼。



Articut 處理不好的地方，原因及解決辦法

- ❖ 文本是由「語言」+「知識」構成的。Articut 是基於「語言」設計的，它沒有「知識」。因此所有的**簡稱或長組織機構名稱**，它都處理不算最好。
- ❖ 中文文本只有「符號」，沒有「聲音」，因此破音字出現時會處理得不好，外文音譯的外來語也會較吃力。(e.g., 派對、臥踢、迪斯科)
- ❖ 解決的辦法：幫它加上「知識」
 - 開啟 OpenData
 - 開啟 WikiData
 - 下載領域字典
 - 自己手動加上字典



Articut 處理特別好的地方及原因

- ❖ Articut 是基於「語言」設計的，它沒有內建「知識」(也就是沒有字典)。因而對 OOV (新詞) 的判斷接受度特別強，也幾乎不受 OOV 影響它的良率。ref. <https://github.com/taedlar/wordcept>
- ❖ 內建「句法」讓 Articut 處理「轉品」(詞性轉變) 的能力更好 (我們甚至封印了一部份以便和其它的方案做相容)。
- ❖ 它只有內建「句法」，沒有模型，沒有資料，故運算速度極快，程式超級小。
- ❖ 無需資料，即無訓練。既可保障隱私、也節約能源、節省模型訓練成本。

The image shows two screenshots of the Articut web interface. The top screenshot shows the input '香蕉' (Banana) and the output '機讀結果：香蕉...(more)'. Below the output, the word '香蕉' is highlighted with a red box and labeled 'OOV'. The bottom screenshot shows the input '我吃了一串香蕉' (I ate a string of bananas) and the output '機讀結果：我/吃了...(more)'. Below the output, the words are highlighted with red boxes and labeled with their parts of speech: '我' (pronoun), '吃了' (verb), '一串' (classifier), and '香蕉' (noun). A red arrow points from the 'OOV' label in the top screenshot to the '香蕉' label in the bottom screenshot.

Articut 處理特別好的地方及原因

❖ Articut 是基於「語言」設計的，它沒有內建「知識」(也就是沒有字典)。因而對 OOV (新詞) 的判斷接受度特別強，也幾乎不受 OOV 影響它的良率。ref. <https://github.com/taedlar/wordcept>

❖ 內建「句法」讓 Articut 處理「轉品」(詞性轉變) 的能力更好 (我們甚至封印了一部份以便和其它的方案做相容)。

❖ 它只有內建「句法」，沒有模型，沒有資料，故運算速度極快，程式超級小。

❖ 無需資料，即無訓練。既可保障隱私、也節約能源、節省模型訓練成本。

Articut 處理特別好的地方及原因

設定：lv2 詞組斷詞
免費字數：1993

香蕉

機讀結果：香蕉...(more)

香蕉 OOV

設定：lv2 詞組斷詞
免費字數：1976

景點資料庫：

我吃了一串香蕉

機讀結果：我/吃了...(more)

我 pronoun 吃了 verbp 一串 classifier 香蕉 nouny

Quiz:

課堂中已經演示了文本處理中，利用 Articut 進行「斷詞工作」、POS 詞性標記以及 NER 命名實體辨識和其它多種不同的 NLP 任務。試思考以下問題：

1. 斷詞系統的運作原理有哪幾種？Articut 是屬於哪一種？
2. 「訓練文本」和「應用文本」的差異，是否會造成 Articut 的表現不佳？
3. 只有斷詞處理，能取出什麼樣的資訊？這個資訊是否足以呈現文本特性？
4. 加上 POS 處理，能取出什麼樣的資訊？用什麼方法？
5. 再加上 NER 處理，能取出什麼樣的資訊？這個資訊是否有什麼限制？

Assignment: 小組作業，每組繳一份至你們的「組名目錄」即可

1. 從課程 github repo 中把課程中提供的 week09 的目錄 git pull 下來。
2. 把 week09.py 改名為 **week09_分組隊名.py**
3. 在 **week09_分組隊名.py** 中，設計你的程式，利用 Articut 完成以下指定規格：
 - a. 把 "tourblog.json" 中的「行政地名」和「景點名稱」取出，另存入兩個 LIST，再將這兩個 LIST 存成 tourblog_geoinfo.json 檔。內容為 {"location": [...], "place": [...]}
 - b. 把 "刑事判決_106,交簡,359_2017-02-21.json" 中 "mainText" 欄位裡的刑罰取出，另存為 justice.json 檔，內容為 {"liability": "<你取出的刑罰>"}
 - c. 把 "news.json" 中 "content" 中的：
 - i. 人名列出，並計算每個人名出現的次數
 - ii. 地點列出
 - iii. 涉案金額列出
 - iv. 將以上資訊另存為 news_info.json 檔，內容為 {"people": [(人名, 次數), (人名, 次數), ...], "location": [...], "money": [...]}