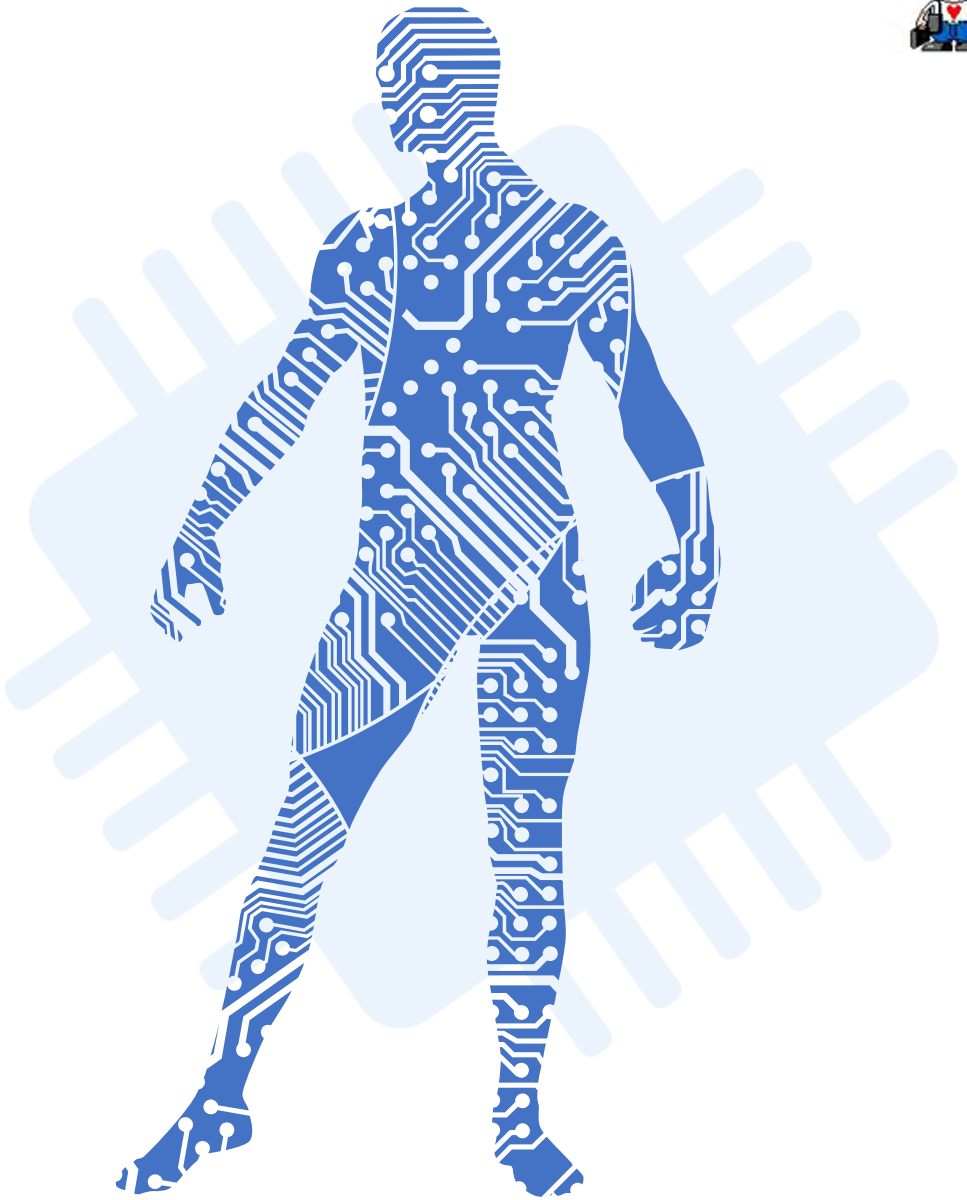
自然語言處理

第 7 章 前處理

講師：紀俊男

本章大綱

- 環境設定
- 語系偵測（Language Detection）
- 斷句（Sentence Segmentation）
- 顏文字處理（Emoji Processing）
- 拼寫檢查（Spelling Check）
- 中文繁簡轉換（Traditional-Simplified Chinese）
- 斷詞與文本清理（Tokenization & Text Cleaning）
- 英文文本標準化（Text Normalization）
- 詞性標註（Part-of-Speech Tagging） PoS
- 名稱實體識別（Named Entity Recognition） NER



AI 環境設定

範例完整原始碼：
https://lurl.cc/3Jk69b

原始程式碼

Natural Language Tool Kit，能針對英文執行
斷句、斷詞、詞性標註...等處理的優秀套件

老牌英文函式庫

能針對各種語言（包括中文）執行
斷句、詞性標註...等處理的優秀套件

spaCy 需要根據處理的語系，
額外下載語言模型安裝包。

small

medium

```
1  # 安裝 NLTK 套件
2  !pip install nltk
3
4  # 安裝 spaCy 套件
5  !pip install spacy
6
7  # 下載中英文語言模型
8  # spaCy 語言模型首頁： https://github.com/explosion/spacy-models
9
10 # 下載英文語言模型（sm = small）
11 !python -m spacy download en_core_web_sm
12
13 # 下載中文語言模型（md = middle）
14 !python -m spacy download zh_core_web_md
15
16 chinese_text = """
17 在這個資訊爆炸的時代，自然語言處理(Natural Language Processing, NLP)技術正在迅速發展，成為人工智慧領域的一個重要分支。NLP的應用範圍廣泛，包括語音識別、機器翻譯、情感分析、文本摘要等。隨著深度學習技術的進步，NLP的研究和應用正在不斷突破傳統的限制，為人們的生活和工作帶來了許多便利。
18
19 自然語言處理涉及到多個步驟，其中包括語言檢測、文本清理、文本標準化、斷句、斷詞、詞性標註等。這些前處理步驟對於後續的分析和模型訓練至關重要。例如，文本清理可以去除無用的標點符號和特殊字符，斷詞則將句子分割成單個的詞彙，便於進一步的處理。
20
21 在NLP的應用中，一個常見的任務是情感分析，即判斷文本中所表達的情感是正面還是負面。這在社交媒體分析、市場研究等領域有著廣泛的應用。另一個重要的應用是機器翻譯，隨著全球化的發展，準確高效的翻譯工具變得越來越重要。
22
23 然而，NLP也面臨著許多挑戰，其中之一是語言的多樣性和複雜性。每種語言都有其獨特的語法結構和詞彙規則，這使得開發通用的NLP模型變得困難。此外，不同語境下同一詞彙的含義可能會有所不同，這就需要模型能夠理解上下文資訊。
24
25 總的來說，自然語言處理是一個充滿挑戰和機遇的領域。隨著技術的不斷進步，我們有理由相信，NLP將在未來發揮更大的作用，為人類社會帶來更多的便利和進步。
26 """
```

隨堂練習：環境設定

- 請將前一頁的原始碼拷貝貼上，並且執行看看：

```
# 安裝 NLTK 套件
!pip install nltk

# 安裝 spaCy 套件
!pip install spacy

# 下載中英文語言模型
# spaCy 語言模型首頁： https://github.com/explosion/spacy-models

# 下載英文語言模型（sm = small）
!python -m spacy download en_core_web_sm

# 下載中文語言模型（md = middle）
!python -m spacy download zh_core_web_md

chinese_text = """
在這個資訊爆炸的時代，自然語言處理(Natural Language Processing, NLP)技術正在迅速發展，成為人工智慧領域的一個重要分支。NLP的應用範圍廣泛，包括語音識別、機器翻譯、情感分析、文本摘要等。隨著深度學習技術的進步，NLP的研究和應用正在不斷突破傳統的限制，為人們的生活和工作帶來了許多便利。

自然語言處理涉及到多個步驟，其中包括語言檢測、文本清理、文本標準化、斷句、斷詞、詞性標註等。這些前處理步驟對於後續的分析和模型訓練至關重要。例如，文本清理可以去除無用的標點符號和特殊字符，斷詞則將句子分割成單個的詞彙，便於進一步的處理。

在NLP的應用中，一個常見的任務是情感分析，即判斷文本中所表達的情感是正面還是負面。這在社交媒體分析、市場研究等領域有著廣泛的應用。另一個重要的應用是機器翻譯，隨著全球化的發展，準確高效的翻譯工具變得越來越重要。

然而，NLP也面臨著許多挑戰，其中之一是語言的多樣性和複雜性。每種語言都有其獨特的語法結構和詞彙規則，這使得開發通用的NLP模型變得困難。此外，不同語境下同一詞彙的含義可能會有所不同，這就需要模型能夠理解上下文資訊。

總的來說，自然語言處理是一個充滿挑戰和機遇的領域。隨著技術的不斷進步，我們有理由相信，NLP將在未來發揮更大的作用，為人類社會帶來更多的便利和進步。
"""
```

語系偵測

Language Detection

何謂「語系偵測」？

- 偵測「文本（Text）」是哪一種語言？

“你好！我叫李傑克。”

中文（zh）

“Hello! My name is Jack Lee.”

英文（en）

為何要做「語系偵測」？

- 不同**語系**、不同**前處理步驟 / 函式庫**

英文（en）

詞語標準化（**Text Normalization**）
- 取動詞原型（Lemmatization）
- 轉成小寫（Lower cases）

斷詞（**Tokenization**）
- import nltk

中文（zh）

詞語標準化（**Text Normalization**）
- 繁簡轉換

斷詞（**Tokenization**）
- import jieba

語系偵測實作

```
1  # 套件安裝（Package Installation）
2  !pip install langid
3
4  # 文本（Text）設定
5  chinese_text = "你好！我的名字叫做李傑克！"
6  english_text = "Hello! My name is Jack Lee."
7
8  # 語系偵測（分數未標準化）
9  import langid
10
11 result = langid.classify(chinese_text)
12 print(result)
```

language id

('zh', -153.1489453315735)

作者自訂之 Confidence Score（$\log_{10}\frac{1}{\text{分數}}$）

常見語系代碼（ISO-639 標準）

Language Code	Language Name	Language Code	Language Name
zh	Chinese	ko	Korean
en	English	ru	Russian
es	Spanish	pt	Portuguese
fr	French	it	Italian
de	German	ar	Arabic
ja	Japanese	tr	Turkish
nl	Dutch	pl	Polish
sv	Swedish	da	Danish
fi	Finnish	no	Norwegian
el	Greek	he	Hebrew
th	Thai	id	Indonesian
vi	Vietnamese		

隨堂練習：語系偵測

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```
1  # 套件安裝（Package Installation）
2  !pip install langid
3
4  # 文本（Text）設定
5  chinese_text = "你好！我的名字叫做李傑克！"
6  english_text = "Hello! My name is Jack Lee."
7
8  # 語系偵測（分數未標準化）
9  import langid
10
11 result = langid.classify(chinese_text)
12 print(result)
```

```
('zh', -153.1489453315735)
```

將分數標準化

載入必要的套件

```
1  # 語系偵測（分數已標準化）
2  from langid.langid import LanguageIdentifier, model
3
4  detector = LanguageIdentifier.from_modelstring(model, norm_probs=True)
5
6  result = detector.classify(chinese_text)
7  print(result)
```

自建語系偵測物件　　載入預訓練好的模型　　標準化分數為 0~1

偵測語系

```
('zh', 0.9999999999667954)
```

指定候選語系

```
1  # 語系偵測（指定候選語系 + 分數標準化）
2  import langid
3  from langid.langid import LanguageIdentifier, model
4
5  langid.set_languages(['en','zh'])
6  detector = LanguageIdentifier.from_modelstring(model, norm_probs=True)
7
8  result = detector.classify(chinese_text)
9  print(result)
10
11 result = detector.classify(english_text)
12 print(result)
```

1 載入必要的套件

2 指定候選語系

3 自建語系偵測器 + 標準化分數

4 ('zh', 0.9999999999667954)

5 ('en', 0.9977290649556259)

隨堂練習：語系偵測

- 請先撰寫好前兩頁的原始碼 ，並且執行看看：

```python
# 語系偵測（分數已標準化）
from langid.langid import LanguageIdentifier, model

detector = LanguageIdentifier.from_modelstring(model, norm_probs=True)

result = detector.classify(chinese_text)
print(result)
```

```python
# 語系偵測（指定候選語系 + 分數標準化）
import langid
from langid.langid import LanguageIdentifier, model

langid.set_languages(['en','zh'])
detector = LanguageIdentifier.from_modelstring(model, norm_probs=True)

result = detector.classify(chinese_text)
print(result)

result = detector.classify(english_text)
print(result)
```

斷句

Sentence Segmentation

英文斷句

```python
# 載入 NLTK
import nltk

# 下載斷句用模型（punkt = Punctuation）
nltk.download('punkt')

# 利用 sent_tokenize 斷句
from nltk.tokenize import sent_tokenize

text = "Hello, world! Welcome to the world of NLP. This is an example of sentence tokenization."
sentences = sent_tokenize(text)
print(sentences)
```

python list

['Hello, world!', 'Welcome to the world of NLP.', 'This is an example of sentence tokenization.']

隨堂練習：英文斷句

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 載入 NLTK
import nltk

# 下載斷句用模型（punkt = Punctuation）
nltk.download('punkt')

# 利用 sent_tokenize 斷句
from nltk.tokenize import sent_tokenize

text = "Hello, world! Welcome to the world of NLP. This is an example of sentence tokenization."
sentences = sent_tokenize(text)

print(sentences)
```

```
['Hello, world!', 'Welcome to the world of NLP.', 'This is an example of sentence tokenization.']
```

中文斷句

```python
import spacy

# 載入中文模型
nlp = spacy.load("zh_core_web_md")

# 使用 spaCy 進行斷句
doc = nlp(chinese_text)

# 去除「空白行」與「行末換行符號 \n」
sentences = [sent.text.strip() for sent in doc.sents if sent.text.strip()]

# 輸出結果
print(sentences)
```

```
['在這個資訊爆炸的時代，自然語言處理(Natural Language Processing, NLP)技術正在迅速發展，成為人工智慧領域的一個重要分支。',
'NLP的應用範圍廣泛，包括語音識別、機器翻譯、情感分析、文本摘要等。',
'隨著深度學習技術的進步，NLP的研究和應用正在不斷突破傳統的限制，為人們的生活和工作帶來了許多便利。',
'自然語言處理涉及到多個步驟，其中包括語言檢測、文本清理、文本標準化、斷句、斷詞、詞性標註等。',
'這些前處理步驟對於後續的分析和模型訓練至關重要。',
'例如，文本清理可以去除無用的標點符號和特殊字符，斷詞則將句子分割成單個的詞彙，便於進一步的處理。',
'在NLP的應用中，一個常見的任務是情感分析，即判斷文本中所表達的情感是正面還是負面。',
'這在社交媒體分析、市場研究等領域有著廣泛的應用。',
'另一個重要的應用是機器翻譯，隨著全球化的發展，準確高效的翻譯工具變得越來越重要。',
'然而，NLP也面臨著許多挑戰，其中之一是語言的多樣性和複雜性。',
'每種語言都有其獨特的語法結構和詞彙規則，這使得開發通用的NLP模型變得困難。',
'此外，不同語境下同一詞彙的含義可能會有所不同，這就需要模型能夠理解上下文資訊。',
'總的來說，自然語言處理是一個充滿挑戰和機遇的領域。',
'隨著技術的不斷進步，我們有理由相信，NLP將在未來發揮更大的作用，為人類社會帶來更多的便利和進步。']
```

隨堂練習：中文斷句

- 請先撰寫好前兩頁的原始碼 ，並且執行看看：

```python
import spacy

# 載入中文模型
nlp = spacy.load("zh_core_web_md")

# 使用 spaCy 進行斷句
doc = nlp(chinese_text)

# 去除「空白行」與「行末換行符號 \n」
sentences = [sent.text.strip() for sent in doc.sents if sent.text.strip()]

# 輸出結果
print(sentences)
```

一次一句

removes leading & trailing whitespaces

list comprehension

顏文字處理

Emoji Processing

顏文字（Emoji）簡介

- 顏文字是什麼？

😂 👍 ❤️ 🤣 😢

- 顏文字的危害與處理

車禍骨折來住院一週🧍林希鼎Doctors 技術很棒 幽默^_^然後骨科醫師的護理人員團隊..細心👍B1有美容院坐洗髮還滿特別的💞

↓

未知字元，造成「無法判斷」或「誤判」

車禍骨折來住院一週 :man:林希鼎Doctors 技術很棒 幽默:smile:然後骨科醫師的護理人員團隊..細心:thumbs_up: B1有美容院坐洗髮還滿特別的:heart:

↓

轉成純文字，勉強可以判讀或處理

原始程式碼

1
2
3

```
1  # 安裝 emoji 套件
2  !pip install emoji
3
4  # 引入 emoji 套件
5  import emoji
6
7  # 用 demojize() 將顏文字轉成純文字
8  print(emoji.demojize('Python is ❤️'))
```

```
Python is :red_heart:
```

隨堂練習：顏文字處理

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```
# 安裝 emoji 套件
!pip install emoji

# 引入 emoji 套件
import emoji

# 用 demojize() 將顏文字轉成純文字
print(emoji.demojize('Python is ❤️'))
```

Python is :red_heart:

AI

拼寫檢查

Spelling Check

英文拼寫檢查

Language	Code	Language	Code	Language	Code
English	en	Dutch	nl	French	fr
German	de	Spanish	es	Portuguese	pt
Arabic	ar	Asturian	ast	Belarusian	be
Breton	br	Catalan	ca	Chinese	zh
Danish	da	Esperanto	eo	Galician	gl
Greek	el	Irish	ga	Italian	it
Japanese	ja	Khmer	km	Norwegian	no
Persian	fa	Polish	pl	Romanian	ro
Russian	ru	Slovak	sk	Slovenian	sl
Swedish	sv	Tagalog	tl	Tamil	ta
Ukrainian	uk				

1
2
3
4
5
6

```python
# 安裝 language-tool-python 套件
!pip install language-tool-python

# 載入 language_tool_python 套件
import language_tool_python

# 指定拼寫檢查之語系
tool = language_tool_python.LanguageTool('en-US')

# 開始拼寫檢查
text = 'Speling erors in sentense are anoying.'
matches = tool.check(text)

# 印出拼寫檢查結果之資料結構
print(matches)

# 以拼寫檢查結果修正原文並印出
corrected_text = language_tool_python.utils.correct(text, matches)
print(corrected_text)
```

```
[Match({'ruleId': 'MORFOLOGIK_RULE_EN_US',
      'message': 'Possible spelling mistake found.',
      'replacements': ['Spelling', 'Spewing', 'Spieling'],
      'offsetInContext': 0,
      'context': 'Speling erors in sentense are anoying.',
      'offset': 0, 'errorLength': 7,
      'category': 'TYPOS', 'ruleIssueType': 'misspelling',
      'sentence': 'Speling erors in sentense are anoying.'}),
 Match({'ruleId': 'MORFOLOGIK_RULE_EN_US',
 ...
]
```

1 typo

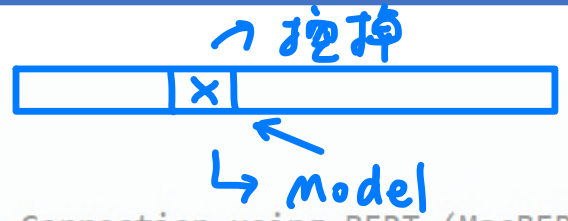
Spelling errors in sentence are annoying.

隨堂練習：英文拼寫檢查

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 安裝 language-tool-python 套件
!pip install language-tool-python

# 載入 language_tool_python 套件
import language_tool_python

# 指定拼寫檢查之語系
tool = language_tool_python.LanguageTool('en-US')

# 開始拼寫檢查
text = 'Speling erors in sentense are anoying.'
matches = tool.check(text)

# 印出拼寫檢查結果之資料結構
print(matches)

# 以拼寫檢查結果修正原文並印出
corrected_text = language_tool_python.utils.correct(text, matches)
print(corrected_text)
```

中文拼寫檢查



Bidirectional Encoder Representations from Transformers

上下文

```
1  # 安裝 PyCorrector 套件
2  !pip install pycorrector
3
4  # 載入 MLM (Masked Language Model) as Correction using BERT (MacBERT) 模型
5  from pycorrector import MacBertCorrector
6  corrector = MacBertCorrector()
7
8  # 測試用中文文本
9  error_sentences = [
10     '今天新情很好',
11     '我遇到一位老友跟我療天。',
12     '他們只能有兩個選擇：接受降新或自動離職。'
13 ]
14
15 # 進行拼寫檢查
16 batch_results = corrector.correct_batch(error_sentences)
17
18 # 印出結果
19 for result in batch_results:
20     print(result)
21     print(result['source'])
22     print(result['target'])
23     print("----------------------------------")
```

一次修正多句

(100句左右為佳)

```
{'source': '今天新情很好', 'target': '今天心情很好', 'errors': [('新', '心', 2)]}
今天新情很好
今天心情很好
----------------------------------
{'source': '我遇到一位老友跟我療天。', 'target': '我遇到一位老友跟我聊天。', 'errors': [('療', '聊', 9)]}
我遇到一位老友跟我療天。
我遇到一位老友跟我聊天。
----------------------------------
{'source': '他們只能有兩個選擇：接受降新或自動離職。',
 'target': '他們只能有兩個選擇：接受降薪或自動離職。', 'errors': [('新', '薪', 13)]}
他們只能有兩個選擇：接受降新或自動離職。
他們只能有兩個選擇：接受降薪或自動離職。
----------------------------------
```

一個錯 = 一筆記錄

隨堂練習：中文拼寫檢查

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 安裝 PyCorrector 套件
!pip install pycorrector

# 載入 MLM (Masked Language Model) as Correction using BERT (MacBERT) 模型
from pycorrector import MacBertCorrector
corrector = MacBertCorrector()

# 測試用中文文本
error_sentences = [
    '今天新情很好',
    '我遇到一位老友跟我療天。',
    '他們只能有兩個選擇：接受降新或自動離職。'
]

# 進行拼寫檢查
batch_results = corrector.correct_batch(error_sentences)

# 印出結果
for result in batch_results:
    print(result)
    print(result['source'])
    print(result['target'])
    print("----------------------------------")
```

```
{'source': '今天新情很好', 'target': '今天心情很好', 'errors': [('新', '心', 2)]}
今天新情很好
今天心情很好
----------------------------------
{'source': '我遇到一位老友跟我療天。', 'target': '我遇到一位老友跟我聊天。', 'errors': [('療', '聊', 9)]}
我遇到一位老友跟我療天。
我遇到一位老友跟我聊天。
----------------------------------
{'source': '他們只能有兩個選擇：接受降新或自動離職。' ,
 'target': '他們只能有兩個選擇：接受降薪或自動離職。', 'errors': [('新', '薪', 13)]}
他們只能有兩個選擇：接受降新或自動離職。
他們只能有兩個選擇：接受降薪或自動離職。
----------------------------------
```

中文繁簡轉換

Traditional-Simplified Chinese

中文繁簡轉換簡介

- **何謂「中文繁簡轉換」**
  - “文本标准化” → “文本標準化”
- **為何需要「中文繁簡轉換」？**
  - 某些 Python 函式庫僅接受簡體中文。
- **中文繁簡轉換的難度**
  - 一對多對應：“后” → “后” / “後” ？
  - 用語不同：“信息” → “資訊”
- **解決方案**
  - OpenCC 套件

原始程式碼

```
1  # 安裝並引入 OpenCC 套件
2  !pip install opencc
3  from opencc import OpenCC
4
5  # 定義一個專門做「繁簡轉換」的函數
6  def convert_chinese(text, target='simplified', dialect_convert=True):
7      # 根據傳入的 'target' 與 'dialect_convert' 參數，取得對應的 JSON 檔
8      configurations = {
9          ('simplified', True): 'tw2sp.json',
10         ('simplified', False): 'tw2s.json',
11         ('traditional', True): 's2twp.json',
12         ('traditional', False): 's2tw.json'
13     }
14     configuration = configurations.get((target, dialect_convert))
15
16     # 初始化對應的轉換器
17     converter = OpenCC(configuration)
18
19     # 轉換並將結果傳回去
20     return converter.convert(text)
21
22 # 呼叫轉換函數，並印出轉換結果
23 print(convert_chinese(chinese_text))
```

1 2 3 4 5 6

configuration = 'tw2sp.json'

在這個資訊爆炸的時代，自然語言處理(Natural Language Processing, NLP)技術正在迅速發展，成為人工智慧領域的一個重要分支。...（後略）...

在这个信息爆炸的时代，自然语言处理(Natural Language Processing, NLP)技术正在迅速发展，成为人工智能领域的一个重要分支。....（後略）...

隨堂練習：中文繁簡轉換

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 安裝並引入 OpenCC 套件
!pip install opencc
from opencc import OpenCC

# 定義一個專門做「繁簡轉換」的函數
def convert_chinese(text, target='simplified', dialect_convert=True):
    # 根據傳入的 'target' 與 'dialect_convert' 參數，取得對應的 JSON 檔
    configurations = {
        ('simplified', True): 'tw2sp.json',
        ('simplified', False): 'tw2s.json',
        ('traditional', True): 's2twp.json',
        ('traditional', False): 's2tw.json'
    }
    configuration = configurations.get((target, dialect_convert))

    # 初始化對應的轉換器
    converter = OpenCC(configuration)

    # 轉換並將結果傳回去
    return converter.convert(text)

# 呼叫轉換函數，並印出轉換結果
print(convert_chinese(chinese_text))
```

在这个信息爆炸的时代，自然语言处理(Natural Language Processing, NLP)技术正在迅速发展，成为人工智能领域的一个重要分支。....（後略）...

斷詞與文本清理

Tokenization & Text Cleaning

英文斷詞與文本清理

```python
# 引入必要函式庫
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string

# 下載 NLTK 的 Tokenizer 和 stopwords 資源
nltk.download('punkt')
nltk.download('stopwords')

# 獲取英語的停用詞列表
stop_words = set(stopwords.words('english'))

# 定義一個函數來進行斷詞、移除標點符號和停用詞
def tokenize_without_punctuation_and_stopwords(text):
    # 使用 NLTK 的 word_tokenize 進行斷詞
    tokens = word_tokenize(text)
    # 移除標點符號和停用詞
    filtered_tokens = [token for token in tokens
      if token not in string.punctuation and token.lower() not in stop_words]
    return filtered_tokens

# 測試文本
text = "Hello, how are you doing today?"

# 呼叫函數進行斷詞並移除標點和停用詞
tokens = tokenize_without_punctuation_and_stopwords(text)
print(tokens)
```

tuple
list
dictionary
set

['Hello', 'today']

隨堂練習：英文斷詞與文本清理

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 引入必要函式庫
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string

# 下載 NLTK 的 Tokenizer 和 stopwords 資源
nltk.download('punkt')
nltk.download('stopwords')

# 獲取英語的停用詞列表
stop_words = set(stopwords.words('english'))

# 定義一個函數來進行斷詞、移除標點符號和停用詞
def tokenize_without_punctuation_and_stopwords(text):
    # 使用 NLTK 的 word_tokenize 進行斷詞
    tokens = word_tokenize(text)
    # 移除標點符號和停用詞
    filtered_tokens = [token for token in tokens
      if token not in string.punctuation and token.lower() not in stop_words]
    return filtered_tokens

# 測試文本
text = "Hello, how are you doing today?"

# 呼叫函數進行斷詞並移除標點和停用詞
tokens = tokenize_without_punctuation_and_stopwords(text)
print(tokens)
```

```
['Hello', 'today']
```

中文斷詞與文本清理

1
2
3
4
5

```
1  # 安裝 Jieba (結巴)
2  !pip install jieba
3
4  # 取得繁體中文自定義辭典 (可自行修改添加, 選用)
5  import os
6  Dictionary_File = 'dict.txt.big'
7
8  if not os.path.isfile(Dictionary_File):
9      os.system('wget https://raw.githubusercontent.com/cnchi/datasets/master/' + Dictionary_File)
10
11 # 取得繁體中文的「停止詞」辭典 (如：的、了、...)
12 StopWords_File = "stopWords_big5.txt"
13
14 if not os.path.isfile(StopWords_File):
15     os.system('wget https://raw.githubusercontent.com/cnchi/datasets/master/' + StopWords_File)
16
17 # 載入 Jieba
18 import jieba
19
20 # 載入繁體中文自定義辭典 (選用)
21 # jieba.set_dictionary(Dictionary_File)
```

名字可任取

紀|後男
↑不要斷詞

中文斷詞與文本清理

```python
# 設定標點符號與停止詞清單
punctuation = set("$!&#%\()+-*/_,.   ?:;'\"<=>^`|~[]{}’0123456789?_“”、。《》！，  ：；？「」（） ")
stopWords = set()
with open(StopWords_File, "rt", encoding="utf-8") as f:
    for line in f:
        line = line.strip() # Remove trailing \n
        stopWords.add(line)

# 將斷完句的文字，一次一句送入斷詞＆文本清理
result = []
for sentence in sentences:
    # 針對本句子斷詞
    seg_sentence = list(jieba.cut(sentence))
    # 移除標點符號
    seg_sentence_no_punct = [word for word in seg_sentence if word not in punctuation]
    # 移除停止詞
    seg_sentence_no_stopwords = [word for word in seg_sentence_no_punct if word not in stopWords]

    # 將結果加入 result 串列
    result.append(seg_sentence_no_stopwords)

# 印出結果
print(result)
```

1. 設定標點符號與停止詞清單
2. 將斷完句的文字，一次一句送入斷詞＆文本清理
3. 針對本句子斷詞
4. 移除標點符號
5. 移除停止詞
6. 將結果加入 result 串列
7. 印出結果

```
[['資訊', '爆炸', '時代', '自然', '語言', '處理', 'Natural',
  'Language', 'Processing', 'NLP', '技術', '正在', '迅速',
  ...（後略）...]
]
```

隨堂練習：中文斷詞與文本清理

- 請先撰寫好前兩頁的原始碼 ，並且執行看看：

```
# 安裝 Jieba (結巴)
!pip install jieba

# 取得繁體中文自定義辭典
import os
Dictionary_File = 'dic

if not os.path.isfile(
    os.system('wget ht

# 取得繁體中文的「停止詞」
StopWords_File = "stop

if not os.path.isfile(
    os.system('wget ht

# 載入 Jieba
import jieba

# 載入繁體中文自定義辭典
# jieba.set_dictionary
```

```
# 設定標點符號與停止詞清單
punctuation = set("$!&#%\()+-*/_,.   ?:;'\"<=>^`|~[]{}’0123456789?_“”、。《》！，：；？「」（） ")
stopWords = set()
with open(StopWords_File, "rt", encoding="utf-8") as f:
  for line in f:
    line = line.strip() # Remove trailing \n
    stopWords.add(line)

# 將斷完句的文字，一次一句送入斷詞＆文本清理
result = []
for sentence in sentences:
  # 針對本句子斷詞
  seg_sentence = list(jieba.cut(sentence))
  # 移除標點符號
  seg_sentence_no_punct = [word for word in seg_sentence if word not in punctuation]
  # 移除停止詞
  seg_sentence_no_stopwords = [word for word in seg_sentence_no_punct if word not in stopWords]

  # 將結果加入 result 串列
  result.append(seg_sentence_no_stopwords)

# 印出結果
print(result)
```

英文文本標準化

Text Normalization

英文文本需要哪些標準化處理？

小寫化
(Lower Cases)

The

↓

the

詞幹提取
(Stemming)

leaves

↓

leav

詞形還原
(Lemmatization)

leaves

↓

leaf

e.g. irregular verbs, plurals

原始程式碼

```python
# 載入必要套件
import nltk
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import wordnet

# 下載 NLTK 資源
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

# 初始化 Stemmer 和 Lemmatizer
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

# 定義一個函數來展示小寫、Stemming 和 Lemmatization
def process_text(text):
    # 轉成小寫
    text = text.lower()
    # 使用 NLTK 的 word_tokenize 進行斷詞
    words = nltk.word_tokenize(text)

    # Stemming
    stemmed_words = [stemmer.stem(word) for word in words]

    # Lemmatization
    # 為了更精確的 Lemmatization，我們需要提供正確的詞性 (Part-of-Speech)
    lemmatized_words = [lemmatizer.lemmatize(word, pos=get_wordnet_pos(word)) for word in words]

    return stemmed_words, lemmatized_words
```

word, ~~sentence~~

一次取一個詞 #取字根

PoS Tagging.

原始程式碼

Python dictionary get() prototype :

get(find, default_value)

```
31  # NLTK 的詞性與 WordNet 詞性的對應
32  def get_wordnet_pos(word):
33      """Map NLTK's part-of-speech tags to those used by WordNet Lemmatizer"""
34      tag = nltk.pos_tag([word])[0][1][0].upper()
35      tag_dict = {"J": wordnet.ADJ,
36                  "N": wordnet.NOUN,
37                  "V": wordnet.VERB,
38                  "R": wordnet.ADV}
39
40      return tag_dict.get(tag, wordnet.NOUN)
41
42  # 測試文本
43  text = "The leaves on the tree have fallen due to strong winds."
44
45  # 處理文本
46  stemmed, lemmatized = process_text(text)
47  print("Stemmed Words:", stemmed)
48  print("Lemmatized Words:", lemmatized)
```

NLTk WordNet

1 ← 根據傳入的 word，取得 “J”，“N”，“V”，“R” 之一的詞性標註

2 ← 定義 NLTK 與 WordNet 辭典的詞性標籤對應表

3 ← 將 NLTK 詞性標籤，轉為 WordNet 標籤。找不到對應值，就以 NOUN 替代。

4 ← 測試文本

5 ← 處理文本

隨堂練習：英文文本標準化

- 請先撰寫好前兩頁的原始碼 ，並且執行看看：

```python
# 載入必要套件
import nltk
from nltk.stem import
from nltk.corpus impo

# 下載 NLTK 資源
nltk.download('punkt'
nltk.download('wordne
nltk.download('averag

# 初始化 Stemmer 和 L
stemmer = PorterStemm
lemmatizer = WordNetL

# 定義一個函數來展示小
def process_text(text
    # 轉成小寫
    text = text.lower
    # 使用 NLTK 的 wo
    words = nltk.word

    # Stemming
    stemmed_words = [

    # Lemmatization
    # 為了更精確的 Lem
    lemmatized_words

    return stemmed_words, lemmatized_words
```

```python
# NLTK 的詞性與 WordNet 詞性的對應
def get_wordnet_pos(word):
    """Map NLTK's part-of-speech tags to those used by WordNet Lemmatizer"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}

    return tag_dict.get(tag, wordnet.NOUN)

# 測試文本
text = "The leaves on the tree have fallen due to strong winds."

# 處理文本
stemmed, lemmatized = process_text(text)
print("Stemmed Words:", stemmed)
print("Lemmatized Words:", lemmatized)
```

#字根
#詞型還原

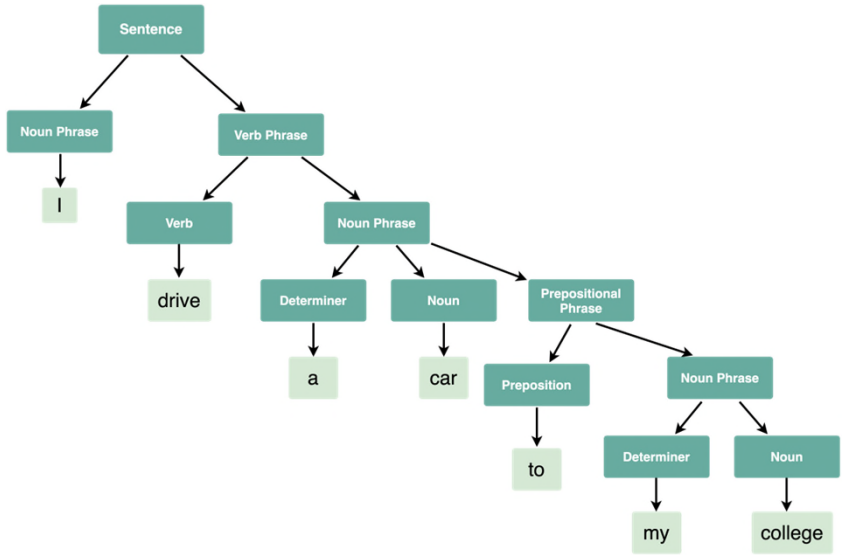
詞性標註

Part-Of-Speech Tagging

詞性標註簡介

- **何謂「詞性標註」（POS Tagging）？**
  - 將詞彙標上「**名詞、動詞、冠詞...**」等標記。

- **為何需要「詞性標註」？**
  - 方便未來的「**句型分析**」（Sentence Analysis），理解「**語意**」。



英文詞性標註

```python
# 載入必要的函式庫 NLTK
import nltk
from nltk.tokenize import word_tokenize

# 下載 NLTK 的 Tokenizer 和 POS Tagger 資源
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# 定義一個函數來執行詞性標註
def pos_tagging(text):
    # 斷詞
    words = word_tokenize(text)
    # 執行詞性標註
    pos_tags = nltk.pos_tag(words)
    return pos_tags

# 測試文本
text = "The quick brown fox jumps over the lazy dog."

# 獲得詞性標註結果
tagged_text = pos_tagging(text)
print("POS Tagging Result:", tagged_text)
```

Tag	Description	Example	Tag	Description	Example
CC	Coordinating conjunction	and, but, or	CD	Cardinal number	one, two, 123
DT	Determiner	the, a, some	EX	Existential there	there is, there was
FW	Foreign word	d'hoevre	IN	Preposition or subordinating conjunction	in, of, like
JJ	Adjective	big, happy	JJR	Adjective, comparative	bigger, happier
JJS	Adjective, superlative	biggest, happiest	LS	List item marker	1, One:
MD	Modal	can, could, will	NN	Noun, singular or mass	dog, car
NNS	Noun, plural	dogs, cars	NNP	Proper noun, singular	John, London
NNPS	Proper noun, plural	Vikings, Americans	PDT	Predeterminer	all, both
POS	Possessive ending	's	PRP	Personal pronoun	I, you, he
PRP$	Possessive pronoun	my, your, his	RB	Adverb	very, too, not
RBR	Adverb, comparative	better	RBS	Adverb, superlative	best
RP	Particle	up, off	SYM	Symbol	+, %, &
TO	to	to	UH	Interjection	ah, oops, ouch
VB	Verb, base form	eat, walk	VBD	Verb, past tense	ate, walked
VBG	Verb, gerund or present participle	eating, walking	VBN	Verb, past participle	eaten, walked
VBP	Verb, non-3rd person singular present	eat, walk	VBZ	Verb, 3rd person singular present	eats, walks
WDT	Wh-determiner	which, whatever	WP	Wh-pronoun	who, whoever
WP$	Possessive wh-pronoun	whose	WRB	Wh-adverb	where, when

NLTK 使用 Penn Treebank 定義之詞性標籤

```
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'),
 ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'),
 ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN'), ('.', '.')]
```

隨堂練習：英文詞性標註

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 載入必要的函式庫 NLTK
import nltk
from nltk.tokenize import word_tokenize

# 下載 NLTK 的 Tokenizer 和 POS Tagger 資源
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# 定義一個函數來執行詞性標註
def pos_tagging(text):
    # 斷詞
    words = word_tokenize(text)
    # 執行詞性標註
    pos_tags = nltk.pos_tag(words)
    return pos_tags

# 測試文本
text = "The quick brown fox jumps over the lazy dog."

# 獲得詞性標註結果
tagged_text = pos_tagging(text)
print("POS Tagging Result:", tagged_text)
```

```
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'),
 ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'),
 ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN'), ('.', '.')]
```

中文詞性標註

```
1  # 載入 spaCy 套件
2  import spacy
3
4  # 載入中文模型
5  nlp = spacy.load("zh_core_web_md")
6
7  # 定義繁體中文的句子
8  text = "我來到位於台北的政治大學校園。"
9
10 # 處理文本
11 doc = nlp(text)
12
13 # 輸出每個詞的文本和詞性標籤
14 for token in doc:
15     print(f"{token.text} ({token.pos_})")
```

Tag	Description	Tag	Description
ADJ	adjective	ADP	adposition
ADV	adverb	AUX	auxiliary
CONJ	conjunction	CCONJ	coordinating conjunction
DET	determiner	INTJ	interjection
NOUN	noun	NUM	numeral
PART	particle	PRON	pronoun
PROPN	proper noun	PUNCT	punctuation
SCONJ	subordinating conjunction	SYM	symbol
VERB	verb	X	other
SPACE	space		

我 (PRON)
來到 (VERB)
位於 (ADP)
台北 (PROPN)
的 (PART)
政治 (NOUN)
大學 (NOUN)
校園 (NOUN)
。 (PUNCT)

隨堂練習：中文詞性標註

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 載入 spaCy 套件
import spacy

# 載入中文模型
nlp = spacy.load("zh_core_web_md")

# 定義繁體中文的句子
text = "我來到位於台北的政治大學校園。"

# 處理文本
doc = nlp(text)

# 輸出每個詞的文本和詞性標籤
for token in doc:
    print(f"{token.text} ({token.pos_})")
```

我 (PRON)
來到 (VERB)
位於 (ADP)
台北 (PROPN)
的 (PART)
政治 (NOUN)
大學 (NOUN)
校園 (NOUN)
。 (PUNCT)

實體名稱識別

Named Entity Recognition

實體名稱識別簡介

- **何謂「實體名稱」（Named Entity）**
  - 具有特殊意義的詞彙。
  - 如：人名、地名、組織名...etc.

- **為何需要識別「實體名稱」？**
  - 抽取一篇文章中，具有特殊意義詞彙之關係。
  - 如：「台北101是台灣的一個地標，位於台北市。」

↑ 建築物名稱（FAC）　↑ 地區或國家（NORP）　↑ 城市名稱（GPE）

英文實體名稱識別

1
2
3
4
5
6

```python
# 載入 spaCy 套件
import spacy

# 載入英文的小型 NLP 模型
nlp = spacy.load("en_core_web_sm")

# 定義一個函數進行命名實體識別
def named_entity_recognition(text):
    # 處理文本
    doc = nlp(text)
    # 提取命名實體
    entities = [(ent.text, ent.label_) for ent in doc.ents]
    return entities

# 測試文本
text = "Apple Inc. announced its new iPhone in San Francisco."

# 執行命名實體識別
entities = named_entity_recognition(text)

# 印出識別出的實體
print("Identified Entities:", entities)
```

[('Apple Inc.', 'ORG'),
('iPhone', 'PRODUCT'),
('San Francisco', 'GPE')]

Entity Type	Description
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

隨堂練習：英文實體名稱識別

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 載入 spaCy 套件
import spacy

# 載入英文的小型 NLP 模型
nlp = spacy.load("en_core_web_sm")

# 定義一個函數進行命名實體識別
def named_entity_recognition(text):
    # 處理文本
    doc = nlp(text)
    # 提取命名實體
    entities = [(ent.text, ent.label_) for ent in doc.ents]
    return entities

# 測試文本
text = "Apple Inc. announced its new iPhone in San Francisco."

# 執行命名實體識別
entities = named_entity_recognition(text)

# 印出識別出的實體
print("Identified Entities:", entities)
```

```
[('Apple Inc.', 'ORG'),
 ('iPhone', 'PRODUCT'),
 ('San Francisco', 'GPE')]
```

中文實體名稱識別

```python
# 載入 spaCy 套件
import spacy

# 載入中文的中型 NLP 模型
nlp = spacy.load("zh_core_web_md")

# 定義繁體中文的句子
text = "台北101是台灣的一個地標，位於台北市。"

# 處理文本
doc = nlp(text)

# 印出識別到的命名實體
for ent in doc.ents:
    print(f"{ent.text} ({ent.label_})")
```

台北 (GPE)
101 (CARDINAL)
台灣 (GPE)
地標 (LOC)
台北市 (GPE)

Entity Type	Description
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

隨堂練習：中文實體名稱識別

- 請先撰寫好前一頁的原始碼 ，並且執行看看：

```python
# 載入 spaCy 套件
import spacy

# 載入中文的中型 NLP 模型
nlp = spacy.load("zh_core_web_md")

# 定義繁體中文的句子
text = "台北101是台灣的一個地標，位於台北市。"

# 處理文本
doc = nlp(text)

# 印出識別到的命名實體
for ent in doc.ents:
    print(f"{ent.text} ({ent.label_})")
```

台北 (GPE)
101 (CARDINAL)
台灣 (GPE)
地標 (LOC)
台北市 (GPE)