自然語言處理

第 1 章 NLP 簡介

講師：紀俊男

本章大綱

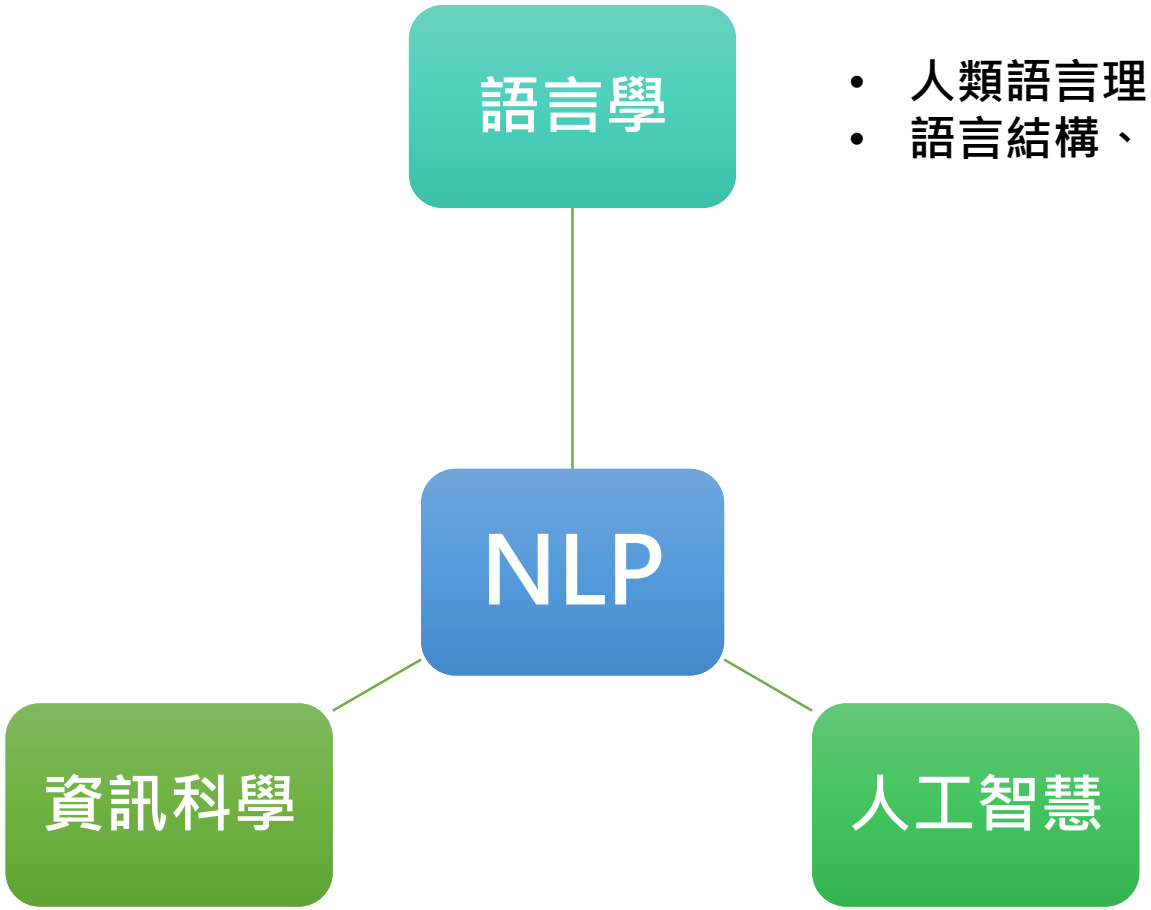
- 何謂「自然語言處理」
- NLP 的歷史與任務
- NLP 程式開發流程
- NLP 的應用與挑戰

何謂
「自然語言處理」

「自然語言處理」定義

- **「自然語言處理」**
  - Natural Language Processing (NLP)
- **「自然語言」**
  - 非由人為定義，自然形成的人類語言。
- **「自然語言處理」核心任務**
  - 理解：理解人類語言的含義和情感
  - 生成：產生自然且流暢的語言回應

與其他領域的關聯



- 人類語言理論、規則
- 語言結構、語法、語義

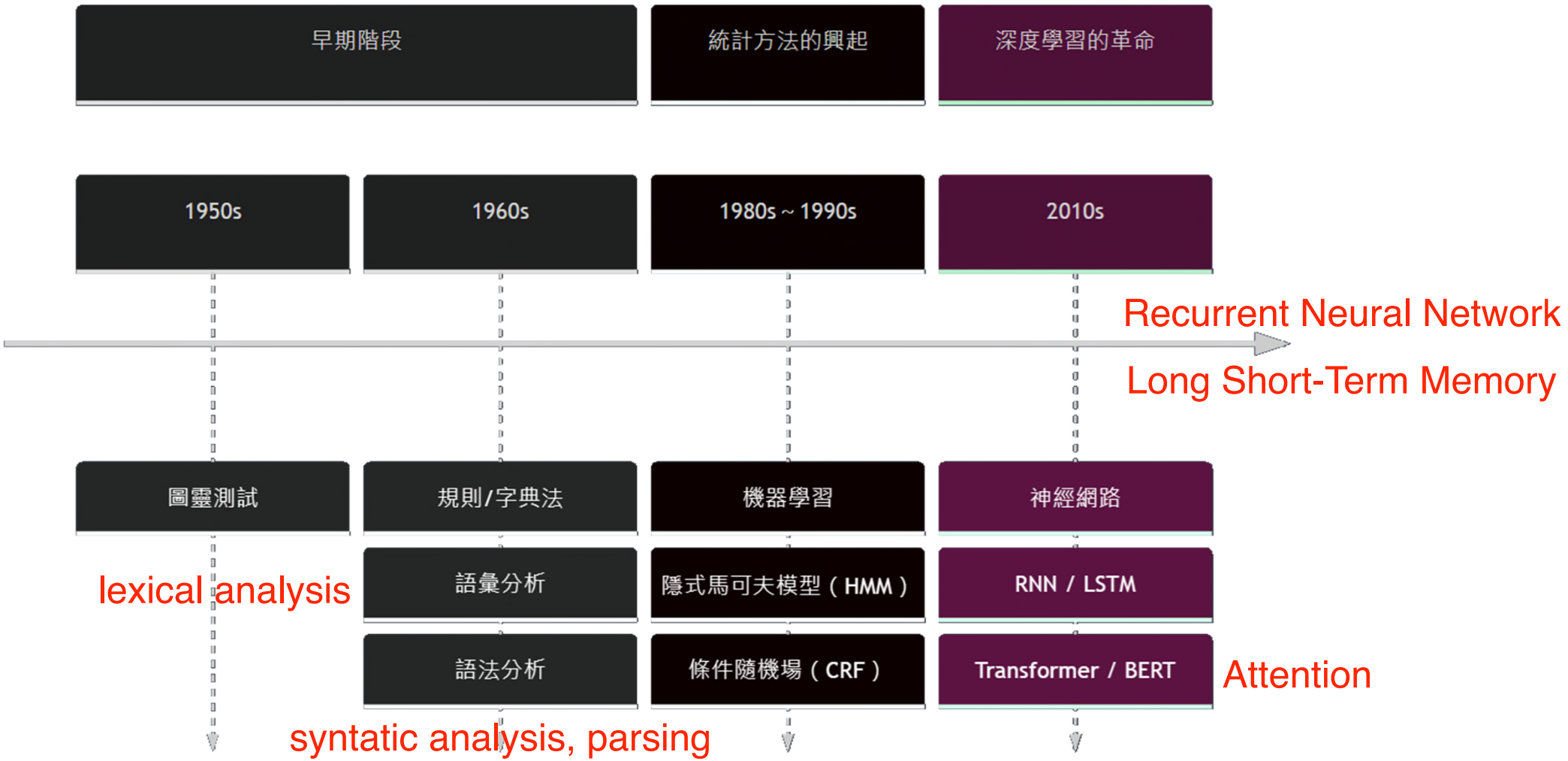
- 計算機演算法、工具
- 資料處理、純化、應用

- 機器學習、深度學習
- 知識圖譜

knowledge graph

NLP 的
歷史與任務

NLP 發展史


早期階段
統計方法的興起
深度學習的革命
1950s
1960s
1980s ~ 1990s
2010s
Recurrent Neural Network
Long Short-Term Memory
圖靈測試
規則/字典法
機器學習
神經網路
lexical analysis
語彙分析
隱式馬可夫模型（HMM）
RNN / LSTM
語法分析
條件隨機場（CRF）
Transformer / BERT
Attention
syntatic analysis, parsing

NLP 常見任務

- **語言分析（Language Analysis）**
  - 斷詞斷句（Tokenization）
    - 將文本分割成單詞或短語。
  - 詞性標註（Part-of-Speech Tagging） POS
    - 識別單詞的詞性（如名詞、動詞等）。
  - 命名實體識別（Named Entity Recognition） NER
    - 識別文本中的特定實體，如人名、地名、機構名等。
  - 語法分析（Parsing）
    - 分析句子的結構和語義關係。
- **情感分析（Sentiment Analysis）**
  - 分析文本中的情緒或觀點傾向，如正面、負面或中立。
- **機器翻譯（Machine Translation）**
  - 將一種語言的文本自動翻譯成另一種語言。
- **自動摘要（Automatic Summarization）**
  - 生成文本的簡短摘要或概述。
- **問答系統（Question Answering）**
  - 根據用戶的問題，生成答案。

e.g. ChatGPT

NLP 程式開發流程

NLP 開發步驟

資料前處理

- 資料收集
- 文本清洗
- 斷句斷詞
- 語彙正規化
- 停用詞去除

文本表示

- 詞袋模型
- TF-IDF
- 詞嵌入
- 文本向量化

模型訓練

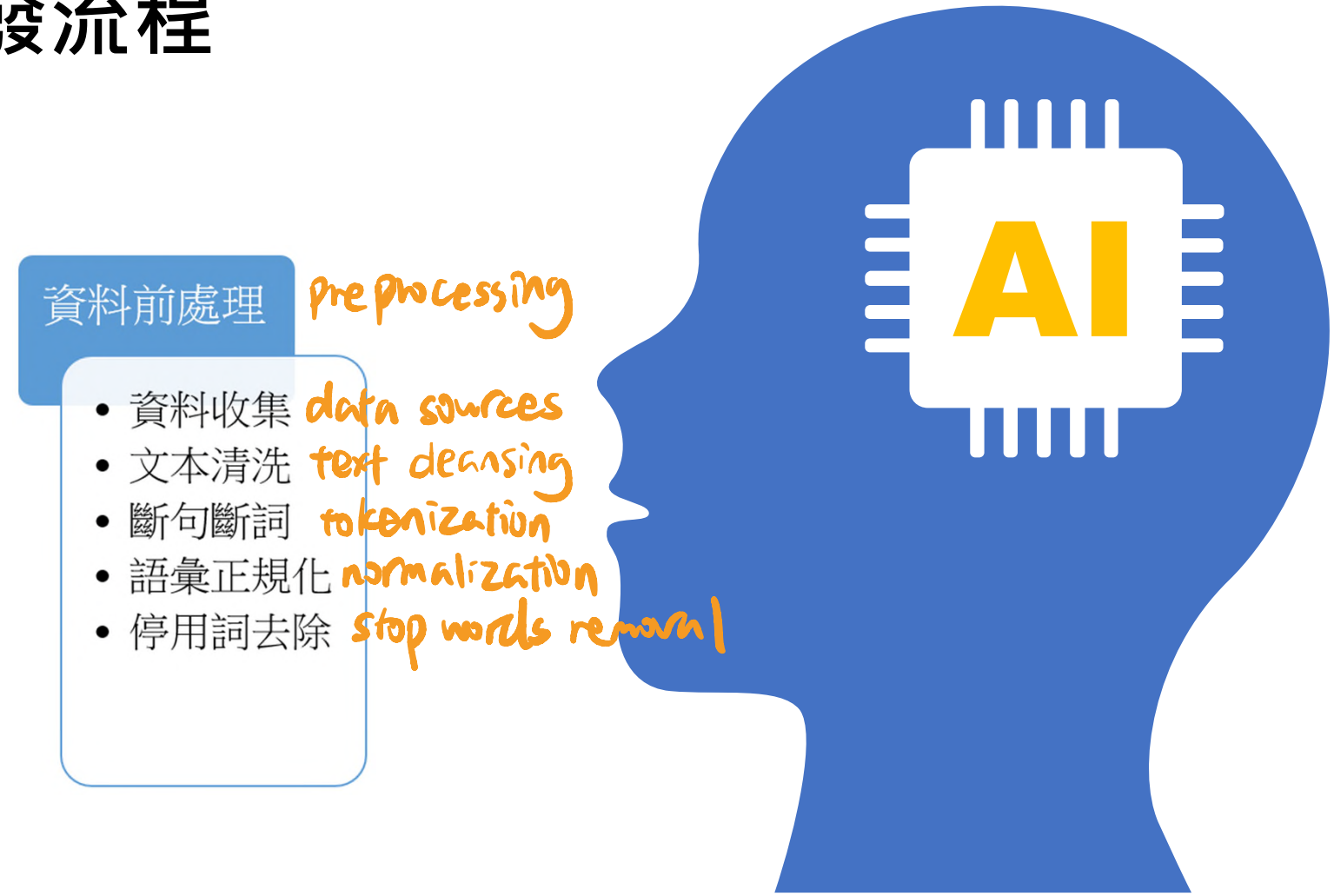
- 模型選擇
- 模型訓練
- 模型優化

評估與部署

- 模型評估
- 參數調整
- 模型部署

NLP 程式開發流程

- **資料前處理**
- 文本表示
- 模型訓練與優化
- 模型評估與部署



資料收集（Data Sources）

- 語料庫（Corpus）
  - 定義
    - 包含詞性、分類...各種標註，用於 NLP 研究的文本資料庫
  - 類型
    - 平行語料庫：多語言對照文本
    - 專業語料庫：特定領域專有名詞
    - 動態語料庫：如「流行語」語料庫
  - 應用
    - 機器翻譯
    - 名稱實體辨認（NER）~~name~~ named entity recognition
    - 語意分析
- 網路爬蟲
  - Excel
    - 優：簡單易用
    - 劣：網頁版型太複雜會爬不到
  - Python Pandas 套件
    - 優：網頁內容大多爬得到
    - 劣：表格以外資料有時不精確
  - Python 爬蟲專用套件（如：BS4）beautiful soup
    - 優：任何資料都爬得到
    - 劣：需學習，且自己做資料清洗

文本清洗（Text Cleansing）

- 包含去除下列字符：

“　　”	，。、：	@ # $ %
空白	標點符號	特殊符號
1 2 3 4	\n \t \r	😀 XD (ノ ̄皿 ̄)ノ
數字	不可見字元	顏文字

斷句斷詞（Tokenization）

- **斷句範例（Segmentation）**
  - 正確示範：我今天去了超市，買了很多水果。除此之外，還有一些零食。
  - 使用空白：我今天去了超市 買了很多水果 除此之外 還有一些零食
  - 相同符號：我今天去了超市...買了很多水果...除此之外...還有一些零食
  - 沒有符號：我今天去了超市買了很多水果除此之外還有一些零食
  - 顏文字：我今天去了超市😁買了很多水果❤️除此之外👈還有一些零食😎

- **斷詞範例（Tokenization）**
  - 原始句子：下雨天留客天留我不留
  - 意思一：下雨天/留客/天留/我不留 → 逐客
  - 意思二：下雨天/留客天/留我不/留 → 留客

Jieba

語彙正規化（Normalization）

- 將詞彙轉化為基本形式，減少詞彙的多樣性（主要為西歐語文）

You→you←YOU

統一大小寫

I’ m→I am

縮寫還原

running→run

詞幹提取（Stemming）

ran→run

詞形還原（Lemmatization）

TV -> television, 北車 -> 台北車站

停用詞去除（Stop Words Removal）

- 移除常見、但對分析沒有貢獻的詞（如：的、是、呢...）

原始逐字稿

這個呢，其實呴，就是一件簡單的糾紛啦。
如果呴，大家各退一步，就沒有那麼複雜啦。

去除停用詞

這個—，其實— ， ——一件簡單—糾紛— 。
如果— ，大家—退一步， —沒有——複雜— 。

該同時套用所有前處理步驟嗎？

不用！！！

不同輸入文本、不同應用場合 → 套用不同前處理步驟

NLP 程式開發流程

- 資料前處理
- **文本表示**
- 模型訓練與優化
- 模型評估與部署

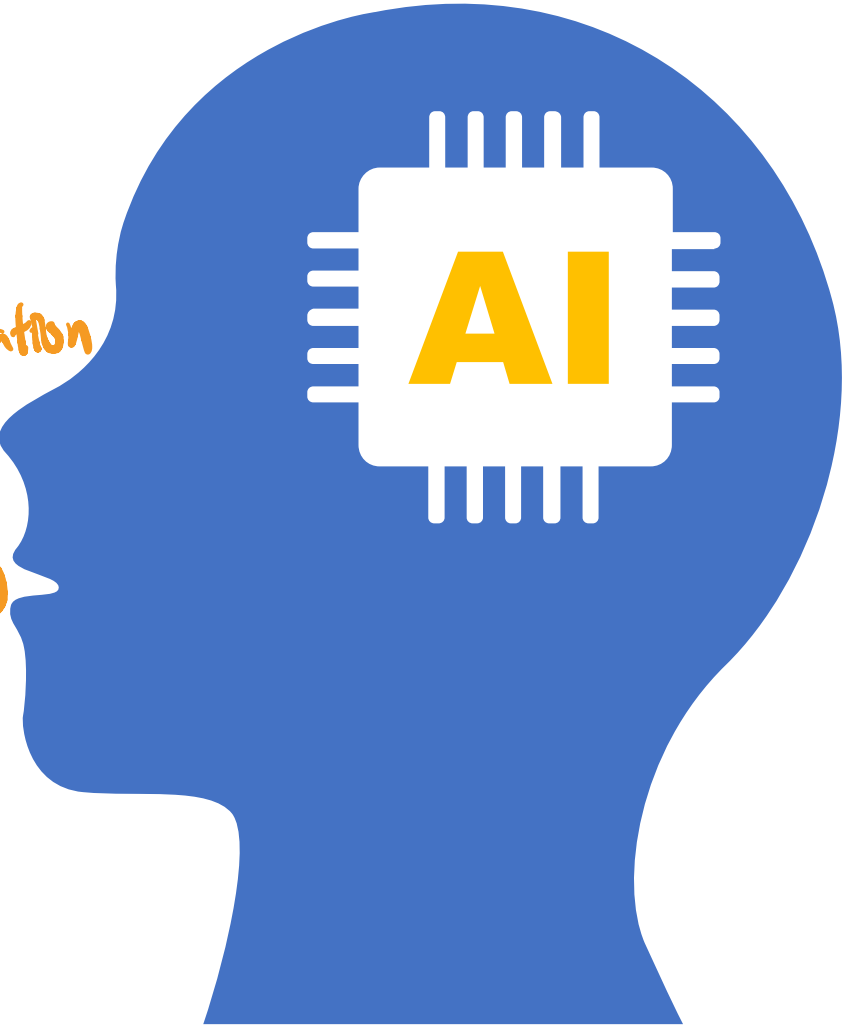
文本表示

- 詞袋模型
- TF-IDF
- 詞嵌入
- 文本向量化

text representation

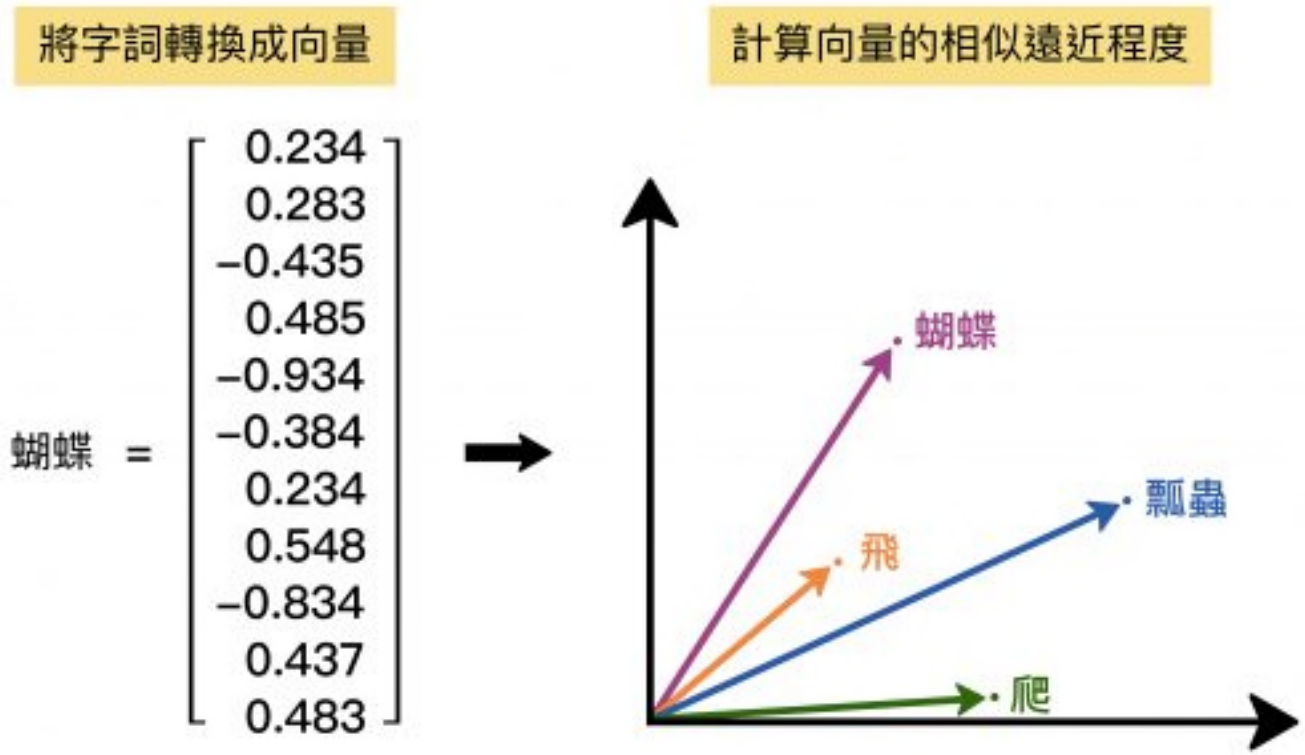
Bag of Words

Word Embedding



何謂「文本表示」（Text Representation）

- 將「自然語言」，轉換成電腦能夠處理的「資料格式」



常見的「文本表示演算法」

- (1) 獨熱編碼法（One-Hot Encoding）

太耗
記憶體

alignment

['', '', 'i', 'love', 'jogging']

```
[[0, 0, 2, 3, 4]
 [5, 2, 3, 7, 1]]
```

['and', 'i', 'love', 'reading', 'too']

0	[1, 0, 0, 0, 0, 0, 0, 0]
1	[0, 1, 0, 0, 0, 0, 0, 0]
2	[0, 0, 1, 0, 0, 0, 0, 0]
3	[0, 0, 0, 1, 0, 0, 0, 0]
4	[0, 0, 0, 0, 1, 0, 0, 0]
5	[0, 0, 0, 0, 0, 1, 0, 0]
6	[0, 0, 0, 0, 0, 0, 1, 0]
7	[0, 0, 0, 0, 0, 0, 0, 1]

```
[[[1., 0., 0., 0., 0., 0., 0., 0.,]
  [1., 0., 0., 0., 0., 0., 0., 0.,]
  [0., 0., 1., 0., 0., 0., 0., 0.,]
  [0., 0., 0., 1., 0., 0., 0., 0.,]
  [0., 0., 0., 0., 1., 0., 0., 0.,]]

 [[0., 0., 0., 0., 0., 1., 0., 0.,]
  [0., 0., 1., 0., 0., 0., 0., 0.,]
  [0., 0., 0., 1., 0., 0., 0., 0.,]
  [0., 0., 0., 0., 0., 0., 0., 1.,]
  [0., 1., 0., 0., 0., 0., 0., 0.,]]]
```

常見的「文本表示演算法」

- (2) 多熱編碼法（Multi-Hot Encoding）

仍然很耗記憶體

['i love jogging!'， 'and i love reading, too!']

1	"too"
2	"i"
3	"love"
4	"jogging"
5	"and"
6	"you"
7	"reading"

0	1	2	3	4	5	6	7
		"I"	"love"	"jogging"			
0	0	1	1	1	0	0	0

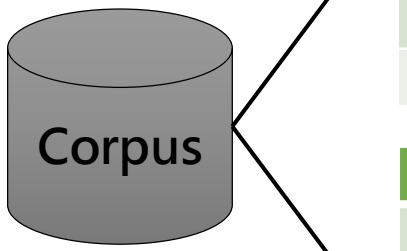
→ [0. 0. 1. 1. 1. 0. 0. 0.]

0	1	2	3	4	5	6	7
	"too"	"I"	"love"		"and"		"reading"
0	1	1	1	0	1	0	1

→ [0. 1. 1. 1. 0. 1. 0. 1.]

常見的「文本表示演算法」

- (3) N-gram 表示法（1950s）



1-gram Dictionary

1	“brown”
2	“dog”
3	“fox”
4	“jump”
5	“lazy”
6	“over”
7	“the”

2-gram Dictionary

1	“brown fox”
2	“fox jump”
3	“jump over”
4	“lazy dog”
5	“over the”
6	“the brown”
7	“the lazy”

The fox jump over a lazy dog

1-gram Representation

the	fox	jump	over	a	lazy	dog
7	3	4	6	0	5	2

0	1	2	3	4	5	6	7
1	0	1	1	1	1	1	1

2-gram Representation

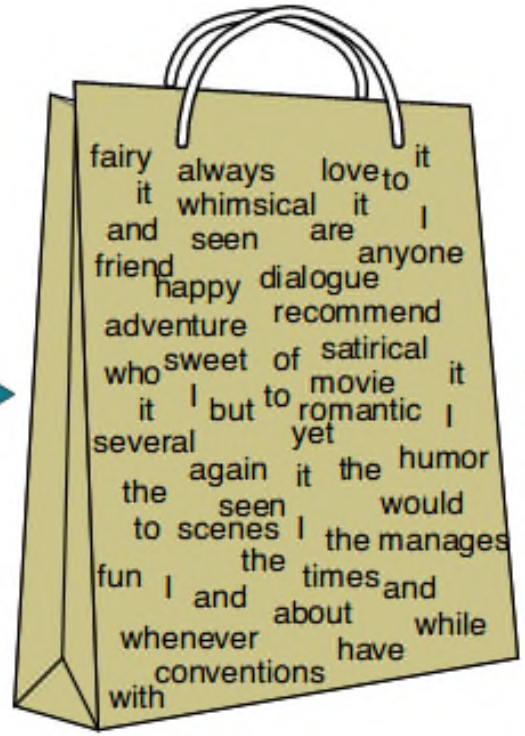
the fox	fox jump	jump over	over a	a lazy	lazy dog
0	2	3	0	0	4

0	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0

常見的「文本表示演算法」

- (4) 詞袋模型（Bag of Words, BoW）（1960s）

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

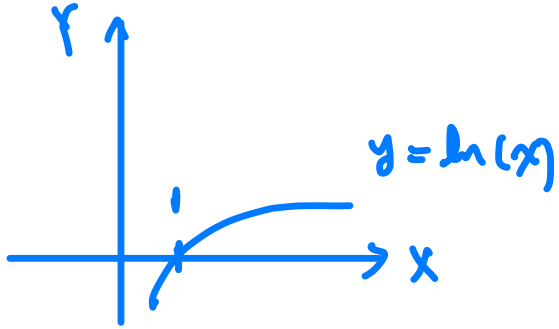


it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

stop words

- ✗ 考慮詞序（Order）
- ✓ 考慮詞頻（Frequency）

常見的「文本表示演算法」



- (5) TF-IDF（1970s）
  - 何謂 TF-IDF？
    - Term Frequency-Inverse Document Frequency
    - 詞頻-逆文檔詞頻

  - TF-IDF 公式
    - $TF \times IDF = \frac{\text{某文檔特定詞彙出現次數} W_T}{\text{某文檔總字數} W} \times \ln\left(\frac{\text{總文件數} N}{\text{包含特定詞彙的文件數} N_T}\right)$
      - IDF 取對數的原因：讓 the, a, … 等高詞頻停用詞之 IDF 值不至於太大。亦可使用 log()。

  - 使用 TF-IDF 的原因
    - 減少像 “the” 、 “a” 這樣常見的停用詞，在文本分析中的影響力。 ln(1) = 0
    - 增強那些在少量文檔中頻繁出現的專有名詞之重要性，使其在文本處理中更加突出。
      （重要名詞 = 集中 & 稀有）

常見的「文本表示演算法」

- (5) TF-IDF 範例

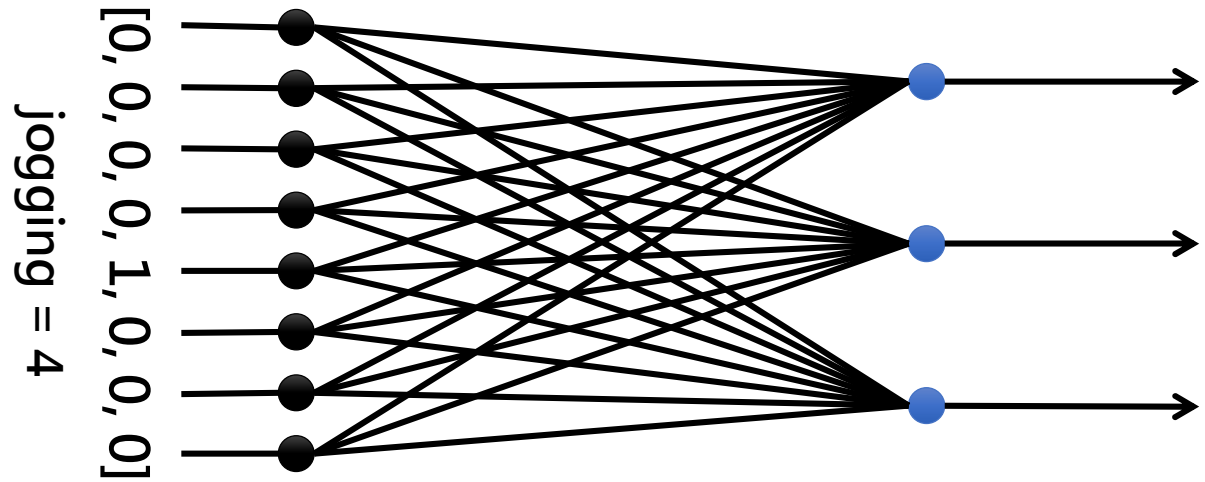
Documents	Text	Total number of words in a document
A	Jupiter is the largest planet	5
B	Mars is the fourth planet from the sun	8

Words	TF (for A)	TF (for B)	IDF	TFIDF (A)	TFIDF (B)
Jupiter	1/5	0	ln(2/1) = 0.69	0.138	0
Is	1/5	1/8	ln(2/2) = 0	0	0
The	1/5	2/8	ln(2/2) = 0	0	0
largest	1/5	0	ln(2/1) = 0.69	0.138	0
Planet	1/5	1/8	ln(2/2) = 0	0.138 0	0
Mars	0	1/8	ln(2/1) = 0.69	0	0.086
Fourth	0	1/8	ln(2/1) = 0.69	0	0.086
From	0	1/8	ln(2/1) = 0.69	0	0.086
Sun	0	1/8	ln(2/1) = 0.69	0	0.086

常見的「文本表示演算法」

- (6) 詞向量嵌入法（Word Embedding）（2000s）

原理：利用神經網路，將詞編碼降維後，得到一個壓縮過後的「詞向量」



以後，" jogging"

=

[-0.00180965 0.00570974 -0.01133073]

詞編碼

[0, 0, 0, 0, 1, 0, 0, 0] ⊙

權重 W
（可自行習得）

[[0.01023025 -0.01604468 0.04070808]
[0.01023025 -0.01604468 0.04070808]
[0.01023025 -0.01604468 0.04070808]
[-0.00180965 0.00570974 -0.01133073]
[0.01023025 -0.01604468 0.04070808]
[0.01023025 -0.01604468 0.04070808]
[0.01023025 -0.01604468 0.04070808]
[0.01023025 -0.01604468 0.04070808]]

= [-0.00180965 0.00570974 -0.01133073]

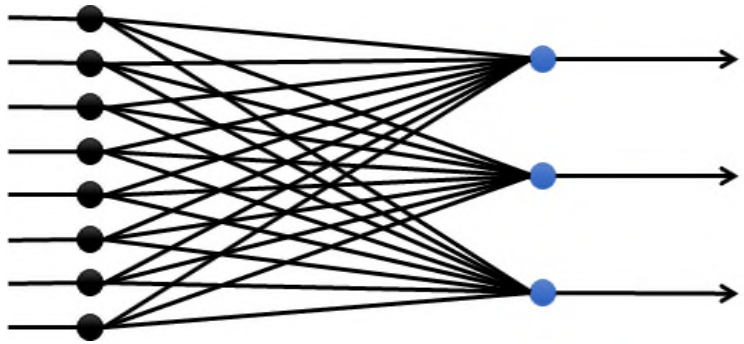
詞向量

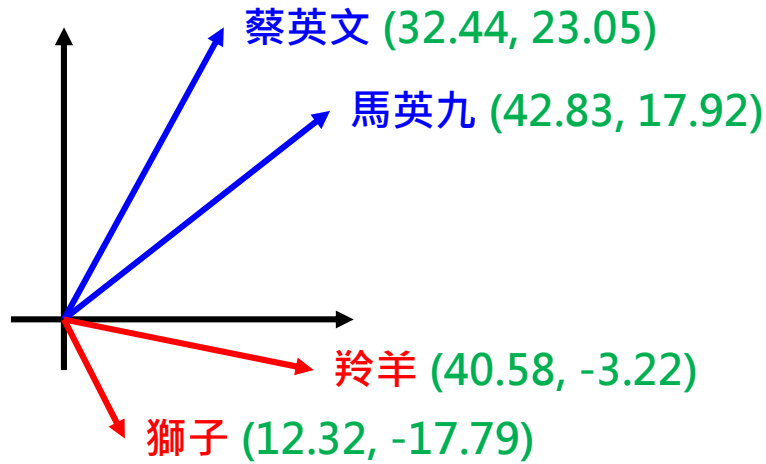
1x8　　8x3　　1x3

常見的「文本表示演算法」

- 詞向量嵌入法的優點

詞向量的歐幾里得距離接近 = 詞的語意接近





小節整理

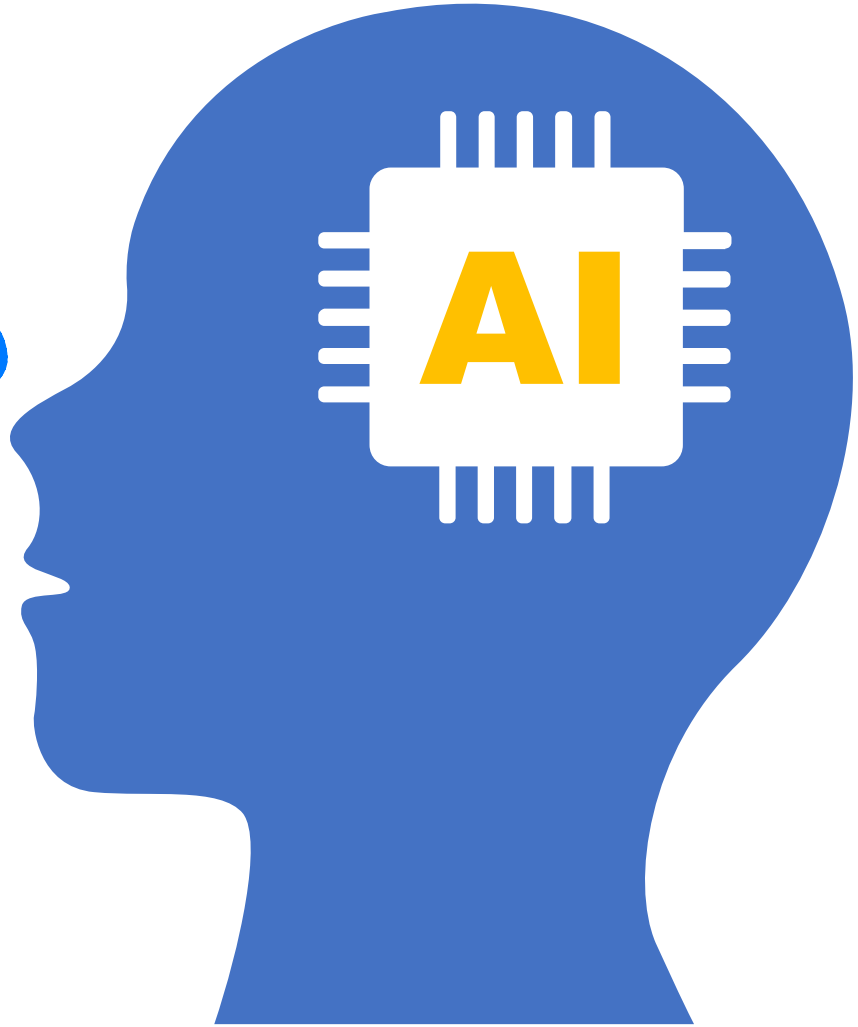
- **文本表示（Text Representation）**
  - 定義：將「自然語言」，轉換成電腦能夠處理的「資料格式」
- **獨熱編碼法（One-Hot Encoding）**
  - 優：結構簡單，實作容易。
  - 劣：詞彙量大時維度過高，無法捕捉詞之間的語義相似性。
- **多熱編碼法（Multi-Hot Encoding）**
  - 優：相較於獨熱編碼，能在一定程度上降低維度。
  - 劣：依然有維度相對較高的問題，不考慮詞序和語境。
- **N-gram 表示法**
  - 優：N≥2 時，能包含一定的詞序資訊。
  - 劣：N 越大，特徵空間會擴大。無法捕捉長範圍的詞序。
- **詞袋模型（Bag of Words, BoW）**
  - 優：將「詞頻」考慮進去。
  - 劣：對高詞頻的停止詞效果不佳。
- **TF-IDF**
  - 優：有效降低高詞頻停止詞的影響力。
  - 劣：仍不考慮詞序。計算 IDF 需要整個語料庫。
- **詞向量嵌入法（Word Embedding）**
  - 優：維度低，能捕捉語義相似性。
  - 劣：需要大量資料來訓練正確權重。對罕見詞/新詞抗性不佳。

NLP 程式開發流程

- 資料前處理
- 文本表示
- **模型訓練與優化**
- 模型評估與部署

模型訓練 Model Training

- 模型選擇
- 模型訓練
- 模型優化



NLP 常見的模型

- **規則/辭典式模型** Rule/Dictionary Based
  - 最早的 NLP 模型。用規則或字典比對法來實作。
  - 不需訓練，但無法抵抗新詞彙。
  - 尋找特定領域的專有名詞很好用。
- **單純貝氏分類器 (Naive Bayes Classifier)**
  - 利用貝氏定理來做分類的模型。
  - 適用於特徵間獨立，執行如「垃圾郵件」這類簡單粗暴的分類任務時使用。
- **支援向量機 (Support Vector Machines, SVM)**
  - 適用於高維空間執行分類任務，尋找精準之決定邊界時使用。
- **決策樹 (Decision Trees)**
  - 適用於分類與迴歸任務。
  - 想知道模型是「怎麼做出來」時使用。 interpretability

機器學習模型

- **長短期記憶網絡 (Long Short-Term Memory, LSTM)**
  - 一種特殊的遞歸神經網絡 (RNN)，能夠學習長距離相依資訊。
  - 1990s~2000s 年代時，NLP 的主力模型。
- **Transformer 模型**
  - 基於「注意力機制」的模型，能夠良好處理與理解文字資料。
  - 常見於機器翻譯、文本摘要和問答系統。
- **BERT (Bidirectional Encoder Representations from Transformers)**
  - 一種使用 Transformer 架構的模型，能雙向理解上下文。
  - 目前 NLP 主力模型，適用於幾乎所有 NLP 任務。
- **GPT (Generative Pre-trained Transformer)**
  - 一種基於 Transformer 架構的自監督學習模型。
  - 能夠生成語氣連貫的文本序列。

神經網路模型

模型優化的目的

generalization

性能提升

改進模型以獲得更高的準確率、更低的損失值。

泛化能力

確保模型對未見數據有良好的預測能力，避免過擬合。

效率增強

使模型更快地訓練和預測，降低計算成本。

模型優化的方法

1 **超參數調整**

Grid Search　　Bayesian Optimization

網格搜索、隨機搜索、貝葉斯優化。

Random Search

2 **模型簡化**

減少模型複雜度，移除不必要的層或神經元。

3 **正則化技術**

L1/L2 正則化、Dropout。

penalize model complexity

4 **學習率調整**

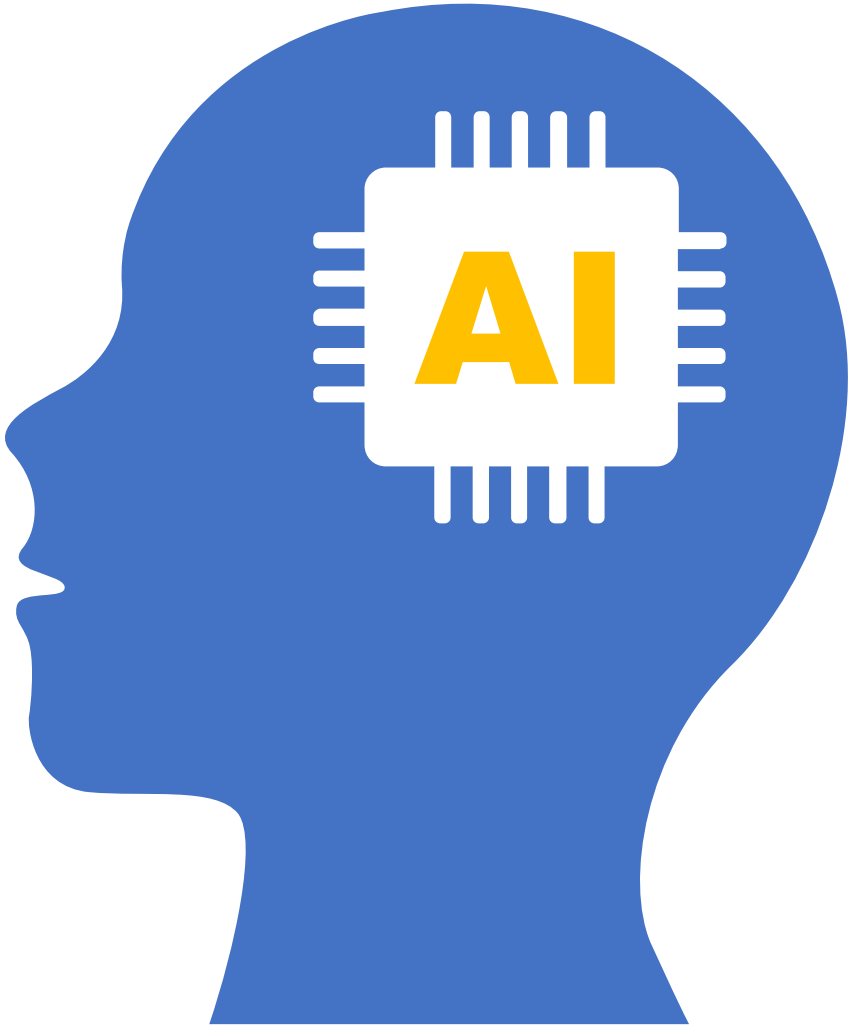
學習率衰減、自適應學習率算法。

NLP 程式開發流程

- 資料前處理
- 文本表示
- 模型訓練與優化
- **模型評估與部署**

評估與部署

- 模型評估
- 參數調整
- 模型部署



模型評估

效能指標

- 準度
- 廣度
- 精度
- F1-Score

交叉分析

- 交換訓練集
- 避免過擬合

錯誤分析

- 分類錯誤
- 效能瓶頸

模型部署

1

模型封裝

將訓練好的模型封裝成可重複使用的服務或 API，提高模型的可用性和擴展性。

2

生產環境部署

將模型部署到生產環境中，進行實際應用，確保模型能夠正確運行並產生預期效果。

3

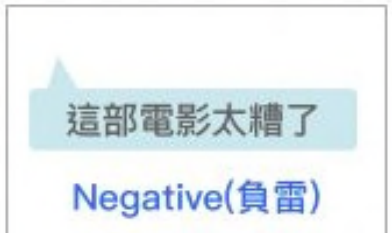
監控與維護

定期監控模型在生產環境中的性能，並進行更新和維護，以確保模型一直處於最佳狀態。

NLP 的
應用與挑戰

NLP 的應用

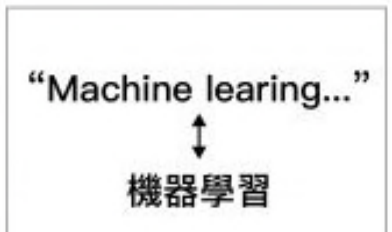
偵測詐騙郵件



情緒分析

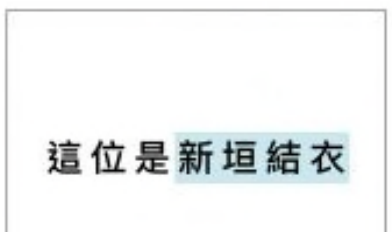
搜尋建議更正

詞類標示

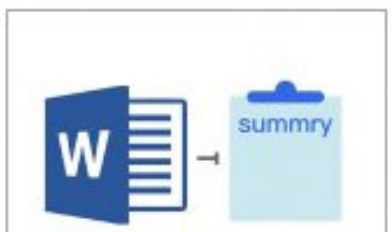


機器翻譯

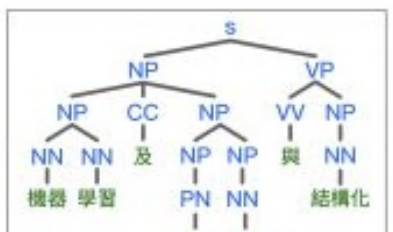
語音辨識



人名辨識擷取



摘要文本大綱



句法分析

NLP 的挑戰

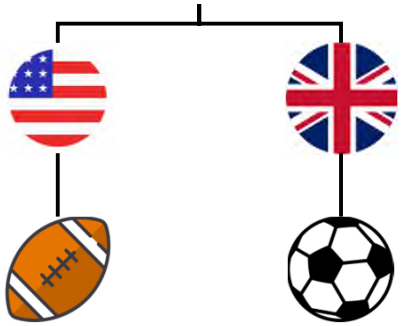
多樣性

歧義性

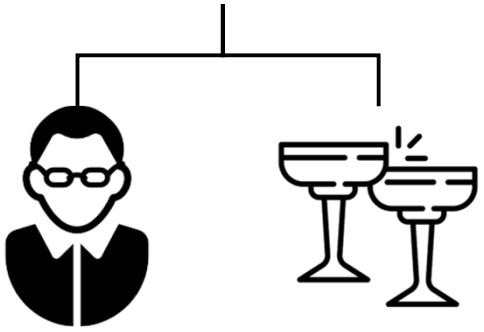
相依性

變動性

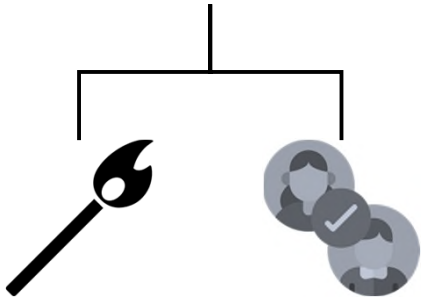
“football”



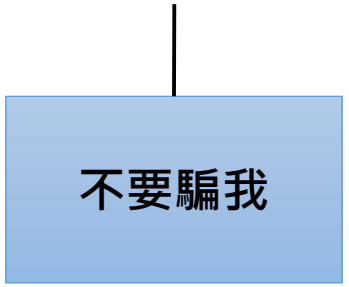
“with glasses”



“have a match”



“不要旋轉我”



本章總結

- **自然語言簡介**
  - 定義：非人為定義的人類語言。
  - 任務：理解&生成
- **NLP 三大時期**
  - 古典期：規則/字典式模型
  - 統計模型期：機器學習模型
  - 神經網路期：深度學習模型
- **NLP 開發流程**
  - 前處理：收集、清洗、斷詞、正規化。
  - 表示法：獨熱/多熱編碼、N-gram、詞袋、TF-IDF、詞嵌入
  - 模型訓練與優化
  - 模型評估與部署
- **NLP 的挑戰**
  - 多樣性、歧義性、相依性、變動性