

# SCU LawTech Class #01

主講者: 何彥南 🐵

# Content

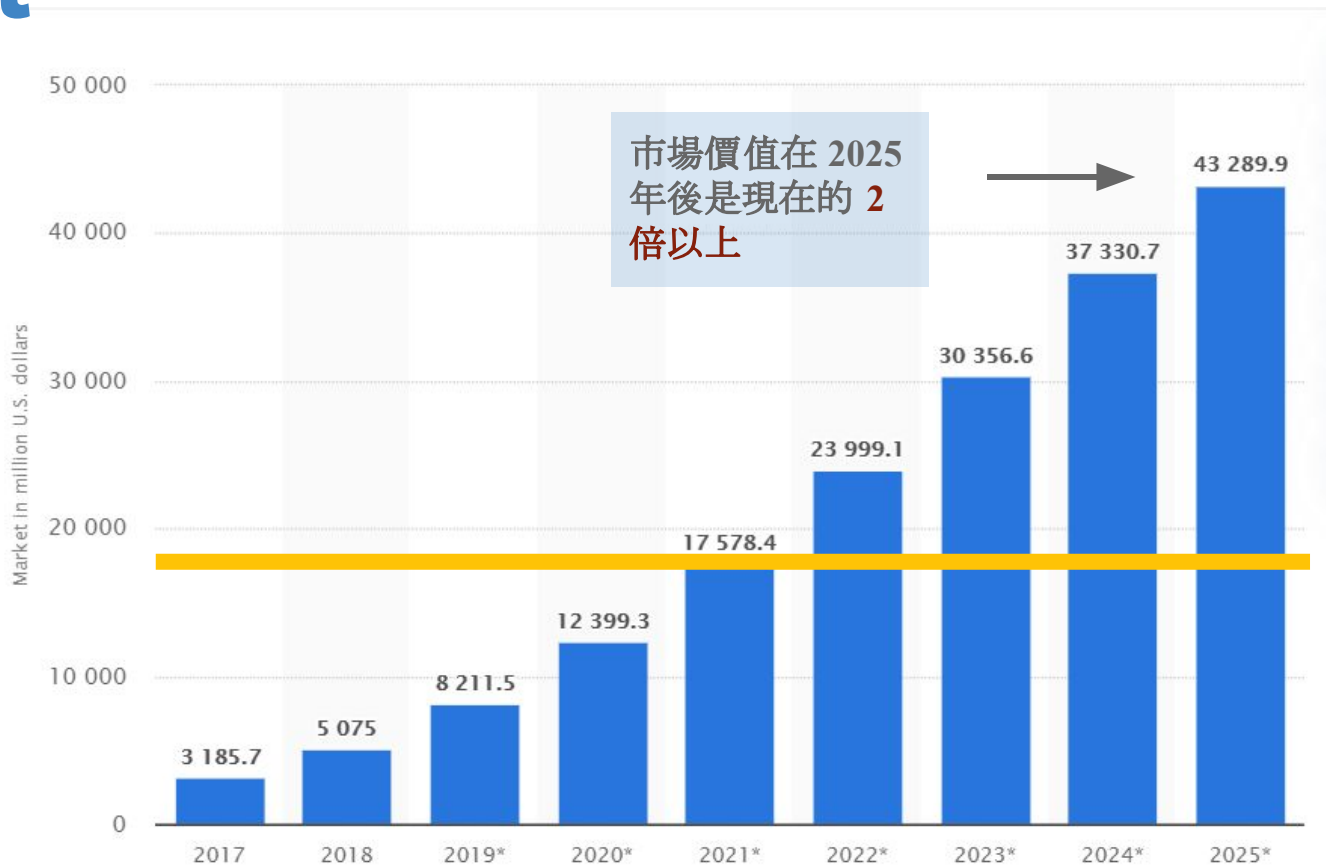
---

- ◆ 自然語言處理 (Natural Language Processing)
- ◆ 資料檢索 (Information Retrieval)
- ◆ 文字前處理 (Text Preprocessing)
- ◆ 斷詞 (Word Segmentation)
- ◆ 詞嵌入 (Word Embedding)

# 自然語言處理

Natural Language Processing

# Market



# Value

---

## 1. 為運營上的效率以及成本的降低

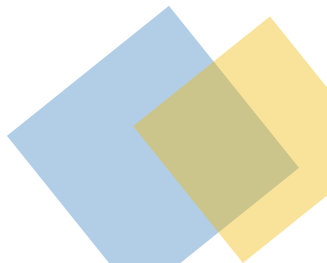
- [AI 篩選履歷](#) → 降低人力篩選成本、提升招募精準度、招募流程縮短
- [airbnb 客服回復建議](#) → 減少手動輸入時間、增加回覆的精準度

## 2. 為顧客旅程以及體驗的優化

- [uber 使用用戶回饋優化地圖](#) → 提升顧客體驗像。

## 3. 各個不同產業透過 NLP 所驅動的商業模式

- [opview](#) → 社群分析
- [awo](#) → SEO 診斷與數據分析



# what is NLP ?

---

造句題目：難過

我家門前有條水溝很難過。

老師更難過

自然語言處理是一種透過複雜的數學模型及演算法來讓機器去認知、理解並運用我們的語言的技術。



# Word

---

中文



努力才能成功 → 努力 才 能 成功

他的領導才能很突出 → 他 的 領導 才能 很 突出

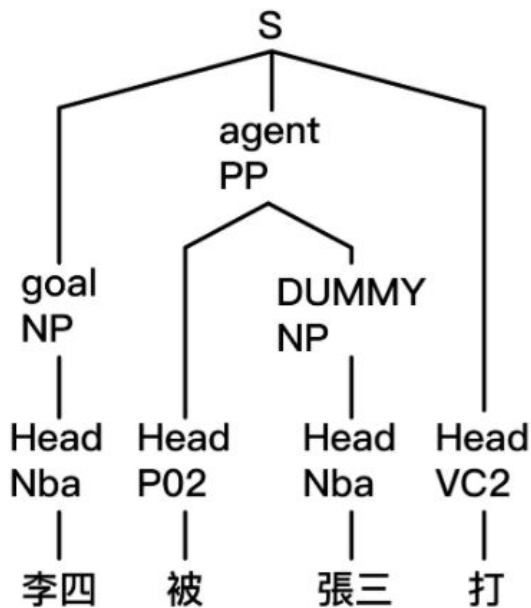
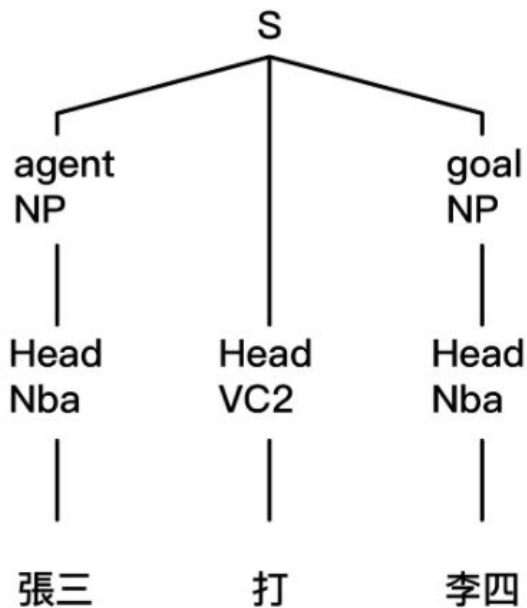
英文



空白



# Sentence



雖然透過建立句子的結構樹，可以讓電腦去理解詞之間的關係。但是無法有效解決指代消解的問題。

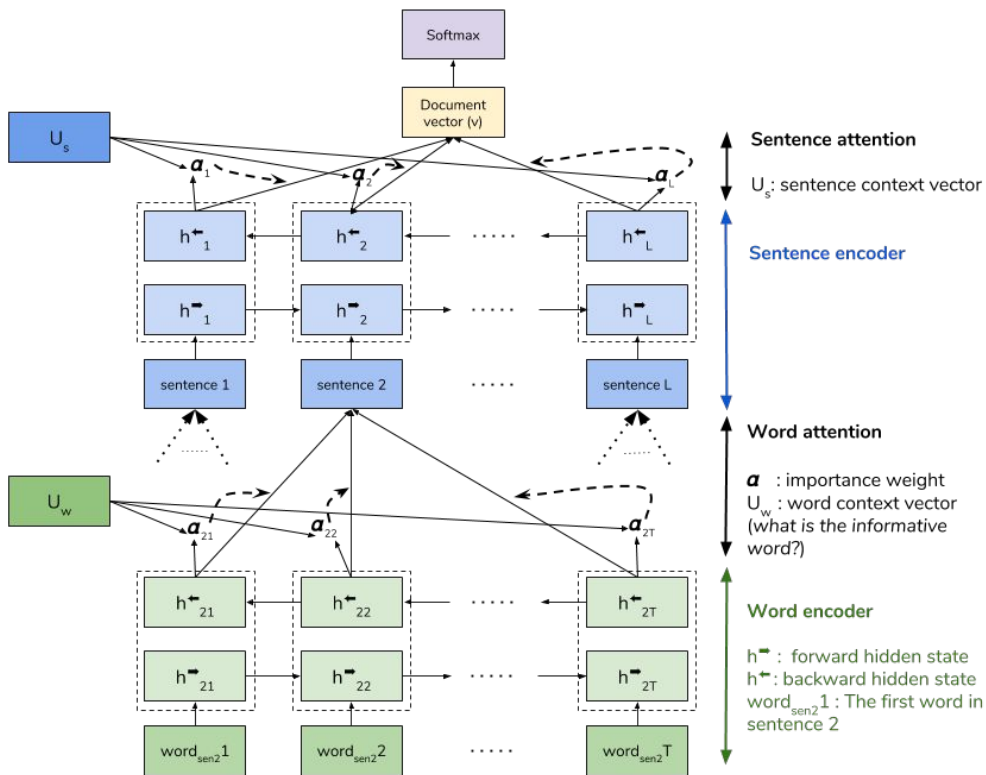


張三打李四，他很痛



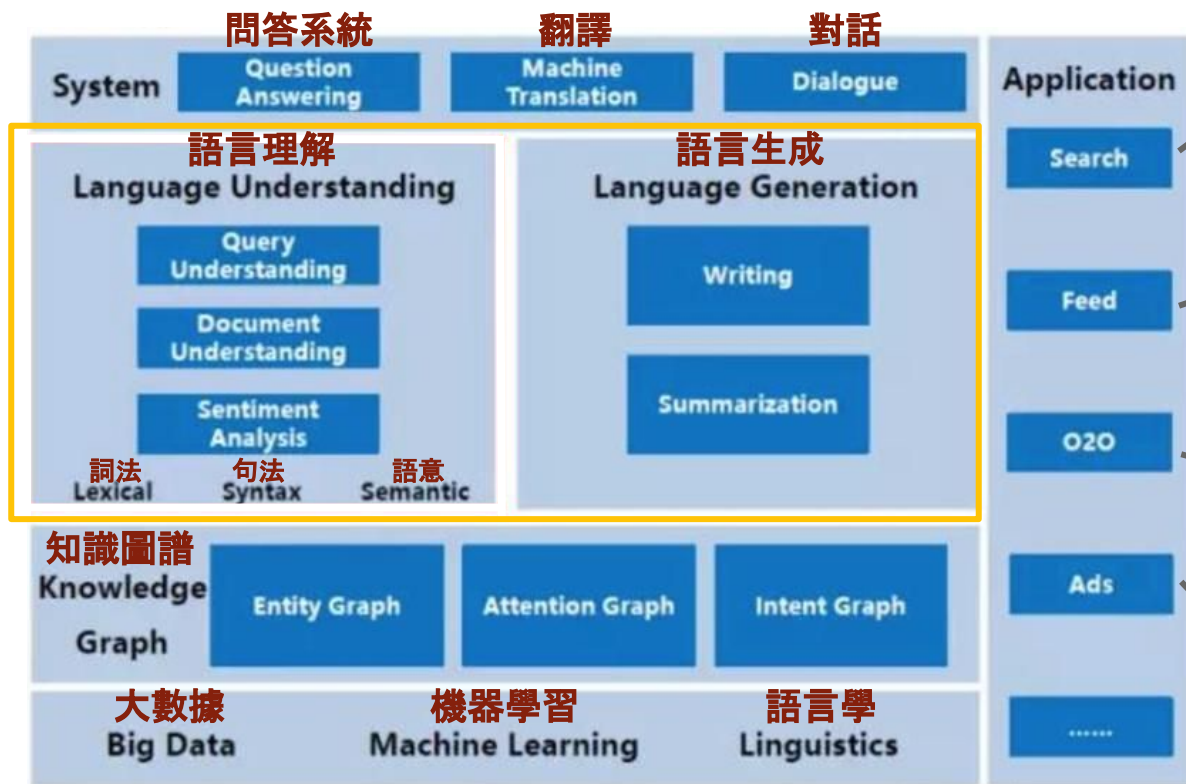
# Document

HAN 模型透過深度學習的**注意力機制**去學習整篇文章中不同層面(句子、詞)與目標之間的關係。此關係是隨著目標不同而改變。



# Levels

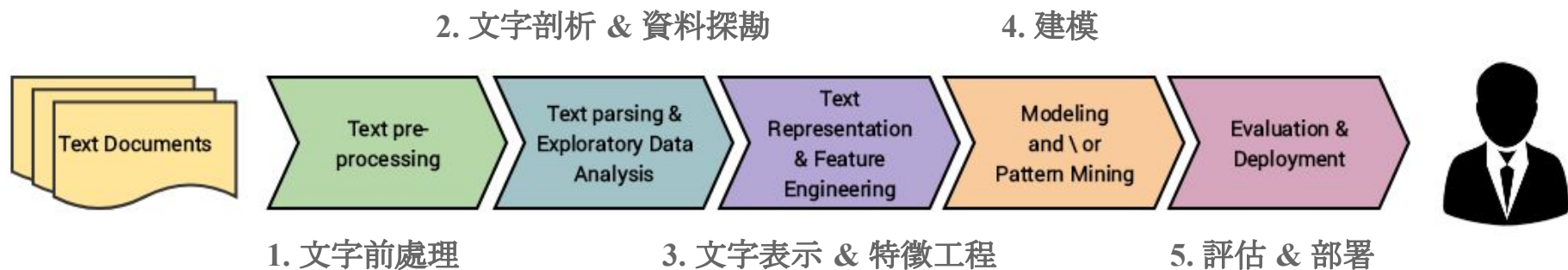
NLP



Online to Offline

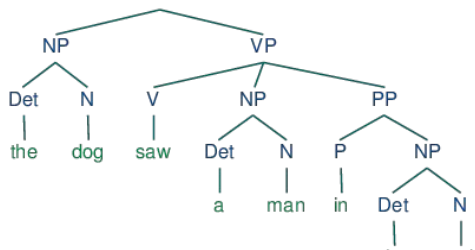


# NLP Pipeline

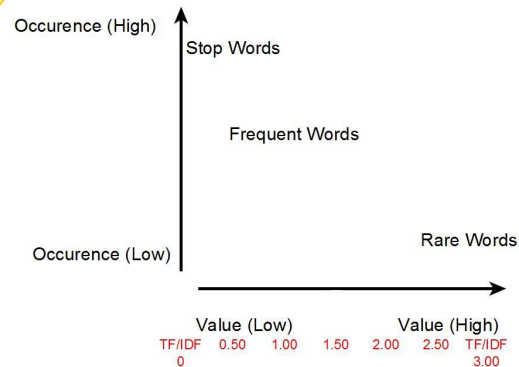


# Change of Solutions

## 1. 規則方法



## 2. 統計方法 & 機器學習

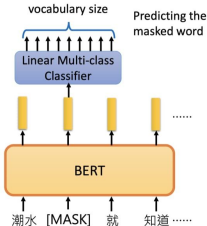


## 3. 深度學習

### 預訓練任務 1：克漏字填空

#### Training of BERT

- Approach 1: Masked LM

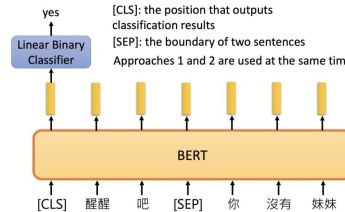


### 預訓練任務 2：下個句子預測

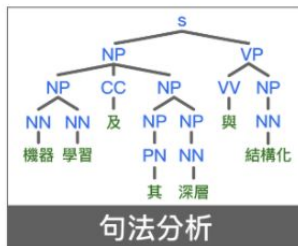
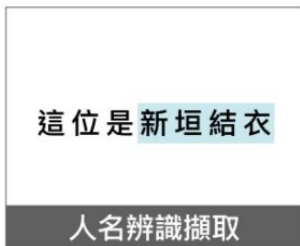
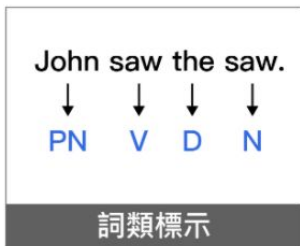
#### Training of BERT

##### Approach 2: Next Sentence Prediction

- [CLS]: the position that outputs classification results
- [SEP]: the boundary of two sentences
- Approaches 1 and 2 are used at the same time.

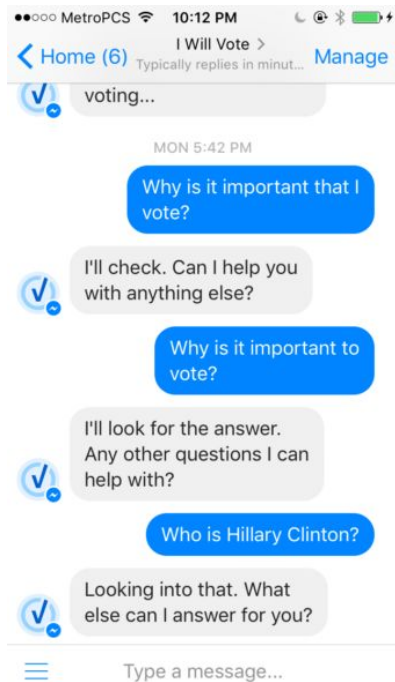


# Application

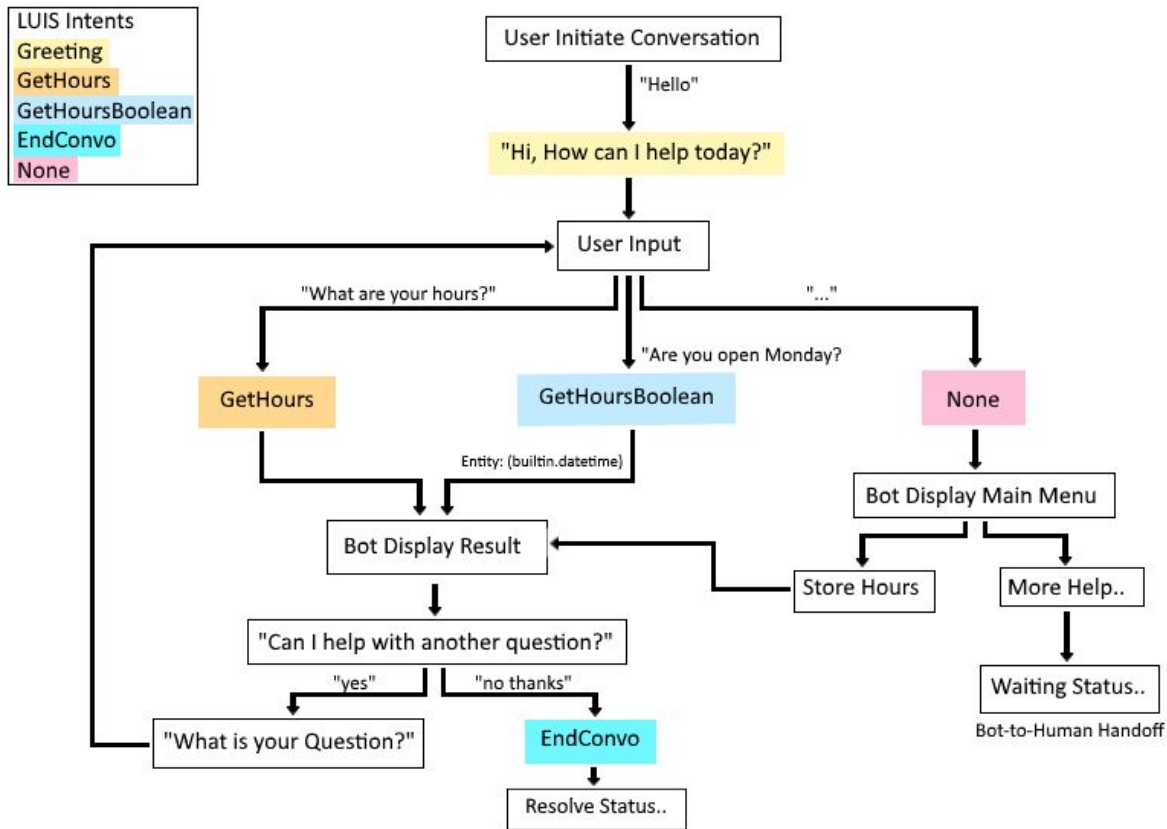


# Example

## Chatbot



### LUIS Intent and Dialog Flow

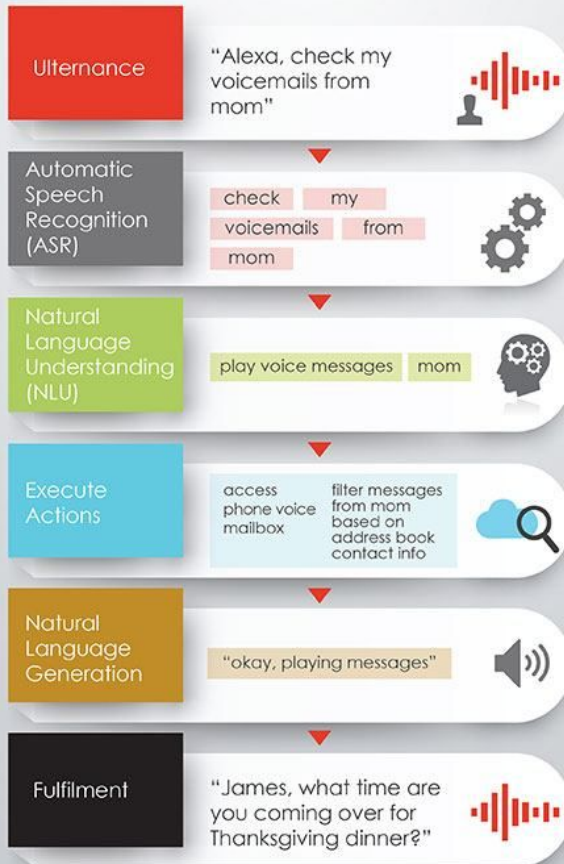


# Example

## Amazon Alexa

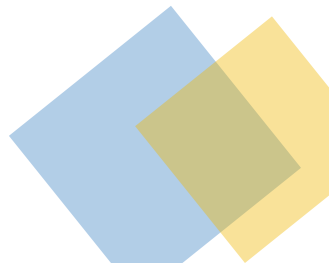


### NLP Mechanics Flow Diagram



# 參考資料

- [中研院 - 開中文的鎖鍊！自然語言處理 \(NLP\)](#)
- [維基百科 - 自然語言處理](#)
- [Quora - How does natural language processing work?](#)
- [自然語言處理 五大應用層面](#)
- [How NLP & Drupal Can Be Combined To Provide The Best User Experience](#)
- [Natural Language Processing \(NLP\) Market](#)
- [Uber Engineering - COTA: Improving Uber Customer Care with NLP & Machine Learning](#)
- [medium - A Practitioner's Guide to Natural Language Processing \(Part I\) — Processing & Understanding Text](#)
- [medium - Hierarchical Attention Networks](#)





# 資料檢索

Information Retrieval

# What is IR ?

- 資訊檢索是從資訊資源集合獲得與資訊需求相關的資訊資源的活動。
- 搜尋可以基於全文、關鍵字或是特殊的標記。
- 自動資訊檢索系統用於減少所謂的「資訊超載」。

(以上截取至維基百科)



# Stand shou

大家都說  
但是在這



好的 IR 系統可以幫你快速找到最高的那個巨人

# Key points



資料性質???

需求在哪???

DEMAND



MEHDI  
PARVATI



產生想法

# In Each Case

- **Location:** 想找附近咖啡廳時..
  - query: 附近咖啡廳、星巴克、北車咖啡廳
  - 需有位置訊息、相對關係、咖啡廳名字, 因此需要先對query 做初步的分類。
- **House:** 想買新房子時..
  - query: 台北市房子、1000 萬內、3房兩廳、大坪數、2樓
  - 大多都是要找符合條件的房子, 我們可以針對房子的屬性做篩選。



Data



Technical



Measurement



Environment



Medical



Statistics



Science



Security



Transport



Personal



Education



Library



Financial



Weather



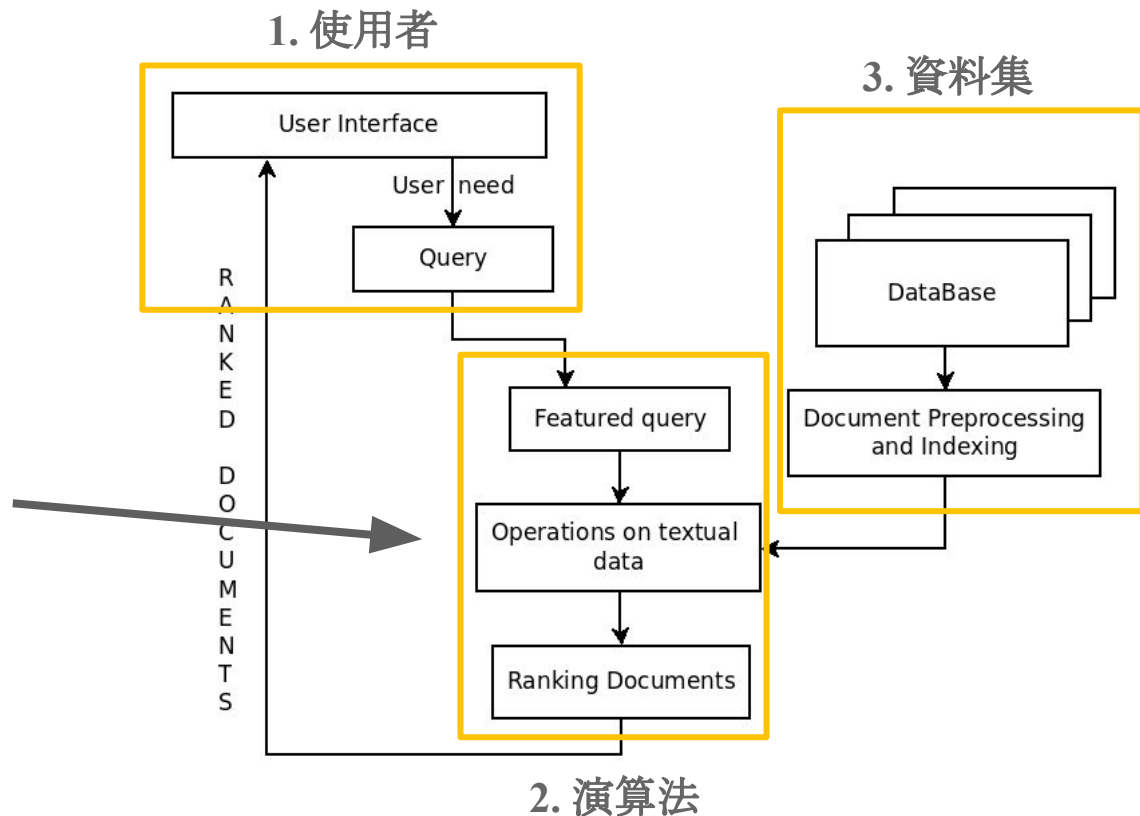
Location



House

# IR System

根據不同的情況與資訊需求，我們需要去設計適合的演算法，為了讓最後排名結果的命中率提高。



那如何計算命中率???

# Confusion Matrix

		預測結果		
		Positive	Negative	
真實情況	Positive	True Positive (TP)	False Negative (FN) Type II Error	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$ 查全率 (recall)
	Negative	False Positive (FP) Type I Error	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$ 查準率	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$ 準確率

# Confusion Matrix

**真實**    VS    **預測**

---

Accuracy(準確率) = 全部資料中, 成功**預測相關與非相關**的比例

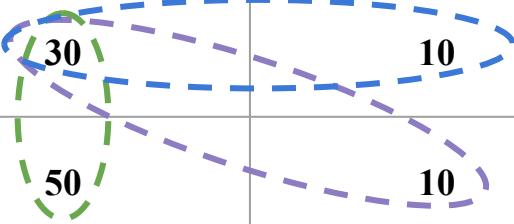
Precision(查準率) = **預測為相關**的資料中, **真的相關**的比例

Recall(查全率) = **真的相關**的資料中, **預測為相關**的比例



# Confusion Matrix

		預測	
		相關	非相關
真實	相關	30	10
	非相關	50	10



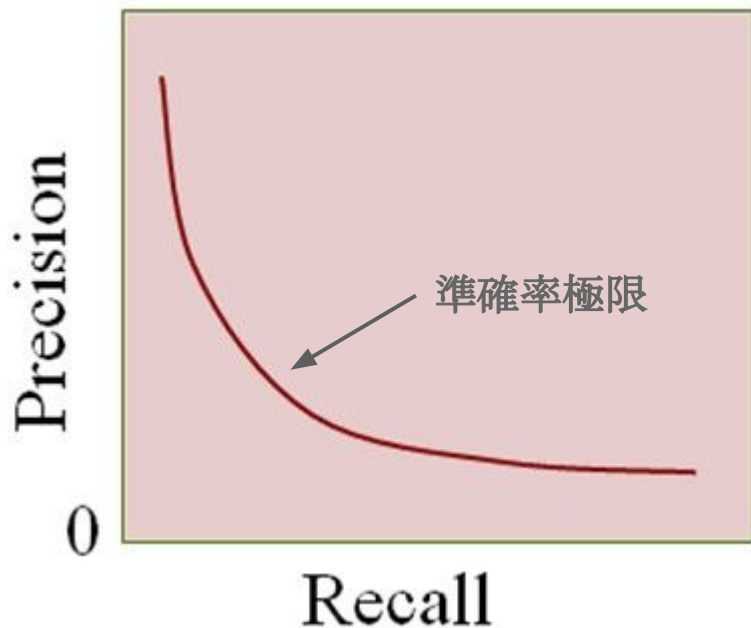
→ **Accuracy** =  $(30 + 10) / (30 + 10 + 50 + 10) = 40 \%$

→ **Precision** =  $30 / (30 + 50) = 37.5 \%$

→ **Recall** =  $30 / (30 + 10) = 75 \%$



# Confusion Matrix



Precision  
(查準率)

vs

Recall  
(查全率)

當我們演算法達到**準確率極限**, 我們可以一定程度控制演算法。當演算法**偏向預測相關**時可以**提高 Recall**, 反之**偏向預測非相關**時 **Precision** 會提高。

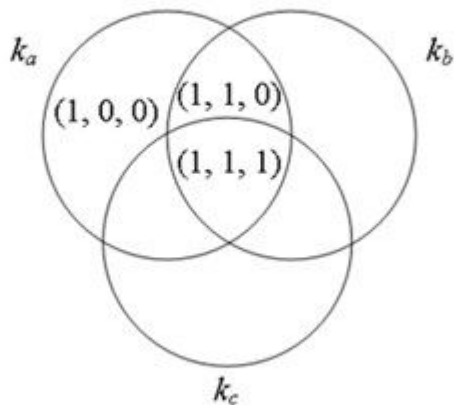
要選擇 Recall 還是 Precision???

# Methon Boolean Model

$$(k_a \wedge \sim k_b \wedge \sim k_c) \vee$$

$$(k_a \wedge k_b \wedge \sim k_c) \vee$$

$$(k_a \wedge k_b \wedge k_c)$$



## 優點

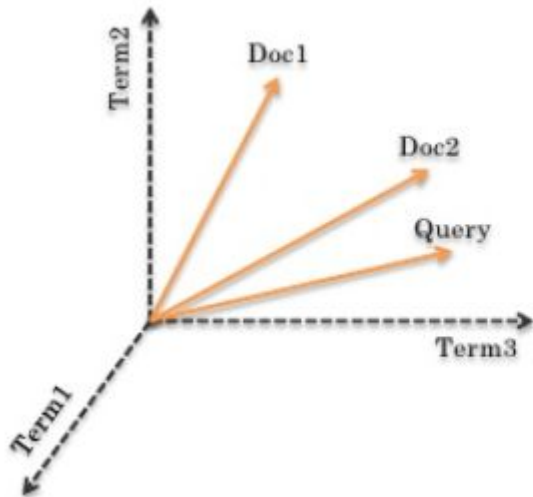
就是二元分類的意思，簡單、好懂、速度快，需要有明確的時可以使用。

## 缺點

只有是與否，無法表示程度關係，所以不能依據相關程度去做排序。

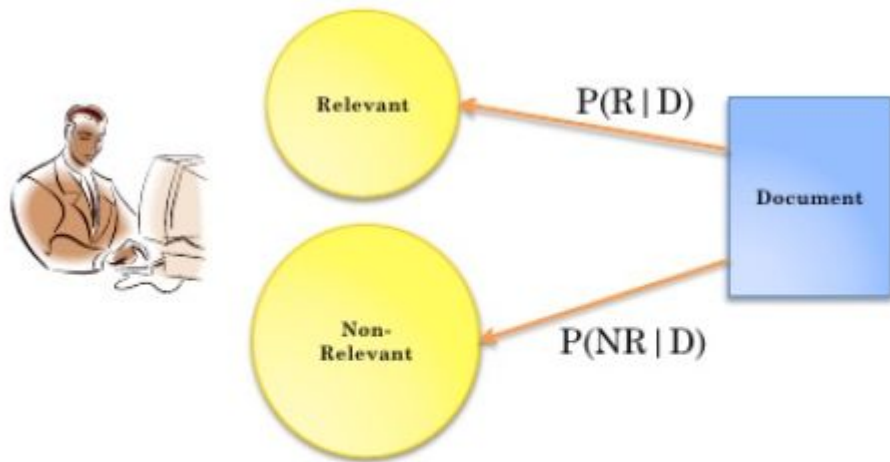
# Methon Vector Model

相較於布林檢索，向量檢  
索會參照部分的相似度，  
因此可以容忍一定程度  
的差異性。



$$\text{Cos}(D_i, Q) = \frac{\sum_{j=1}^t d_{ij} * q_j}{\sqrt{\sum_{j=1}^t d_{ij}^2 * \sum_{j=1}^t q_j^2}}$$

# Methon Probabilistic Model



$$\text{Bayes' Rule } P(R | D) = \frac{P(D | R)P(R)}{P(D)}$$

機率模型透過計算已標記資料的分布關係, 建立模型, 此模型可以估計新的query 對應各個文件的相關機率, 並以此作為結果排序。

# Document-term matrix

	it	is	puppy	cat	pen	a	this
it is a puppy	1	1	1	0	0	1	0
it is a kitten	1	1	0	0	0	1	0
it is a cat	1	1	0	1	0	1	0
that is a dog and this is a pen	0	2	0	0	1	2	1
it is a matrix	1	1	0	0	0	1	0

直觀，但是浪費許多空間!!

# Inverted Index

docID		geo-scopeID
1		Europe
2		Europe
3		France
4		England
5		Portugal
6		Quebec
7		Europe
8		Spain

Forward Index

geo-scopeID		docID
Europe		1 2 7
France		3
Portugal		5
England		4
Quebec		6
Spain		8

Inverted Index

可以看到相較於 DTM 節省了許多空間

# Edit Distance

## Levenshtein

編輯距離，就是在比較兩個詞之間要更正多少次才會一樣。可以用來糾正輸入錯誤的情況。

字母一樣: +0  
字母不一樣: +1

(左上, 上, 左)  
= (1+0, 2+1, 2+1)  
= (1, 3, 3)

→ ? = 1  
**min**

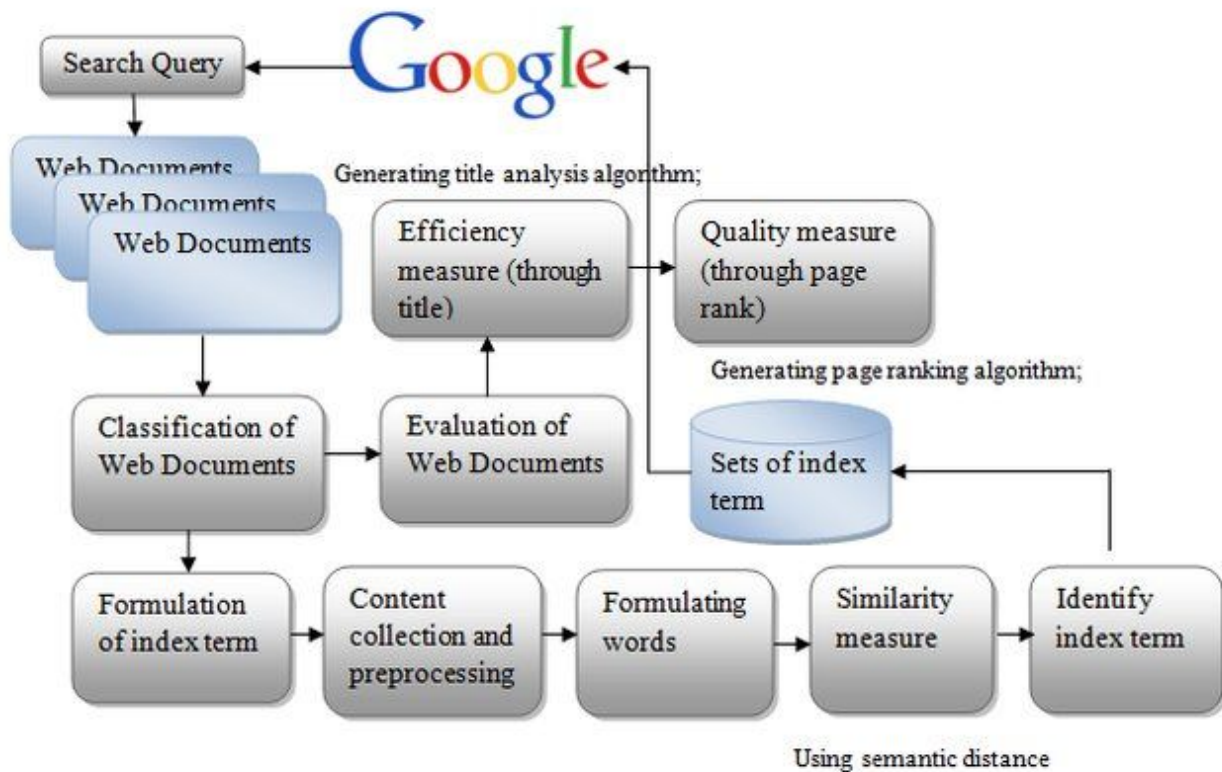
左上	上 +1
左 +1	?

	""	P	A	I	R	S
""	0	1	2	3	4	5
C	1	1	2	3	4	5
A	2	2	1	2	3	4
R	3	3	2	2	2	3
S	4	4	3	3	3	2

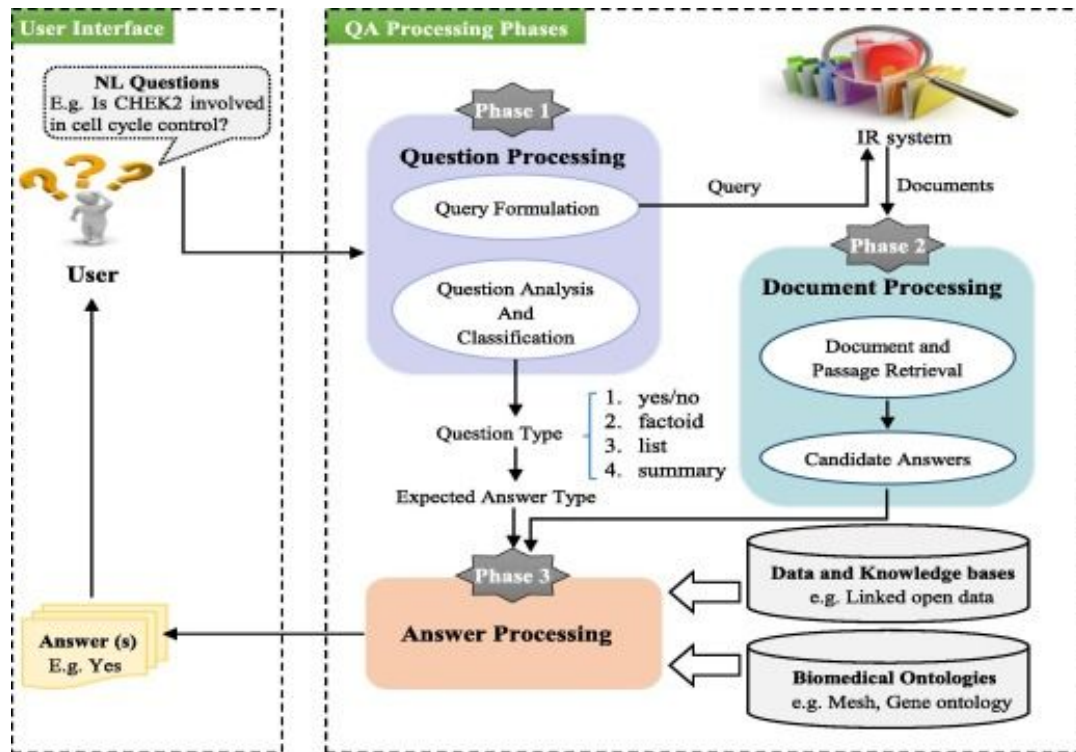




# Example Google Search

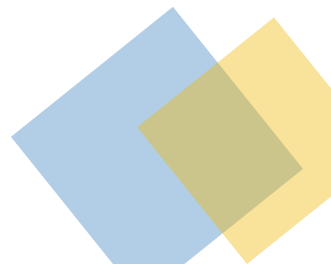


# Example QA system



# 參考資料

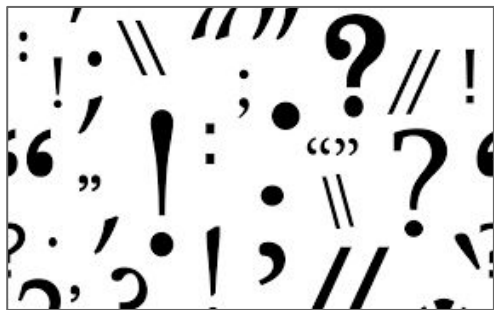
- [維基百科 - Information retrieval](#)
- [4. Information Retrieval](#)
- [The Process of Information Retrieval](#)
- [編輯距離演算法 \(Edit Distance\)](#)
- [如何辨別機器學習模型的好壞？秒懂 Confusion Matrix](#)
- [parper - Semantic Recognition of Web Structure to Retrieve Relevant Documents from Google by Formulating Index Term](#)
- [parper - A passage retrieval method based on probabilistic information retrieval model and UMLS concepts in biomedical question answering](#)



# 文字前處理

Text Preprocessing

# Pipeline



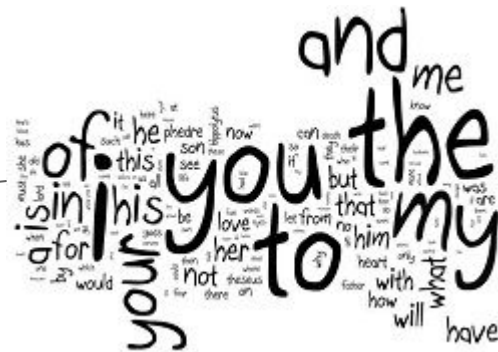
## Lower Casing

## Removal of Punctuations

## Removal of stopwords

## Removal of Frequent words

## Removal of Rare words



## 词干提取 – Stemming

## Stemming

## Lemmatization

相較於 Stemming 只是去掉詞中的詞綴。Lemmatization 他是依造詞庫(**WordNet**)去做指定詞性的詞性還原，所以會出現原本沒有的字。像是: **ate** -> **eat**



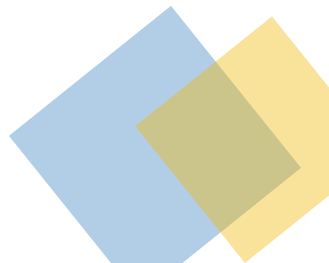
斷詞

Word Segmentation

# Process

---

1. 初步斷詞 → 先將句子拆成一個個詞
2. 未知詞偵測 → 找出哪些詞不存在於資料庫
3. 中國人名擷取 → 判斷這個未知詞是不是中文人名
4. 歐美譯名擷取 → 判斷這個未知詞是不是歐美人名翻譯
5. 複合詞擷取 → 分析未知詞的詞性結構
6. bottom-up merging algorithm → 運用統計方法找出專有詞彙
7. 重新斷詞 → 重新將句子拆成一個個詞

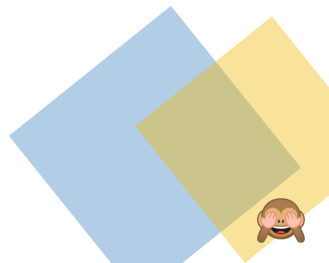


# Tools

---

- CkipTagger
  - 中研院 CKIP lab 在繁體中文上效果最好
- Stanza
  - 由 stanford nlp 建立的, 支持 66 種語言, 功能多且穩定。
- jieba
  - 最簡單的開源工具, 速度快, 可擴充字典。

不用怕我們有工具!!





詞嵌入

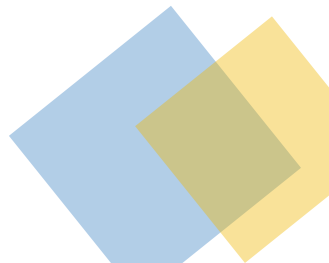
Word Embedding

# What is Word Embedding?

---

- 詞嵌入 (Word embedding) 是自然語言處理 (NLP) 中**語言模型 (Language Model)**與**表徵學習 (Representation Learning)**技術的統稱。概念上而言，它是指把一個維數為所有詞的數量的高維空間嵌入到一個維數低得多的連續向量空間中，**每個單詞或詞組被映射為實數域上的向量**。
- 詞嵌入的方法包括**人工神經網絡**、**對詞語同現矩陣降維**、**概率模型**以及**單詞所在上下文的顯式表示**等。
- 在**底層輸入**中，使用詞嵌入來表示詞組的方法極大提升了NLP中語法分析器和文本情感分析等的效果。

(以上截取至維基百科)



# One-hot encoding

是最簡單的編碼方式。和DTM一樣有維度災難且浪費空間。將各單詞視為獨立，無法表示詞之間的關係。

## 1-of-N Encoding

apple = [ 1 0 0 0 0 ]

bag = [ 0 1 0 0 0 ]

cat = [ 0 0 1 0 0 ]

dog = [ 0 0 0 1 0 ]

elephant = [ 0 0 0 0 1 ]

# Bag of Words

```
I like apple.  
{"I": 1, "like": 1, "apple": 1}
```

```
I like mango.  
{"I": 1, "like": 1, "mango": 1}
```

忽略掉了很重要的次序關係

```
I like apple, but I don't like mango.  
I like mango, but I don't like apple.
```

# Word vector

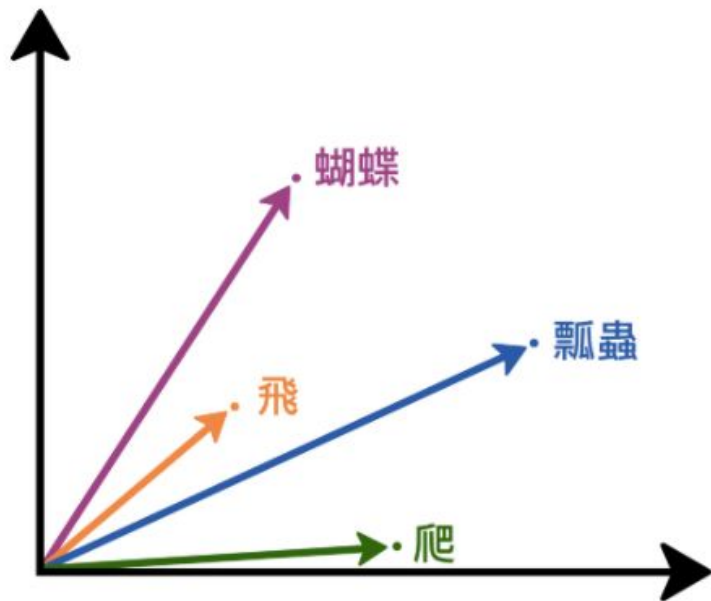
將字詞轉換成向量

$$\text{蝴蝶} = \begin{bmatrix} 0.234 \\ 0.283 \\ -0.435 \\ 0.485 \\ -0.934 \\ -0.384 \\ 0.234 \\ 0.548 \\ -0.834 \\ 0.437 \\ 0.483 \end{bmatrix}$$



可視覺  
化來觀察字詞之間的關聯性

計算向量的相似遠近程度



# Count Vector Frequency based

	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

← Word Vector (Passage Vector)

Document Vector

右圖為 counter vector matrix, 和之前介紹的DTM類似, 只是他是用數量表示一個term在對應doc裡面。其中count vector就是直接取其文字的部分, 所以count vector長度就是doc的總數。

# TF - IDF

Frequency based

$$w_{x,y} = \text{tf}_{x,y} \times \log \left( \frac{N}{\text{df}_x} \right)$$

**TF-IDF**

Term  $x$  within document  $y$

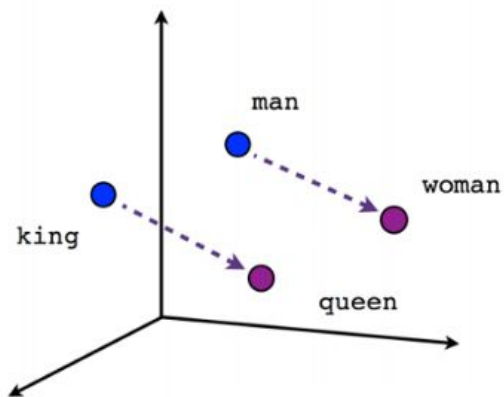
$\text{tf}_{x,y}$  = frequency of  $x$  in  $y$

$\text{df}_x$  = number of documents containing  $x$

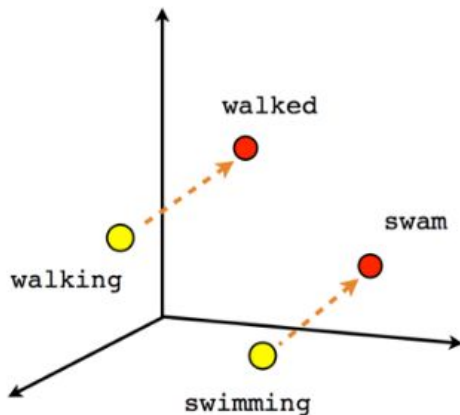
$N$  = total number of documents

太多的詞和太少出現的詞是雜訊

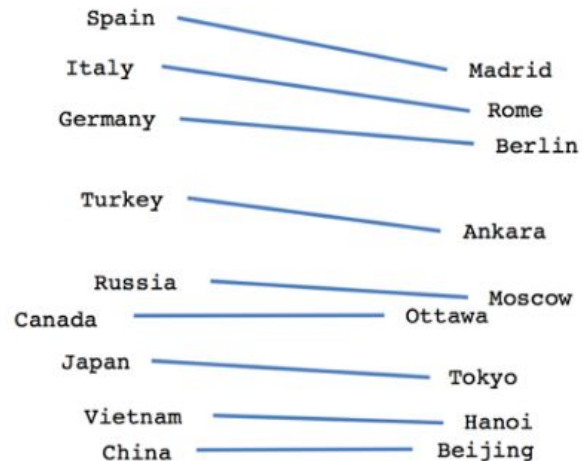
# Wrod2vec Predicted Based



Male-Female



Verb tense



Country-Capital



# Doc2Vec

## Frequency based

# 參考資料

- [維基百科 - 詞嵌入](#)
- [Top NLP Algorithms & Concepts](#)
- [詞向量 Word Embedding](#)
- [Word Embedding 編碼矩陣](#)
- [Neural Network Embeddings Explained](#)
- [〈Gensim Word2Vec 簡易教學〉](#)

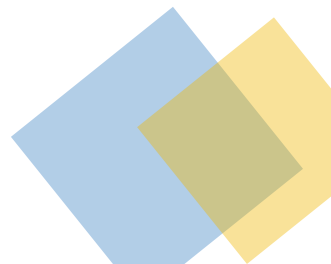


# 詞語關聯

Word Association

# 參考資料

- [Top NLP Algorithms & Concepts](#)



# 主題模型

Topic Model

# 參考資料

- 直觀理解 LDA (Latent Dirichlet Allocation) 與文件主題模型

