

# 補充說明

---

# 產學合作計畫

## 義守大學醫學核心專題研究計畫補助辦法

108 年 08 月 21 日行政會議通過（全文），108 年 08 月 30 日校長核定公告

**第一條** 為鼓勵專任教師積極與醫學相關領域進行多元合作研究，提升醫學院研究能量、跨領域及與新興科技領域結合等合作發展，特訂定「義守大學醫學核心專題研究計畫補助辦法」（以下簡稱本辦法）。

**第二條** 為發展本校醫學特色研究，本辦法補助計畫（以下簡稱專題計畫）區分為以下二類型：

- 一、醫學核心個別型計畫。
- 二、醫學核心整合型計畫。

**第三條** 醫學核心個別型計畫為與醫學相關之跨領域計畫，主持人應由本校分屬醫學院及非醫學院之二位專任教師共同擔任，並推派一位負責計畫聯絡及經費核銷工作。

**第四條** 醫學核心整合型計畫為與醫學相關之跨領域整合計畫，其中應包含三個以上之子計畫，每一子計畫執行期間為二至三年。子計畫主持人應為本校專任教師，其中至少一個子計畫主持人應為本校醫學院專任教師。

**第五條** 各類型專題計畫提出時間，於每年公告時程向研究發展處學術發展組（以下簡稱學發組）提出申請，每位教師每學年於各類型計畫以申請一件為限。

**第六條** 申請之專題計畫，不得重覆申請同一學年度本校校內專題研究計畫、校內專題產學合作計畫、與義大醫療財團法人研究合

# Python 語法

---

- 套件: NLTK/Scikit-Learn/Keras
- set/dictionary/tuple/list 語法

# Structured Learning

---

- Output: Scalar/Class
- Output: Sequence, Matrix, Graph, Tree,...
  - Sequence
    - Machine Translation
    - Speech Recognition
    - Chat-bot
  - Matrix
    - Image to Image
    - Text to Image

# Structured Learning

The screenshot shows a browser window titled "Text to Speech Demo". The URL is [text-to-speech-demo.ng.bluemix.net](https://text-to-speech-demo.ng.bluemix.net). The page has two main sections: "Input Text" and "Voice Selection".  
The "Input Text" section contains a text area with the following text:

The text language must match the selected voice language: Mixing language (English text with a Spanish male voice) does not produce valid results. The synthesized audio is streamed to the client as it is being produced, using the HTTP chunked encoding. The audio is returned in mp3 format which can be played using VLC and Audacity players.

  
The "Voice Selection" section contains a dropdown menu set to "American English (en-US): AlisonV3 (female, enhanced dnn)".  
Below the browser window, the URL <https://text-to-speech-demo.ng.bluemix.net/> is displayed.

The screenshot shows a browser window titled "Text To Image API | DeepAI". The URL is [deepai.org/machine-learning-model/text2img](https://deepai.org/machine-learning-model/text2img). The page displays four generated images: a white wolf, a red mushroom, spaghetti, and a fountain.

Text To Image API

by Scott Ellison Reed - ❤ 52 · ⚡ share

Creates an image from scratch from a text description.

Submit

API Docs

Below the browser window, the URL <https://deepai.org/machine-learning-model/text2img> is displayed.

The screenshot shows a browser window titled "Caption generation demo". The URL is <https://milhidaka.github.io/chainer-image-caption/>. The page features a "Generate image caption demo" heading and an "Input image (can drag-drop image file):" section with a preview of a cityscape image. Below the image is a "Generate caption" button and a "浏览..." (Browse...) button.

Generating image caption demo

Input image (can drag-drop image file):

Generate caption

浏览...

Load models > Analyze image > Generate text

The screenshot shows a browser window titled "Chatbots with Seq2Seq". The URL is <http://complx.me/2016-06-28-easy-seq2seq/>. The page features a header with the author's name "Suriyadeepan Ram" and a GitHub logo. The main content is a blog post titled "Chatbots with Seq2Seq" with the subtitle "Learn to build a chatbot using TensorFlow".

ABOUT ME SLIDES ▾ PROFILE

Suriyadeepan Ram

Chatbots with Seq2Seq

Learn to build a chatbot using TensorFlow

Posted on June 28, 2016

全部顯示

# Structured Learning

- Output: Matrix
  - Image to Image
  - Text to Image

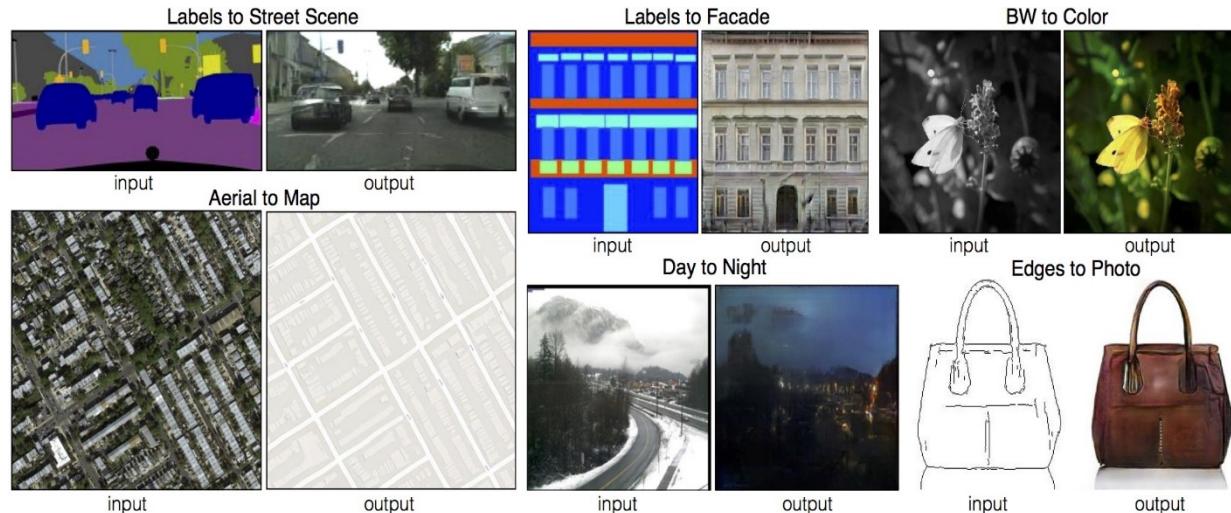


Image-to-Image Translation with Conditional Adversarial Nets

# Scikit-Learn Packages

The screenshot shows a web browser window displaying a DataCamp community tutorial titled "Scikit-Learn 教學 : Python 與機器學習". The page content discusses the `sklearn` package and its `datasets` module, specifically the `load\_digits()` function. It includes a code editor showing a script named `script.py` with the following code:

```
script.py  IPython Shell
1 # Import `datasets` from `sklearn`
2 from sklearn import __
3
4 # Load in the `digits` data
5 digits = datasets.load_digits()
6
7 # Print the `digits` data
8 print(__)
```

The page also features a sidebar with navigation links like News, Resource Center, Tutorials, Cheat Sheets, Open Courses, Podcast - DataFramed, Chat, and Official Blog. There are social sharing icons for Facebook, Twitter, and LinkedIn, along with a comment section.

<https://www.datacamp.com/community/tutorials/scikit-learn-python>

# 中文自然語言

---

- 他一邊脫衣服，一邊穿褲子。
- 我的其中一只左腳受傷了
- 下班了，爸爸陸陸續續的回家了。
- 我家門前有條水溝很難過。
- 你看什麼看！沒看過啊
- 欣欣向榮榮告白。
- 今天真熱。
- 先生，再見！
- 一列火車經過，況且況且況且況且況
- 這個婆娘不是人，九天仙女下凡塵

# 自然語言處理應用



[https://www.youtube.com/watch?v=iqHAL\\_8Ug2Y](https://www.youtube.com/watch?v=iqHAL_8Ug2Y)

# 台灣本土AI「雅婷」誕生

The screenshot shows a news article titled "PTT創辦人杜奕瑾成功打造人工智能語音辨識 APP:雅婷逐字稿". The article discusses the creation of an AI transcription app called "YaTing" by the founder of PTT, Du Yijin. It highlights that the app can convert speech into text, which is particularly useful for people who need to transcribe audio recordings. The article includes a video thumbnail showing a person speaking into a microphone and a smartphone displaying the app's interface.

PTT 創世神新創舉：台灣本土AI「雅婷」誕生

被鄉民稱為「PTT創世神」的PTT創辦人杜奕瑾，日前回到台灣，創辦了台灣人工智慧實驗室（Taiwan AI Labs）。他們推出了MIT台灣在地自行開發的AI人工智慧APP「雅婷逐字稿」。這個AI「雅婷」可以將語音轉成逐字稿，對於經常需要藉由聽口述而打逐字稿的人，如記者等來說，簡直就是一個福音！這套人工智慧語音辨識APP也支援了Android、iOS雙平台。

業者做出明音的商業決策  
● 是Boss就需要SEO!7個理由:3. SEO是免費的  
● 是Boss就需要SEO!7個理由:2. SEO帶來更高的流量與轉換  
● 是Boss就需要SEO!7個理由:1. 為新創業者帶來客戶行為資料

## 分類

- AI課程 (61)
- HTML5教學 (37)
- Java (1)
- photoshop (2)
- Python課程 (55)
- SEO優化 (25)
- 前端工程師 (46)
- 未分類 (33)
- 程式語言 (4)
- 網路行銷課程 (24)
- 設計課程 (45)
- 達內教育 (4)

# Class 02: 自然語言處理

---

林義隆  
義守大學 資訊工程系 副教授

# 自然語言處理

---

- 問答 Question Answering
- 機器翻譯 Machine Translation
- 摘要 Summarization
- 自然語言推理 Natural Language Inference
- 情感分析 Sentiment Analysis
- 語意角色標註 Semantic Role Labeling
- 關聯萃取 Relation Extraction
- 目標導向對話 Goal-oriented Dialogue
- 語意剖析 Database Query Generation
- 代名詞解析 Pronoun resolution

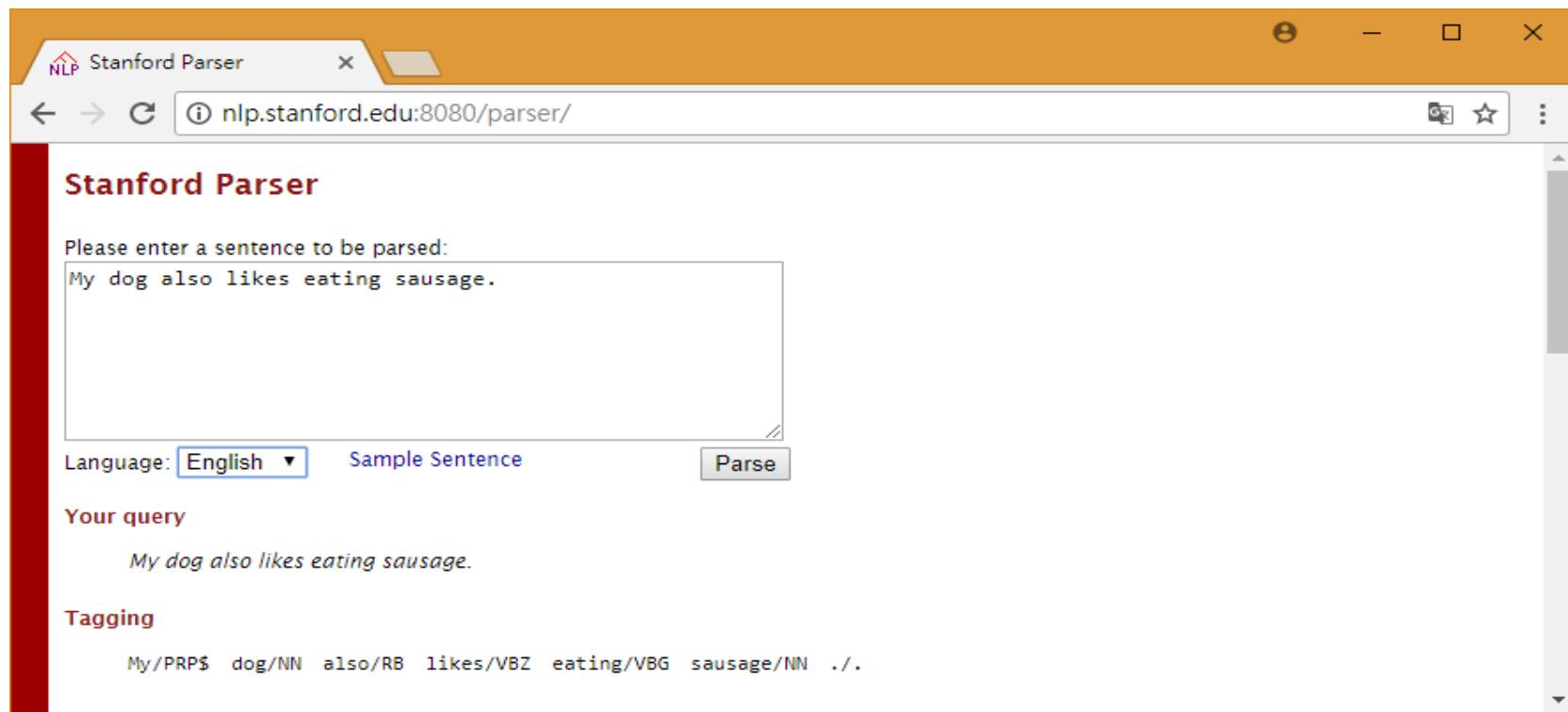
<https://blog.einstein.ai/the-natural-language-decathlon/>

# 自然語言工處理開源套件

---

- 自然語言工具箱 (NLTK)
- 斯丹福自然語言處理 (Stanford NLP)
- 哈工大語言技術平台 (LTP)
- 台灣中央研究院

# Stanford NLP



<http://nlp.stanford.edu:8080/parser/index.jsp>

# NLTK

```
import nltk  
nltk.download()  
python -m nltk.downloader all
```

File View Sort Help

Collections Corpora Models All Packages

Identifier	Name	Size	Status
rte	PASCAL RTE Challenges 1, 2, and 3	377.2 KB	installed
sample_grammars	Sample Grammars	19.8 KB	not installed
semcor	SemCor 3.0	4.2 MB	installed
senseval	SENSEVAL 2 Corpus: Sense Tagged Text	2.1 MB	installed
sentence_polarity	Sentence Polarity Dataset v1.0	478.8 KB	not installed
sentiwordnet	SentiWordNet	4.5 MB	out of date
shakespeare	Shakespeare XML Corpus Sample	464.3 KB	not installed
sinica_treebank	Sinica Treebank Corpus Sample	878.2 KB	installed
smultron	SMULTRON Corpus Sample	162.3 KB	not installed
snowball_data	Snowball Data	6.5 MB	not installed
spanish_grammars	Grammars for Spanish	4.0 KB	not installed
state_union	C-Span State of the Union Address Corpus	789.8 KB	installed
stopwords	Stopwords Corpus	10.2 KB	out of date
subjectivity	Subjectivity Dataset v1.0	509.4 KB	not installed
swadesh	Swadesh Wordlists	22.3 KB	installed
switchboard	Switchboard Corpus Sample	772.6 KB	not installed

Download Refresh

Server Index: [https://raw.githubusercontent.com/nltk/nltk\\_data/gh-pages/](https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/)

Download Directory: E:\nltk\_data

<http://www.nltk.org/>

NLTK Data www.nltk.org/nltk\_data/

## NLTK Corpora

NLTK has built-in support for dozens of corpora and trained models, as listed below. To use these within NLTK we recommend that you use the NLTK corpus downloader, >>> `nltk.download()`

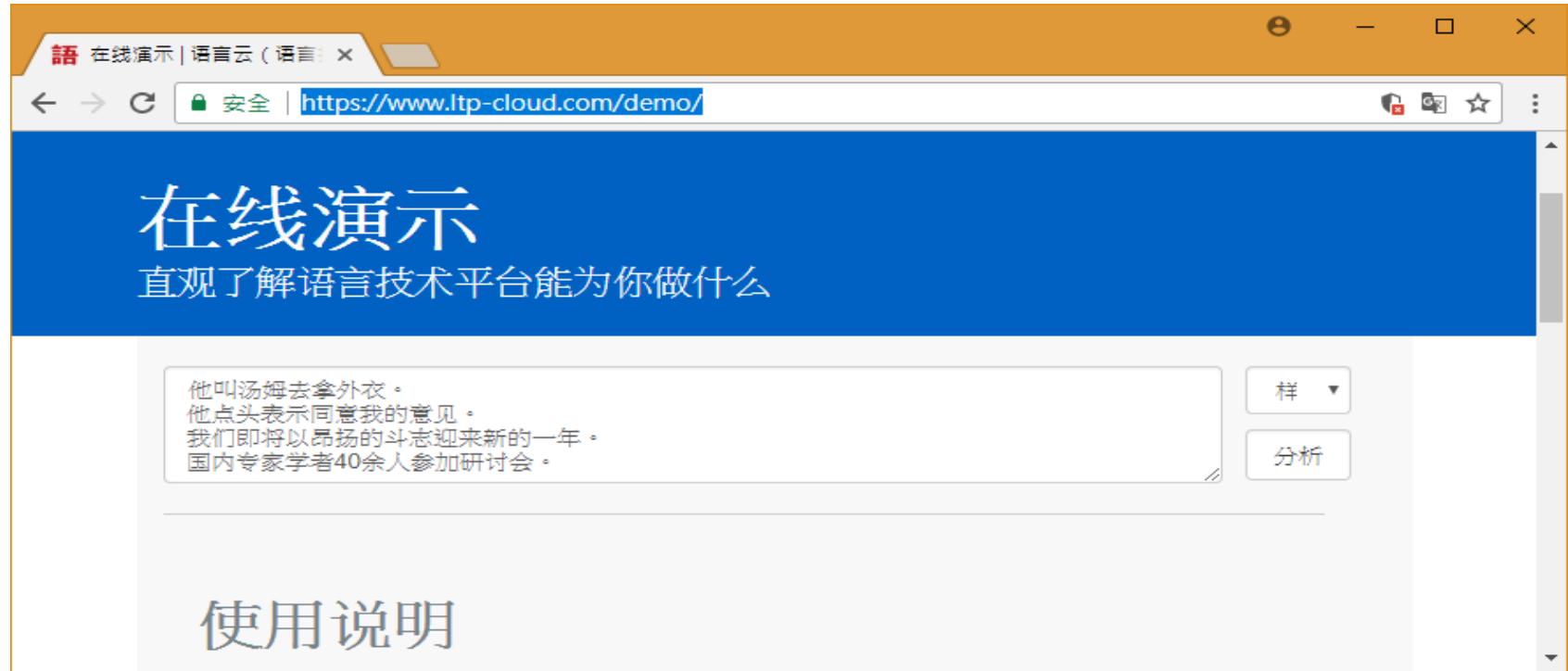
Please consult the README file included with each corpus for further information.

- ACE Named Entity Chunker (Maximum entropy) [[download](#) | [source](#)]  
id: maxent\_ne\_chunker; size: 13404747; author: ; copyright: ; license: ;
- Australian Broadcasting Commission 2006 [[download](#) | [source](#)]  
id: abc; size: 1487851; author: Australian Broadcasting Commission; copyright: ; license: ;
- Alpino Dutch Treebank [[download](#) | [source](#)]  
id: alpino; size: 2797255; author: ; copyright: ; license: Distributed with permission of Gertjan van Noord;
- BioCreative (Critical Assessment of Information Extraction Systems in Biology) [[download](#) | [source](#)]  
id: biocreative\_ppi; size: 223566; author: ; copyright: Public Domain (not copyrighted); license: Public Domain;
- Brown Corpus [[download](#) | [source](#)]  
id: brown; size: 3314357; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes.;
- Brown Corpus (TEI XML Version) [[download](#) | [source](#)]  
id: brown\_tei; size: 8737738; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes.;

[http://www.nltk.org/nltk\\_data/](http://www.nltk.org/nltk_data/)

# LTP

- Language Technology Platform (LTP)
- pip install pyltp



语言技术平台云 <https://www.ltp-cloud.com/demo/>

# CKIP

The screenshot shows a web browser window for the CKIP Chinese Segmentation System. The address bar displays the URL [ckipsvr.iis.sinica.edu.tw](http://ckipsvr.iis.sinica.edu.tw). The main content area features a red header with the text "中文斷詞系統". Below the header, there is a navigation menu with links: [授權辦法], 相關系統, 斷詞系統, 剖析系統, 詞首詞尾, 平衡語料庫, 廣義知網, and 句結構樹庫. On the left side, there is a sidebar with a list of links: 簡介, 未知詞擴取做法, 詞類標記列表, 線上展示, 線上服務申請, 線上資源, 公告, and 聯絡我們. The main content area contains two large blocks of text. The first block discusses the nature of words as the smallest meaningful units in language processing. It mentions that automatic segmentation is a crucial technique for further processing like machine translation and language analysis. The second block discusses the challenges of automatic segmentation for Chinese, such as handling domain-specific vocabulary and new words, and how the system addresses these issues.

<http://ckipsvr.iis.sinica.edu.tw/>

# 課程大綱

---

- 本課程與自然語言處理之關聯
- 斷辭斷字
- 去停用字
- 詞向量表示
- 詞袋模型

# 課程重點

---

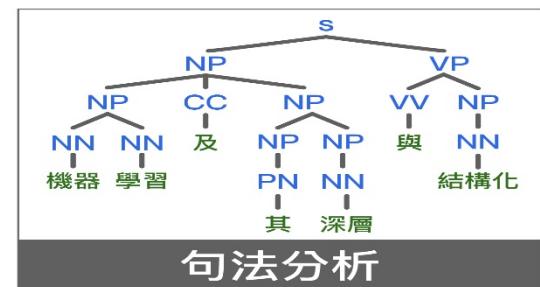
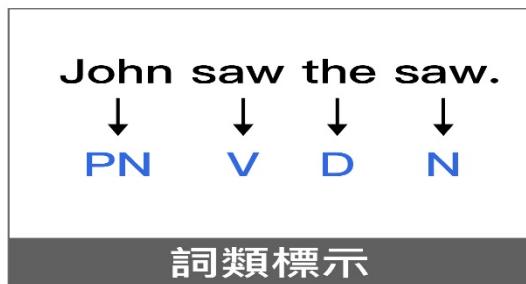
- 句子分割
- 去停用字
- 文字編碼
- 詞彙
- 使用Python字典模型建立
  - Index to word
  - Word to index

# 自然語言處理



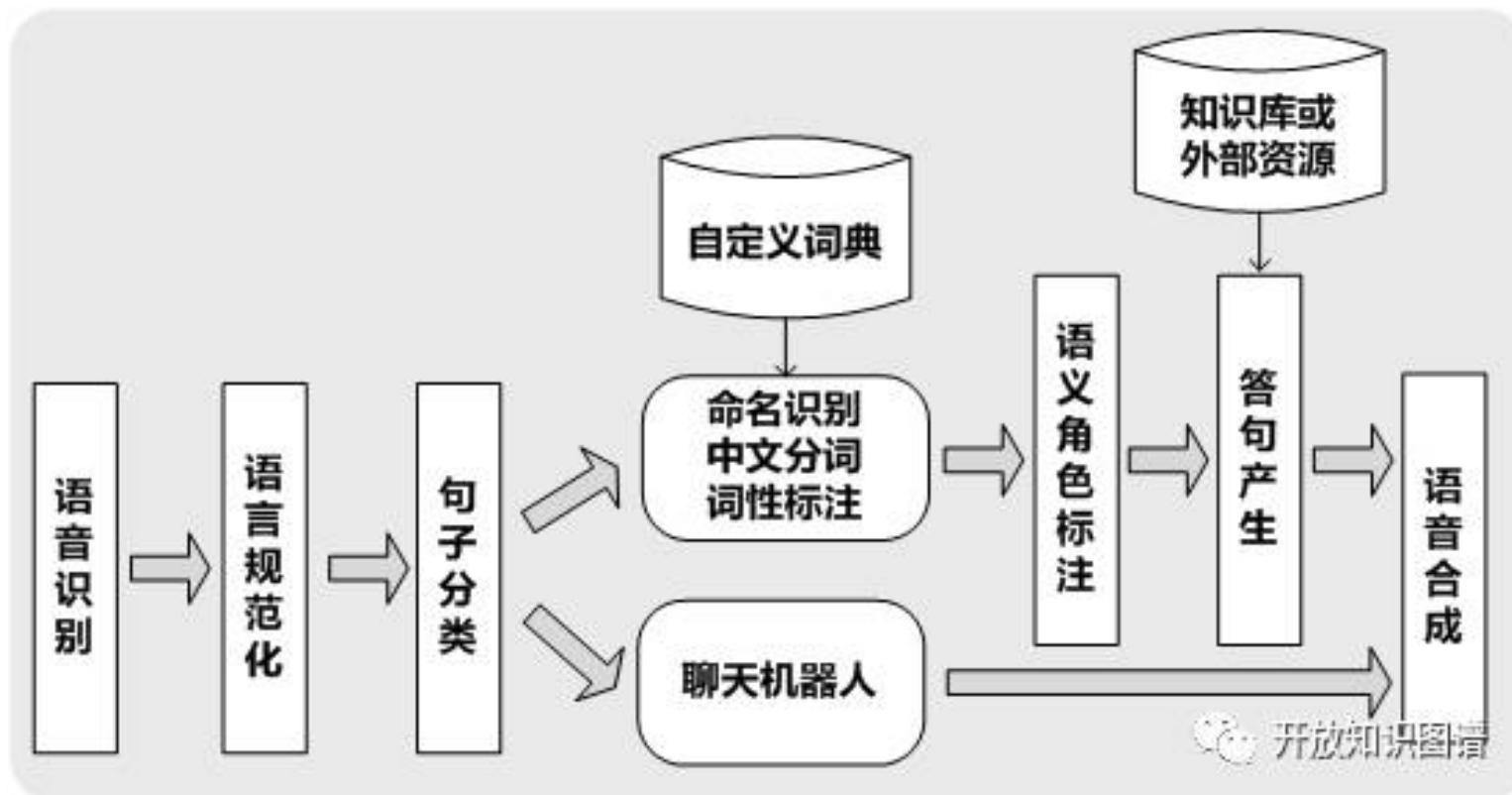
<http://www.gdhcfunds.com/index.php?c=article&id=338>

# 自然語言處理

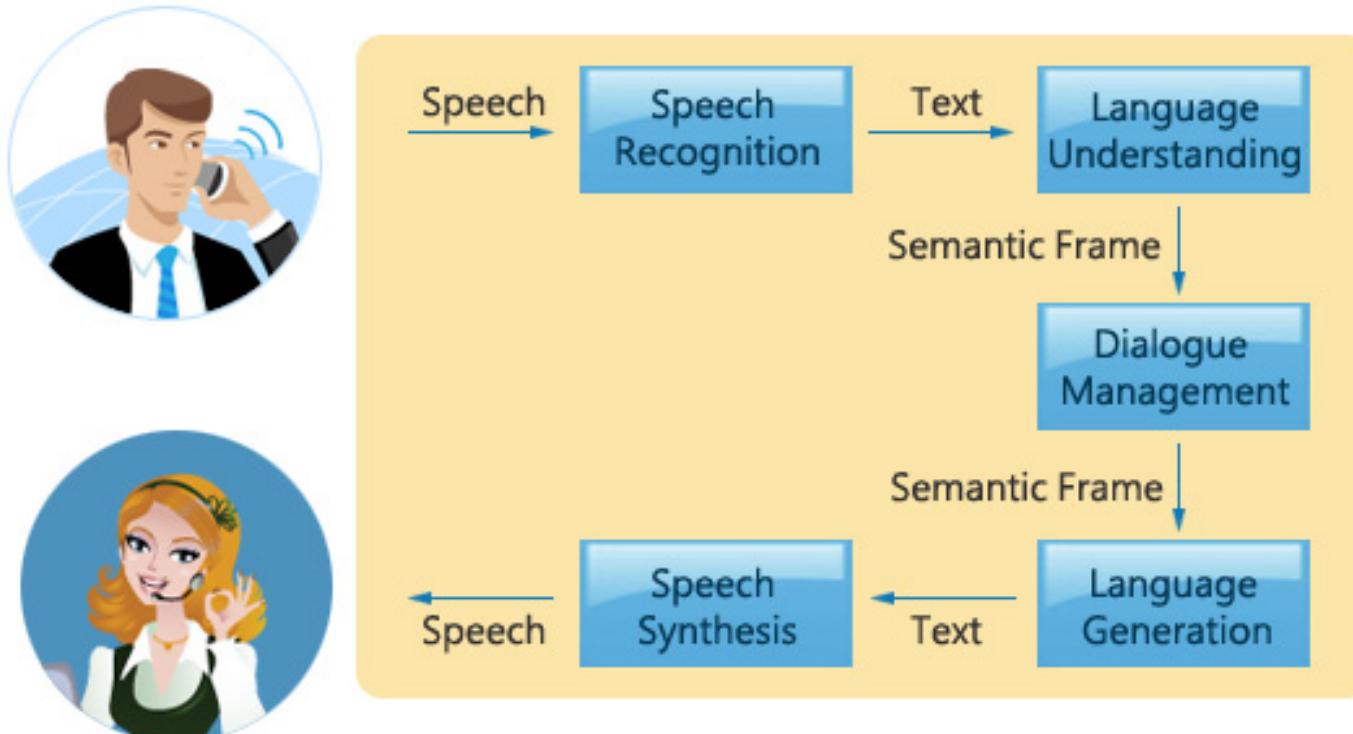


資料來源 | 中研院: 研之有物

# 對話系統



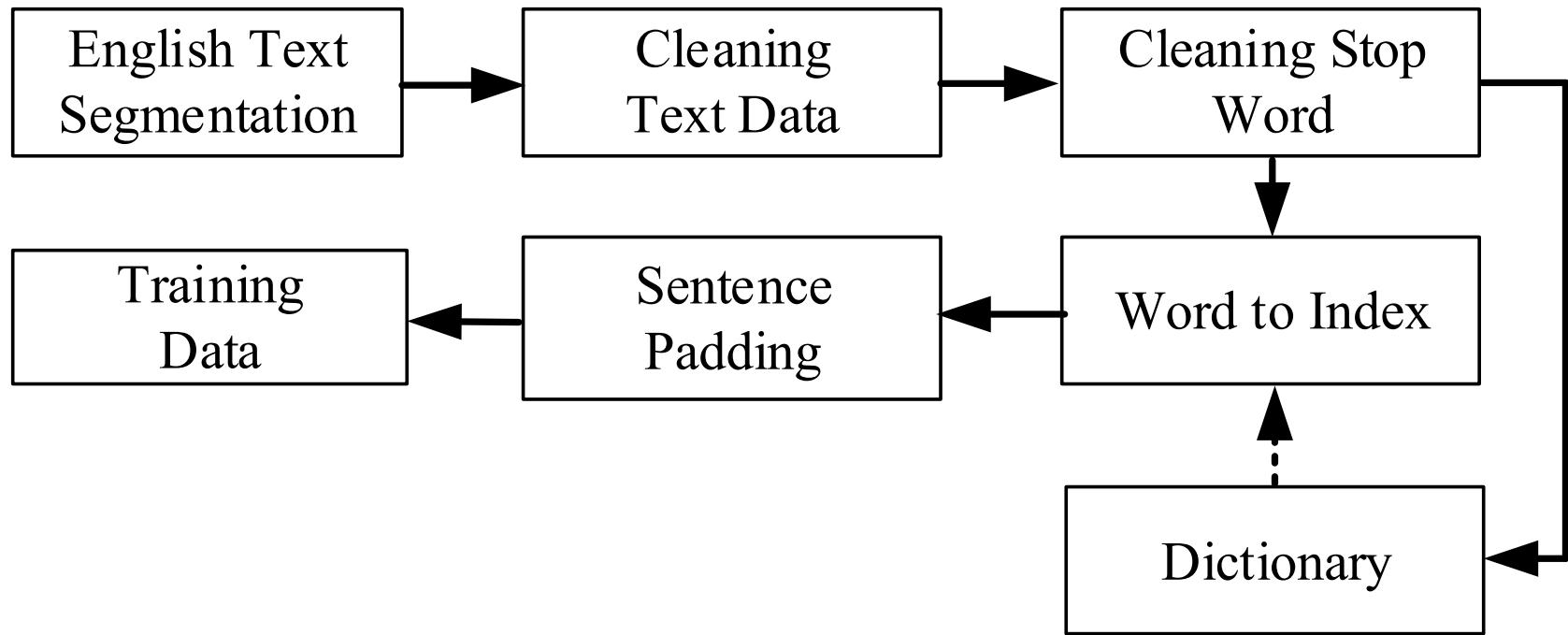
# 對話系統



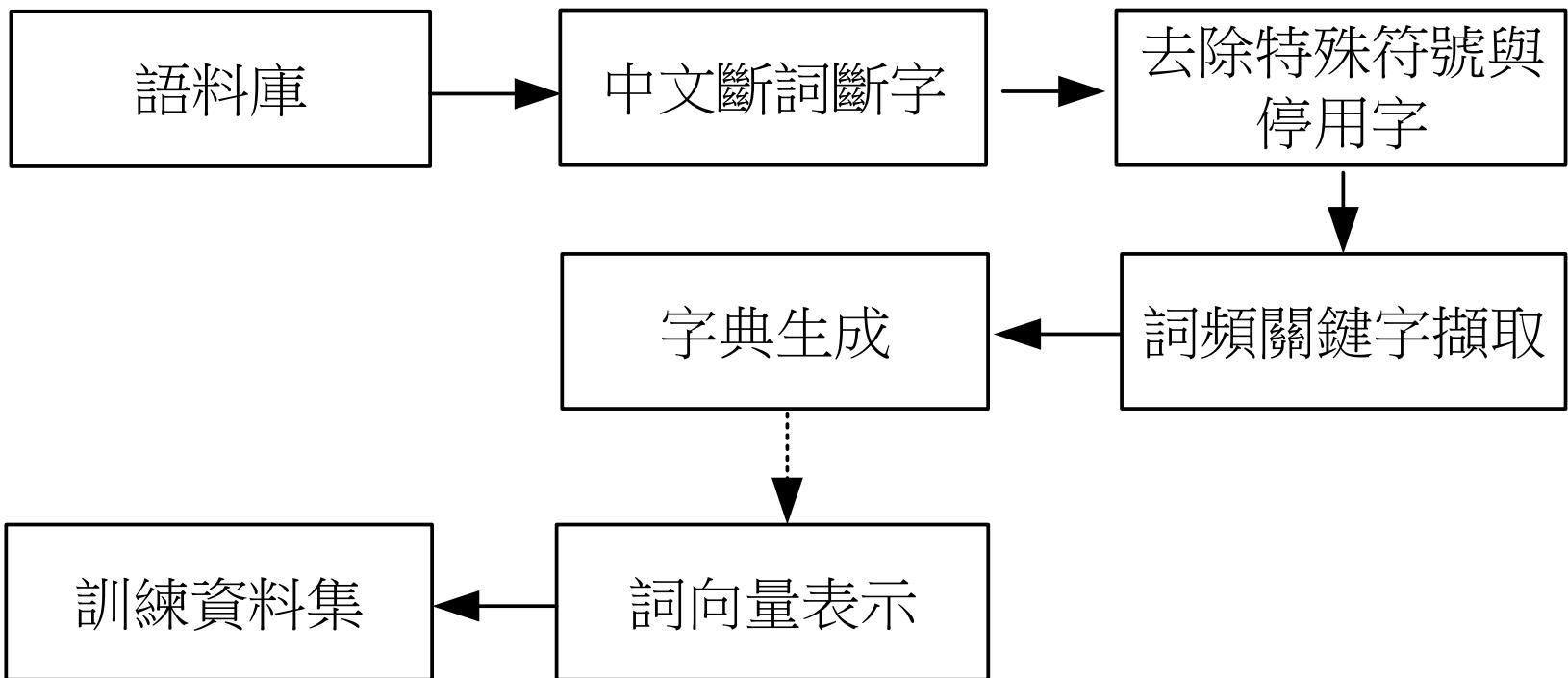
口語對話系統架構示意圖

<http://atc.ccl.itri.org.tw/interaction/conversation.php>

# 英文前處理



# 中文前處理



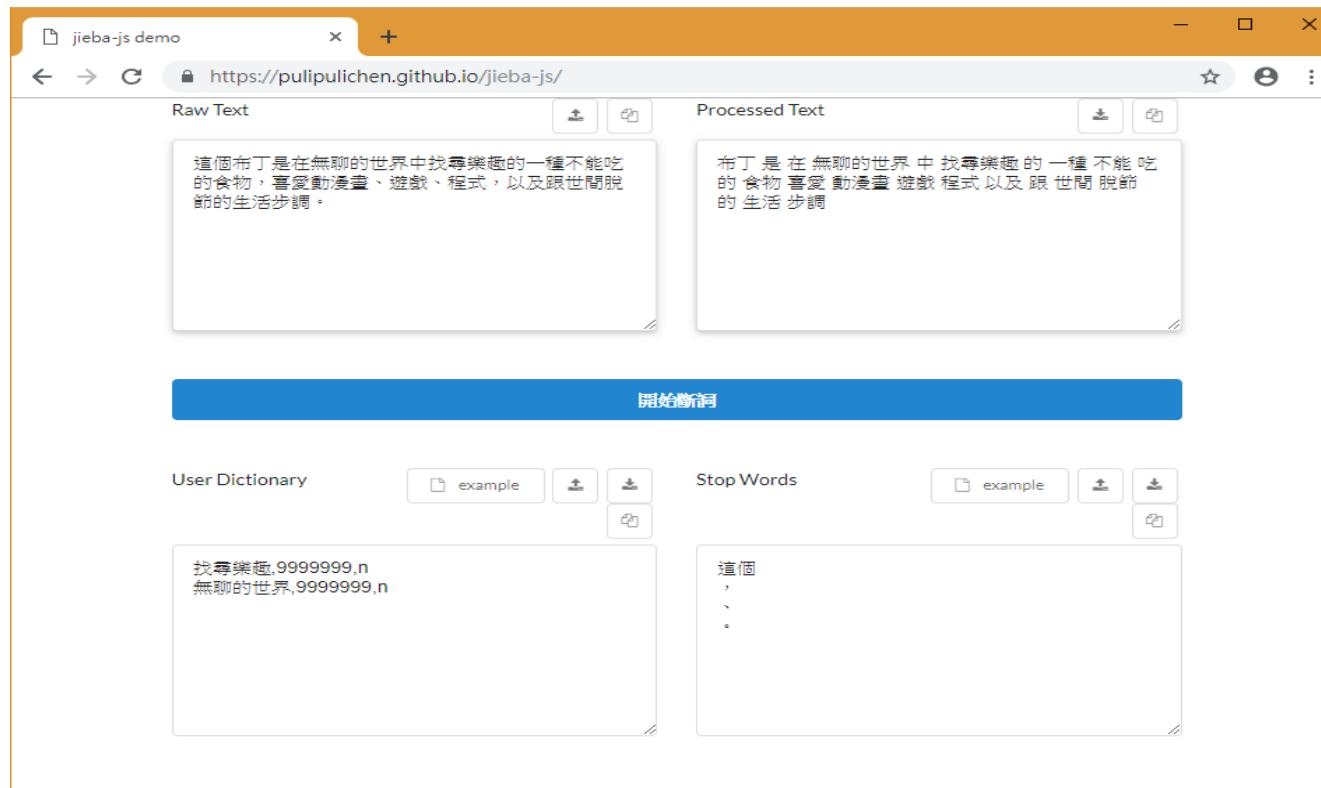
# 中文分詞器

---

- KIP: 台灣中研院研發的一款斷詞器，不過並未對外公布技術節。
- HanLP: 這是一個開源的分詞器(java),我在這篇Hanlp自然語言處理器有使用範例
- Ansj: 這也是一個開源的中文分詞器(java)
- jieba: Python 的中文分詞器

# 中文分詞器

- Jieba Package



<https://pulipulichen.github.io/jieba-js/>

# Jieba 中文斷詞器

---

- Raw Text
- Processed Text
- User Dictionary (user\_dict.txt)
- Stop Words (stop\_words.txt)

# 基於Jieba的詞性標註

```
In [9]: import jieba
import jieba.posseg as psg
s='皮膚科在哪裡？'
cut = jieba.cut(s)
print(','.join(cut))
print([(x.word,x.flag) for x in psg.cut(s)])
皮膚科,在,哪裡,?
[('皮膚', 'n'), ('科', 'n'), ('在', 'p'), ('哪裡', 'r'), ('?', 'x')]

In [10]: s='陳大義醫師在哪裡？'
cut = jieba.cut(s)
print(','.join(cut))
陳,大義,醫師,在,哪裡,?
[('陳', 'n'), ('大', 'n'), ('義', 'n'), ('醫師', 'n'), ('在', 'p'), ('哪裡', 'r'), ('?', 'x')]

In [11]: s='心臟科在哪裡？'
cut = jieba.cut(s)
print(','.join(cut))
print([(x.word,x.flag) for x in psg.cut(s)])
心臟科,在,哪裡,?
[('心臟', 'n'), ('科', 'n'), ('在', 'p'), ('哪裡', 'r'), ('?', 'x')]

In [12]: cut = jieba.cut(s)
print([(x.word,x.flag) for x in psg.cut(s)])
s='腦外科怎麼走？'
cut = jieba.cut(s)
print(','.join(cut))
[('心臟', 'n'), ('科', 'n'), ('在', 'p'), ('哪裡', 'r'), ('?', 'x')]
[('腦', 'zg'), ('外', 'n'), ('科', 'n'), ('怎', 'zg'), ('麼', 'zg'), ('走', 'zg')]

In [14]: print([(x.word,x.flag) for x in psg.cut(s)])
s='腦外科在哪裡？'
cut = jieba.cut(s)
print(','.join(cut))
[('腦', 'zg'), ('外', 'n'), ('科', 'n'), ('在', 'p'), ('哪裡', 'r'), ('?', 'x')]

In [15]: print([(x.word,x.flag) for x in psg.cut(s)])
[('腦', 'zg'), ('外', 'n'), ('科', 'n'), ('在', 'p'), ('哪裡', 'r'), ('?', 'x')]
```

# 問題討論

---

- 使用Python建立字典

# 基於Jieba的詞性標註

```
In [7]: import jieba  
import jieba.posseg as psg  
s='周兆明老師在哪裡'  
cut = jieba.cut(s)  
print(','.join(cut))  
print([(x.word,x.flag) for x in psg.cut(s)])
```

周兆明,老師,在,哪裡  
[('周兆明', 'nr'), ('老師', 'n'), ('在', 'p'), ('哪裡', 'r')]

```
In [8]: s='我要找王小明老師'  
cut = jieba.cut(s)  
print(','.join(cut))  
print([(x.word,x.flag) for x in psg.cut(s)])
```

我要,找,王小明,老師  
[('我', 'r'), ('要', 'v'), ('找', 'v'), ('王小明', 'nr'), ('老師', 'n')]

```
In [9]: s='丁銚在哪裡'  
cut = jieba.cut(s)  
print(','.join(cut))  
print([(x.word,x.flag) for x in psg.cut(s)])
```

丁,銚,在,哪裡  
[('丁銚', 'nr'), ('在', 'p'), ('哪裡', 'r')]

```
In [10]: s='我要找丁銚'  
cut = jieba.cut(s)  
print(','.join(cut))  
print([(x.word,x.flag) for x in psg.cut(s)])
```

我要,找,丁,銚  
[('我', 'r'), ('要', 'v'), ('找', 'v'), ('丁', 'nr'), ('銚', 'n')]

# 詞袋模型簡介

---

- One-Hot Encoding
- 字母形式/字串形式轉換成數字形式
- Token (字符)
- Vocabulary (詞彙)
  - 整個文件中的字詞(word)
- 文件的特徵
  - 特徵向量
  - 字詞與出現的次數

# 字詞(word)轉特徵向量

---

- One Hot Encoding
- 字詞出現的次數(Count)建立詞袋
  - 單元模型(1-gram model或unigram model)
- 詞頻-反向文件詞頻(TF-IDF)
- 詞特徵向量

# One-Hot Encoding

- 使用 keras 的 Tokenizer 類別

```
#OneHotEncoder
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
encoder.fit([
    ['A', 'C', 'B', 'L'],
    ['B', 'D', 'F', 'D'],
    ['C', 'D', 'C', 'L'],
    ['B', 'C', 'E', 'D']
])
encoded_vector = encoder.transform([['C', 'D', 'F', 'D']]).toarray()
print("\n Encoded vector =", encoded_vector)
```

Encoded vector = [[0. 0. 1. 0. 1. 0. 0. 0. 1. 1. 0.]]

# 次數詞袋範例說明

- 使用scikit-learn中的 CountVectorizer 類別

```
from sklearn.feature_extraction.text import CountVectorizer
# List of text documents
text = ["The quick brown fox jumped over the lazy dog."]
# create the transform
vectorizer = CountVectorizer()
# tokenize and build vocab
vectorizer.fit(text)
# summarize
print(vectorizer.get_feature_names())
print(vectorizer.vocabulary_)
# encode document
vector = vectorizer.transform(text)
# summarize encoded vector
print(vector.shape)
print(type(vector))
print(vector.toarray())

['brown', 'dog', 'fox', 'jumped', 'lazy', 'over', 'quick', 'the']
{'the': 7, 'quick': 6, 'brown': 0, 'fox': 2, 'jumped': 3, 'over': 5, 'lazy': 4, 'dog': 1}
(1, 8)
<class 'scipy.sparse.csr.csr_matrix'>
[[1 1 1 1 1 1 1 2]]
```

# TF-IDF 詞袋

---

- 使用 scikit-learn 中的 TfidfTransformer

When we are analyzing text data, we often encounter words that occur across multiple documents from both classes. Those frequently occurring words typically don't contain useful or discriminatory information. In this subsection, we will learn about a useful technique called term frequency-inverse document frequency (tf-idf) that can be used to downweight those frequently occurring words in the feature vectors. The tf-idf can be defined as the product of the term frequency and the inverse document frequency:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \times \text{idf}(t, d)$$

Here the  $\text{tf}(t, d)$  is the term frequency that we introduced in the previous section, and the inverse document frequency  $\text{idf}(t, d)$  can be calculated as:

$$\text{idf}(t, d) = \log \frac{n_d}{1 + \text{df}(d, t)},$$

where  $n_d$  is the total number of documents, and  $\text{df}(d, t)$  is the number of documents  $d$  that contain the term  $t$ . Note that adding the constant 1 to the denominator is optional and serves the purpose of assigning a non-zero value to terms that occur in all training samples; the log is used to ensure that low document frequencies are not given too much weight.

<https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch08/ch08.ipynb>

# TF-IDF 詞袋

---

- 使用 scikit-learn 中的 TfidfTransformer

However, if we'd manually calculated the tf-idfs of the individual terms in our feature vectors, we'd have noticed that the TfidfTransformer calculates the tf-idfs slightly differently compared to the standard textbook equations that we defined earlier. The equations for the idf and tf-idf that were implemented in scikit-learn are:

$$\text{idf}(t, d) = \log \frac{1 + n_d}{1 + \text{df}(d, t)}$$

The tf-idf equation that was implemented in scikit-learn is as follows:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \times (\text{idf}(t, d) + 1)$$

While it is also more typical to normalize the raw term frequencies before calculating the tf-idfs, the TfidfTransformer normalizes the tf-idfs directly.

By default (`norm='l2'`), scikit-learn's TfidfTransformer applies the L2-normalization, which returns a vector of length 1 by dividing an un-normalized feature vector  $v$  by its L2-norm:

$$v_{\text{norm}} = \frac{v}{\|v\|_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \dots + v_n^2}} = \frac{v}{(\sum_{i=1}^n v_i^2)^{\frac{1}{2}}}$$

To make sure that we understand how TfidfTransformer works, let us walk through an example and calculate the tf-idf of the word is in the 3rd document.

<https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch08/ch08.ipynb>

# TF-IDF 詞袋範例說明

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
# List of text documents
text = ["The quick brown fox jumped over the lazy dog.", "The dog.", "The fox"]
# create the transform
vectorizer = TfidfVectorizer()
# tokenize and build vocab
vectorizer.fit(text)
np.set_printoptions(precision=5)
# summarize
print(vectorizer.vocabulary_)
print(vectorizer.get_feature_names())
print(vectorizer.idf_)
vector = vectorizer.transform(text)
#summarize encoded vector
print(vector.shape)
print(vector.toarray())

{'the': 7, 'quick': 6, 'brown': 0, 'fox': 2, 'jumped': 3, 'over': 5, 'lazy': 4, 'dog': 1}
['brown', 'dog', 'fox', 'jumped', 'lazy', 'over', 'quick', 'the']
[1.69315 1.28768 1.28768 1.69315 1.69315 1.69315 1.69315 1.69315]
(3, 8)
[[0.36389 0.27675 0.27675 0.36389 0.36389 0.36389 0.36389 0.42983]
 [0. 0.78981 0. 0. 0. 0. 0.61336]
 [0. 0. 0.78981 0. 0. 0. 0. 0.61336]]
```

the

# 詞向量

- 另一種方法來表達詞彙
  - “Word to Vector” or “Word Embedding”



<https://ckmarkoh.github.io/blog/2016/07/10/nlp-vector-space-semantics/>

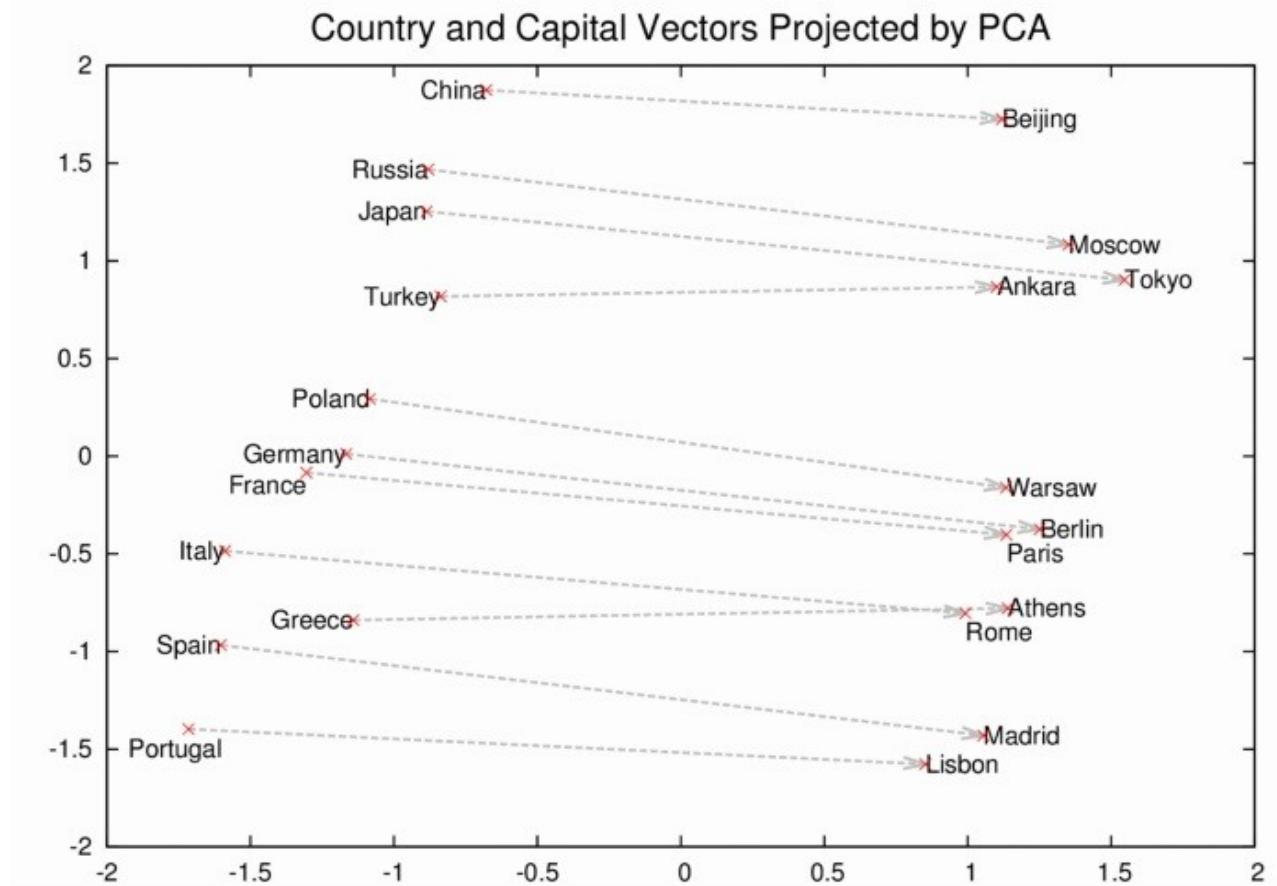
# Word Embedding

---

- Google's **word2vec** [Mikolov et al. 2013]
  - COBW
  - Skip-gram
- **GloVe** [Levy et al. 2014 ]

<https://code.google.com/archive/p/word2vec/>  
<https://nlp.stanford.edu/projects/glove/>

# Word to Vector



Google Open Source Blog

# Word to Vec JS Demo

The screenshot shows a web browser window with the URL <http://turbomaze.github.io/word2vecjson/>. The page displays a list of words and their corresponding numerical values, likely word embeddings. A scroll bar is visible on the right side of the content area.

Word	Value
daughter	0.8706230019086026
grandmother	0.8442235585377782
aunt	0.8435924872067456
niece	0.8070082160701633
father	0.7901481870716061
son	0.7683207312810011
sister	0.763335126711493
wife	0.755067714099042
stepmother	0.7531879305425351

## Word Algebra

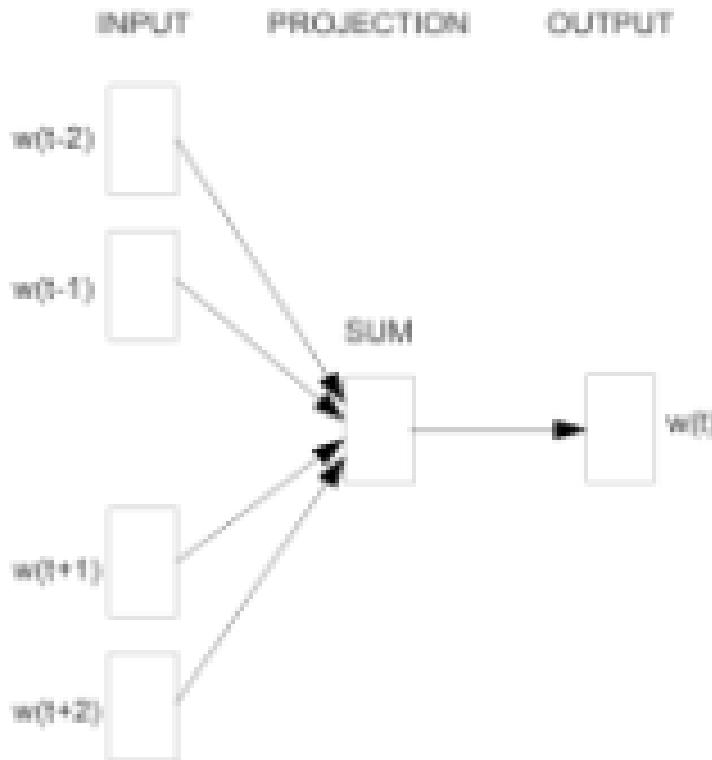
Enter all three words, the first two, or the last two and see the words that result.

+ ( - ) =

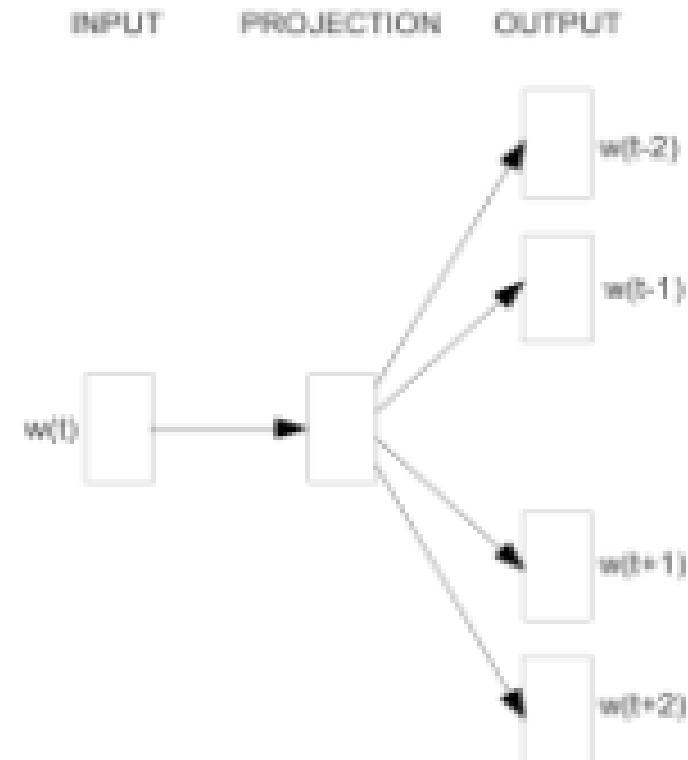
son      0.9048675802514527  
father    0.8820846506379102

<http://turbomaze.github.io/word2vecjson/>

# BOW & Skip-Gram Models



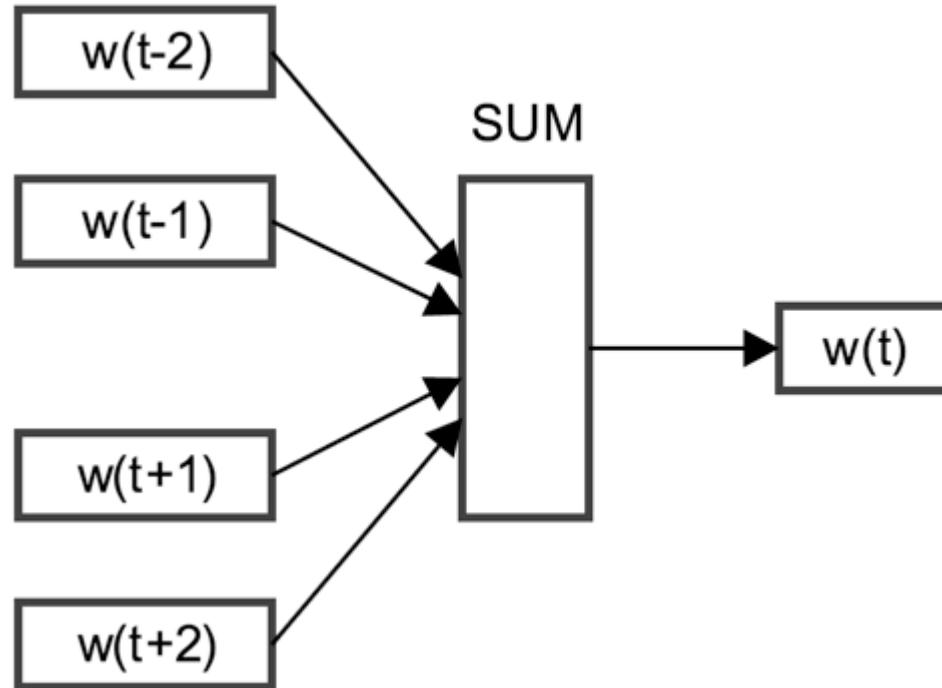
CBOW



Skip-gram

# Bag of Words Model

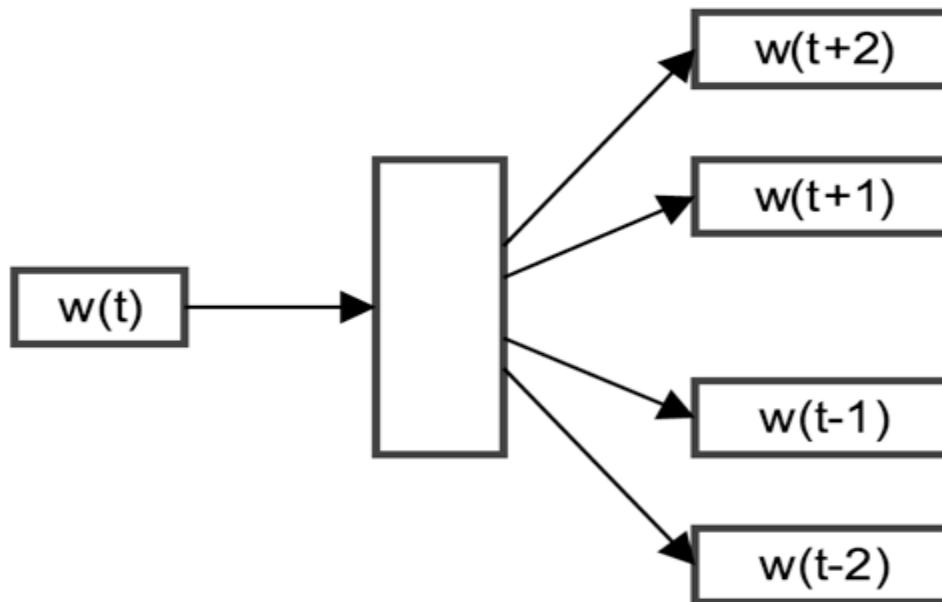
- Predict the surrounding words, based on the current word.



Mikolov et. al. 2013. Efficient Estimation of Word Representations in Vector Space.

# Skip-Gram Model

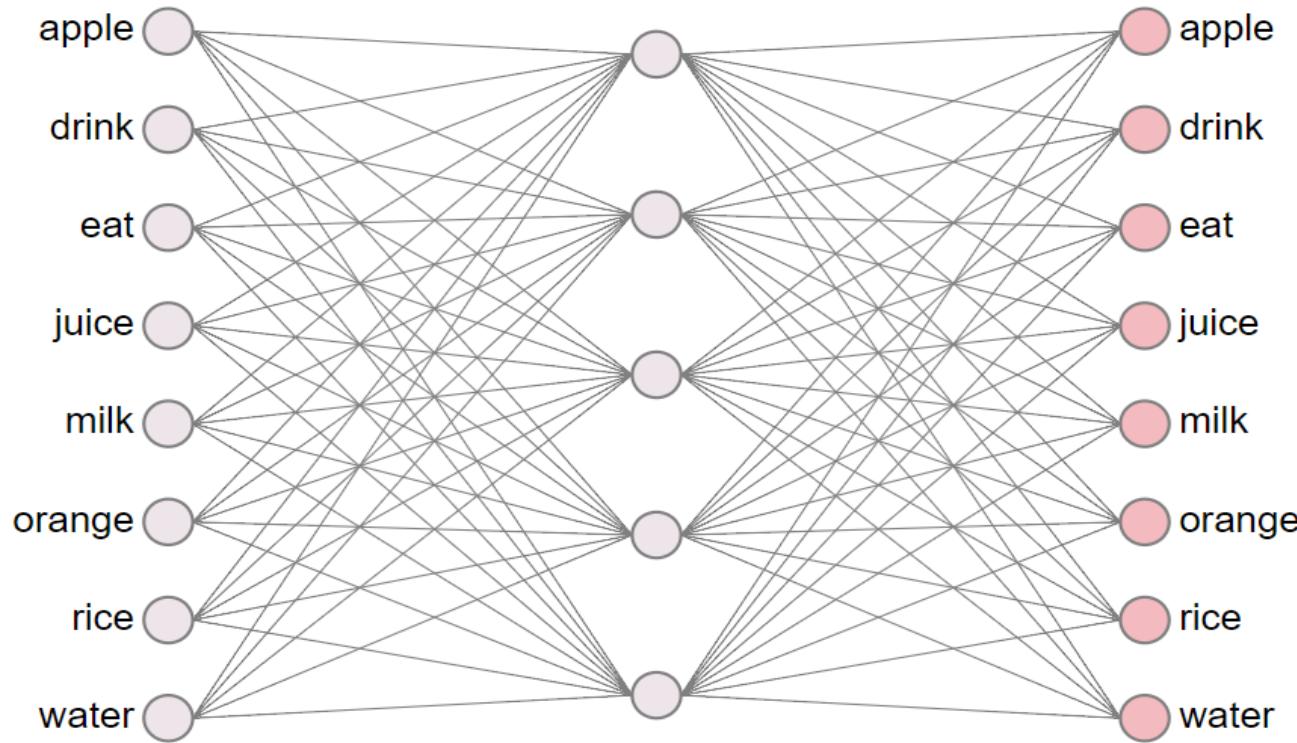
- Predict the surrounding words, based on the current word.



Mikolov et. al. 2013. Efficient Estimation of Word Representations in Vector Space.

# Word to Vector

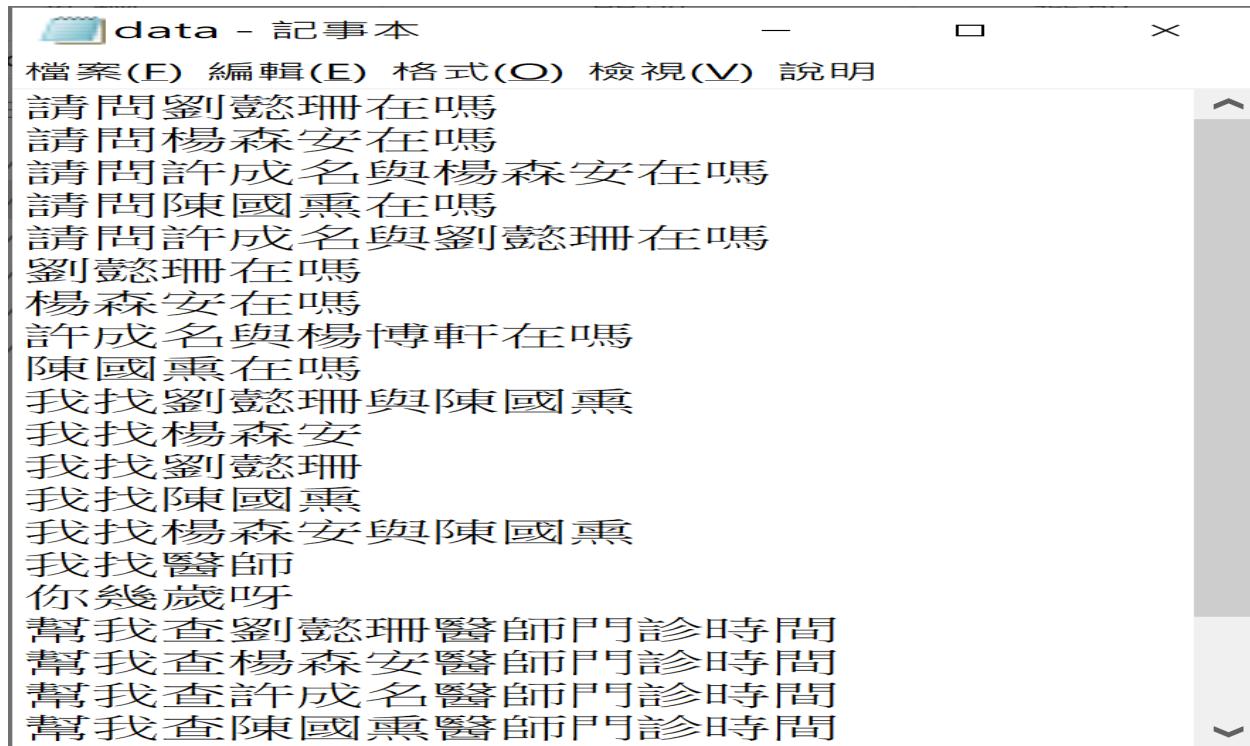
- WEVI



<https://ronxin.github.io/wevi/>

# 作業

- 判斷文件中醫師有幾位



# 作業

---

- n -gram model
  - CountVectorizer(ngram\_range=(2, 4 ))
- 解釋說明結果

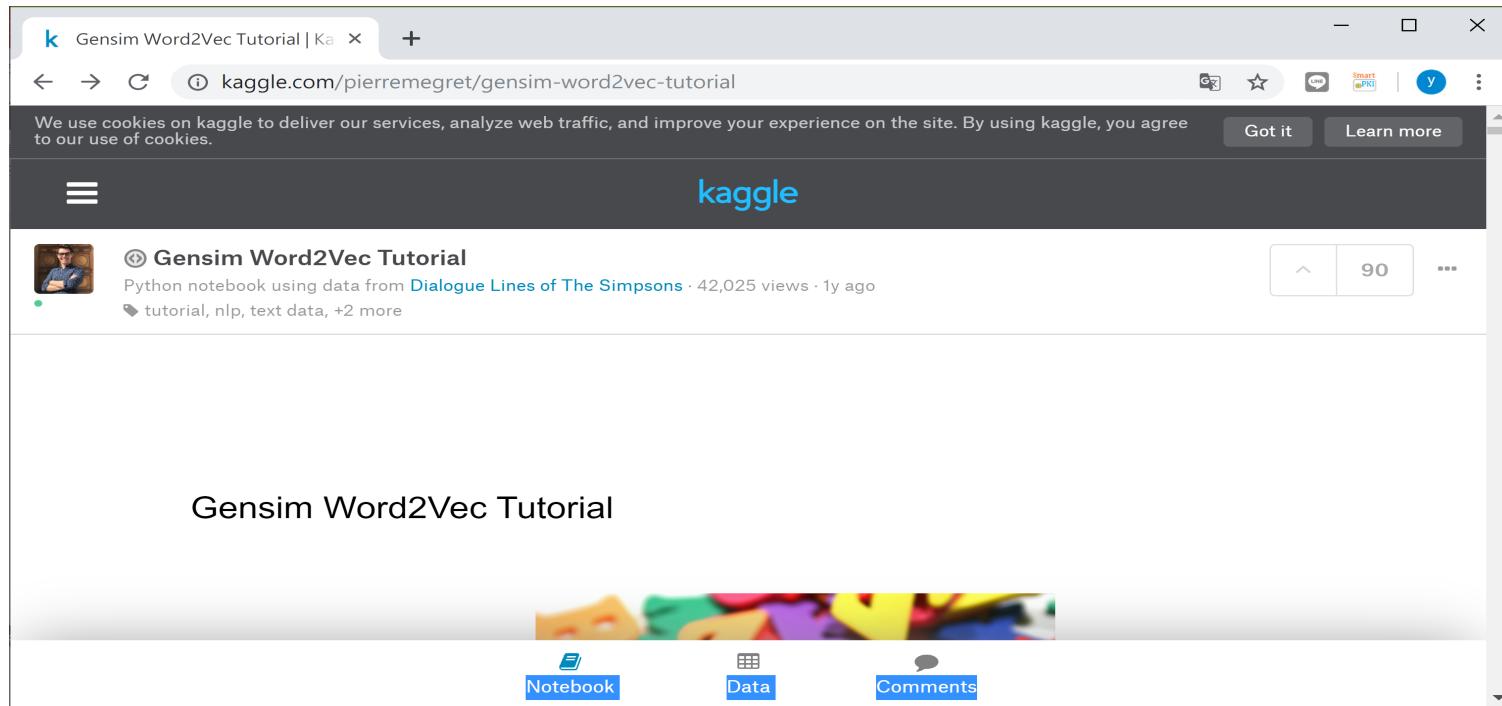
# 作業

---

- 使用 The 在計數與TF-IDF 詞袋差別
  - 詞彙總長度 vs 出現次數
  - 詞彙總長度 vs ?
- 為何取 TFIDF 要取 log 函數
- 說明 The 在三個文件中都有出
  - 文件一: 0.42983
  - 文件二: 0.61336
  - 文件三: 0.61336
- 計算文件一的TF-IDF

# 作業

- 實作練習



<https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial>