



FOUNDATIONS OF COMPUTER VISION 電腦視覺的原理與應用

電腦視覺的步驟 (1/3)

1. 圖像取得：數位圖像是由一或多個圖像傳感器產生，傳感器可以是各種攝錄影機，包括X-Ray斷層掃描，雷達，超聲波等，圖片可以是二維、三維圖組或者一個圖像序列。圖片像素值對應在一個或多個光譜上（如灰階圖）。
2. 預處理：對圖像提取某種特定的資訊，使圖像滿足後繼方法的要求。如：
 - 二次取樣保證圖像坐標的正確
 - 平滑去噪來濾除傳感器引入的裝置雜訊
 - 提高對比度來保證實現相關資訊可以被檢測到
 - 調整尺度空間使圖像結構適合局部應用

電腦視覺的步驟 (2/3)

3. 特徵提取：從圖像中提取各種複雜度的特徵。

- 線、邊的提取
- 局部的特徵點檢測，如邊角、斑點檢測
- 更複雜的特徵可能與紋理或形狀有關。

4. 檢測/分割：對圖像分割並提取用於後繼處理的部分。

- 篩選特徵點
- 分割一或多幅圖片中含有特定目標的部分

電腦視覺的步驟 (3/3)

5. 進階處理：資料往往已經精煉到很小的數量，如含有目標物體的部分。

- 驗證得到的資料是否符合前提要求
- 估測特定係數，比如目標的姿態，體積
- 對目標進行分類

COLOR HISTOGRAM



- 顏色都是由紅綠藍三原色（RGB）構成的，所以左圖共有4張直方圖（三原色直方圖 + 最後合成的直方圖）。
- 每種原色都可以取256個值，那麼整個顏色空間共有1600萬種顏色（256的三次方）。
- 針對1600萬種顏色比較直方圖，計算量太大，因此需要簡化。
 - 可以將0~255分成四個區：0~63為第0區，64~127為第1區，128~191為第2區，192~255為第3區。
 - 紅綠藍分別有4個區，總共可以構成64種組合（4的3次方）。

COLOR HISTOGRAM

红	绿	蓝	像素数量	红	绿	蓝	像素数量	红	绿	蓝	像素数量	红	绿	蓝	像素数量
0	0	0	7414	1	0	0	891	2	0	0	1146	3	0	0	11
0	0	1	230	1	0	1	13	2	0	1	0	3	0	1	0
0	0	2	0	1	0	2	0	2	0	2	0	3	0	2	0
0	0	3	0	1	0	3	0	2	0	3	0	3	0	3	0
0	1	0	8	1	1	0	592	2	1	0	2552	3	1	0	856
0	1	1	372	1	1	1	3462	2	1	1	9040	3	1	1	1376
0	1	2	88	1	1	2	355	2	1	2	47	3	1	2	0
0	1	3	0	1	1	3	0	2	1	3	0	3	1	3	0
0	2	0	0	1	2	0	0	2	2	0	0	3	2	0	0
0	2	1	0	1	2	1	101	2	2	1	8808	3	2	1	3650
0	2	2	10	1	2	2	882	2	2	2	53110	3	2	2	6260
0	2	3	1	1	2	3	16	2	2	3	11053	3	2	3	109
0	3	0	0	1	3	0	0	2	3	0	0	3	3	0	0
0	3	1	0	1	3	1	0	2	3	1	0	3	3	1	0
0	3	2	0	1	3	2	0	2	3	2	170	3	3	2	3415
0	3	3	0	1	3	3	0	2	3	3	17533	3	3	3	53929

- 左圖是某張圖片的顏色分佈表，將表中最後一欄提取出來，組成一個64維向量(7414, 230, 0, 0, 8, ..., 109, 0, 0, 3415, 53929)。這個向量就是這張圖片的**特徵值**或者叫**"指紋"**。
- 尋找相似圖片就變成**找出與其最相似的向量**，可利用Pearson相關係數或Cosine相似度算出。

PERCEPTUAL HASH ALGORITHM

- 利用以下步驟對每張圖片生成“指紋”(fingerprint)字符串來比較不同圖片的指紋。
 1. 縮小尺寸：只保留圖片的結構，明暗等基本資訊，避免不同尺寸的差異。如縮成 8×8 。
 2. 簡化色彩：將圖片轉為64級灰度，所有像素點總共只有64種顏色。
 3. 計算平均值：計算所有64個像素的灰度平均值。
 4. 比較像素的灰度：將每個像素的灰度與平均值比較， \geq 平均值，記為1、 $<$ 平均值，記為0。
 5. 計算 Hash 值：這64位的整數組合，就是圖片的指紋。組合的次序並不重要，只要保證所有圖片都採用同樣次序就行了，在此以 Hash 加密方法來組成。
 - 雜湊演算法(Hash Algorithm)是一種從資料中建立「數位指紋(Digital fingerprint)」的方法，可以將任何長度的資料轉換成一個長度較短的「雜湊值(Hash value)」，又稱為「訊息摘要(MD：Message Digest)」。

PERCEPTUAL HASH ALGORITHM

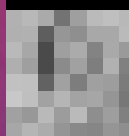
原圖



1. 縮小 (8*8)



2. 簡化色彩 (64級灰度)



8級灰度



32級灰度



3. 計算64個像素的灰度平均值

4. 每個像素與灰度平均值比較



5. 得出 Hash 值

8f373714acfcf4d0

- 得到指紋以後，就可以對比不同的圖片，看看64位中有多少位是不一樣的。如果不相同的數據位不超過5，就說明兩張圖片很相似；如果大於10，就說明這是兩張不同的圖片。



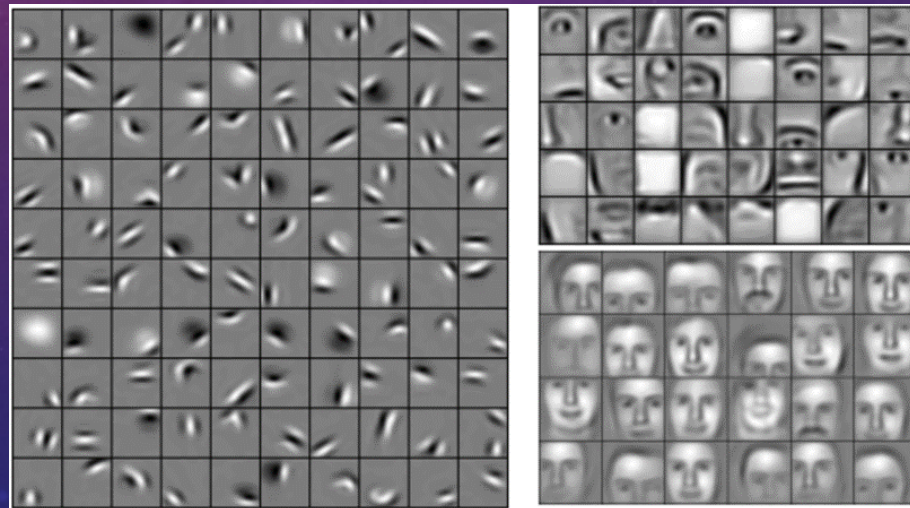
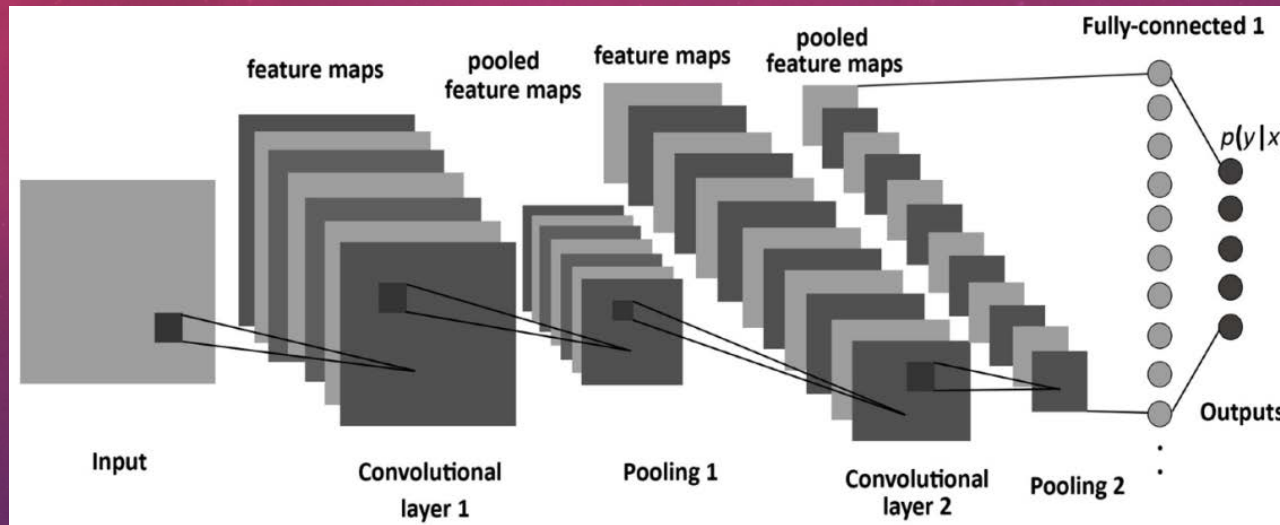
卷積神經網路 CONVOLUTIONAL NEURAL NETWORK

參考文獻

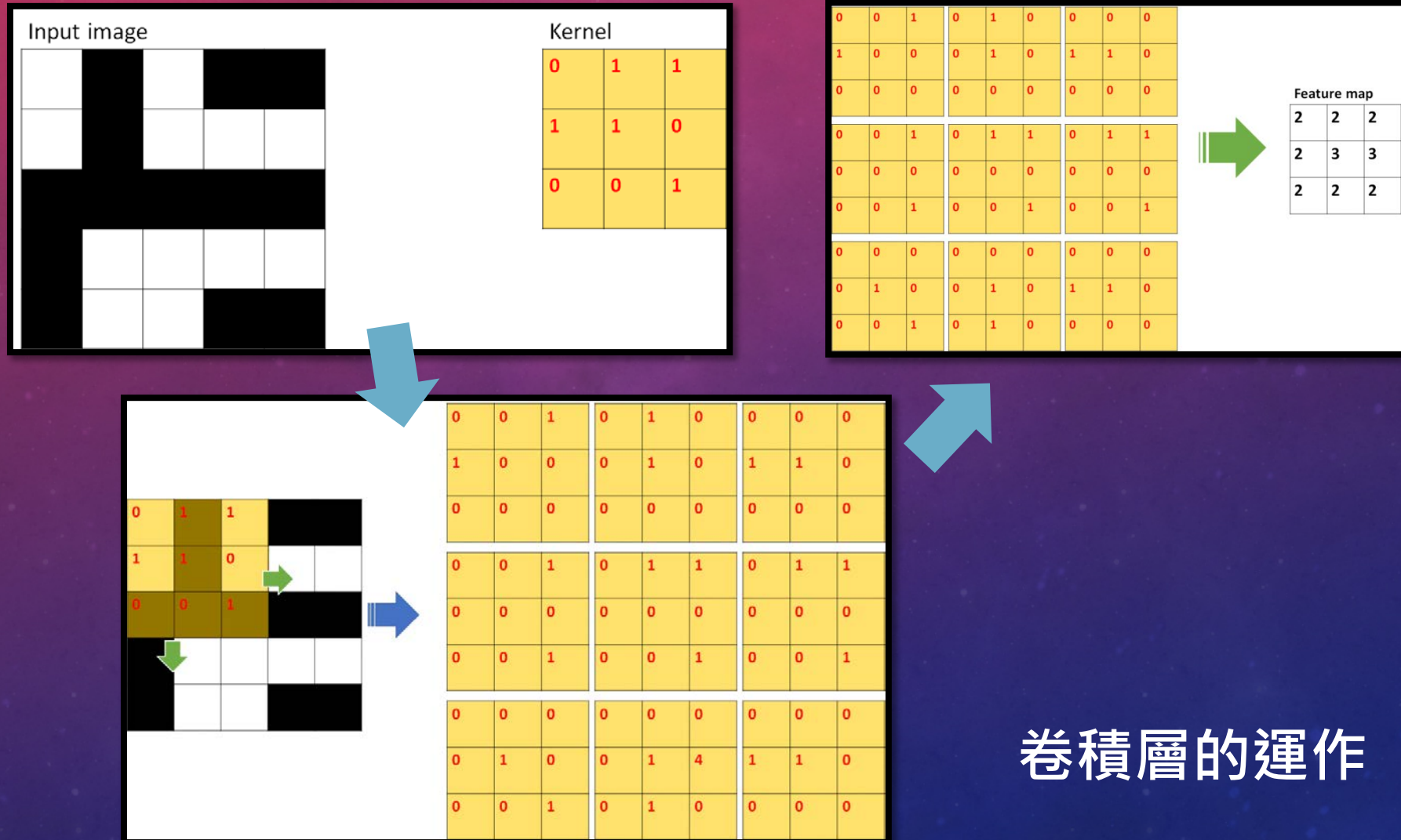
[HTTPS://GOO.GL/FJVJG1](https://goo.gl/FJVJG1)

[HTTPS://GOO.GL/Q5YKPK](https://goo.gl/Q5YKPK)

CONVOLUTIONAL NEURAL NETWORK



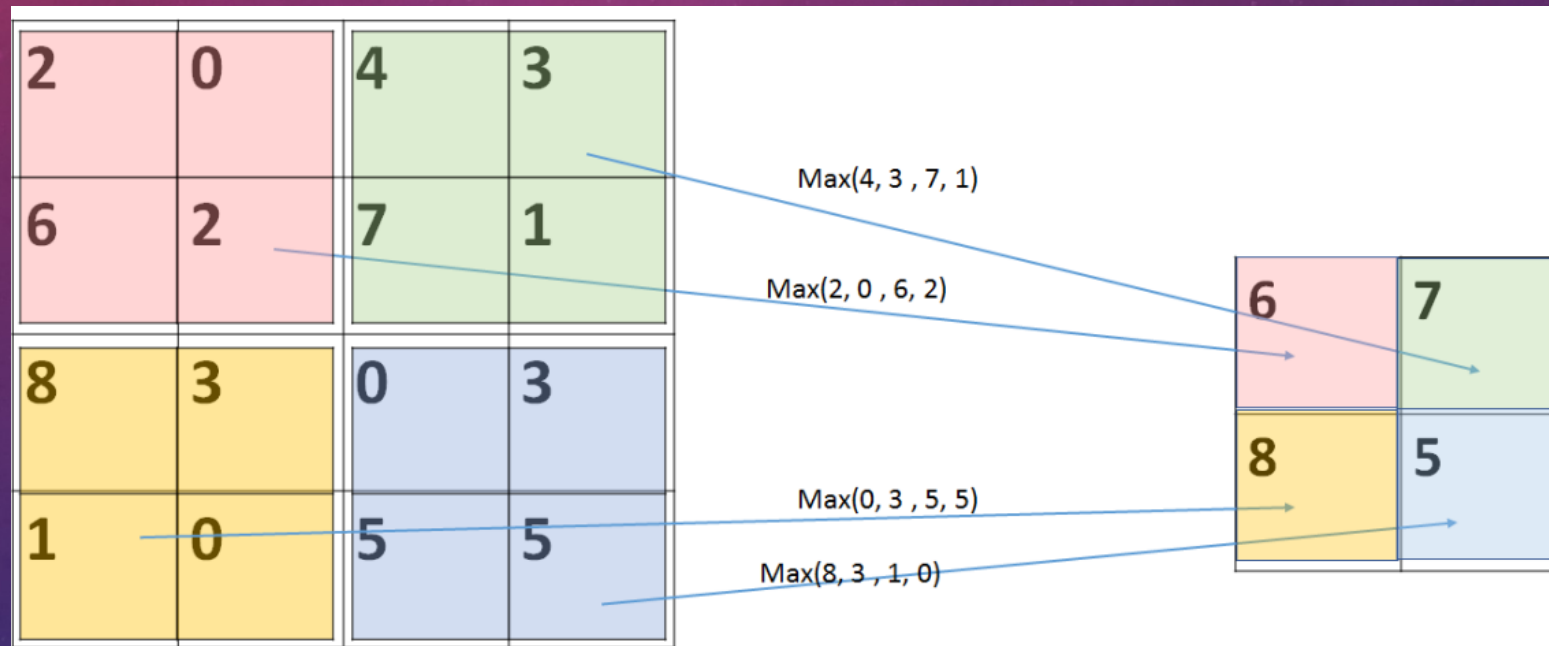
CONVOLUTIONAL NEURAL NETWORK



卷積層的運作

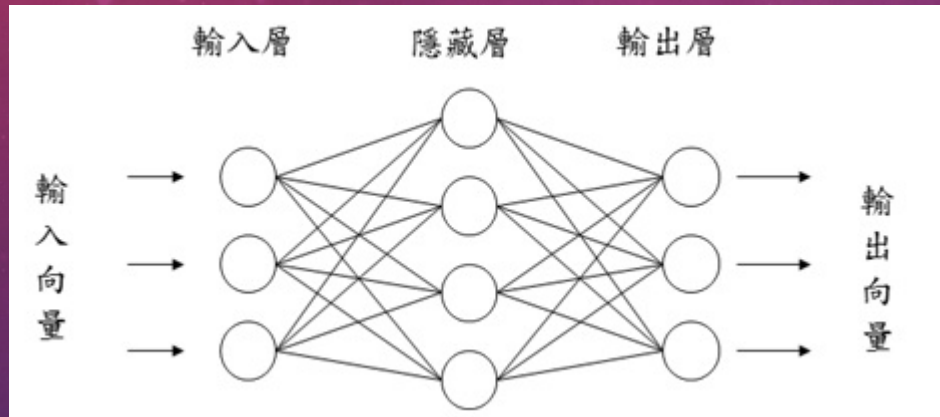
CONVOLUTIONAL NEURAL NETWORK

最大池化層的運作



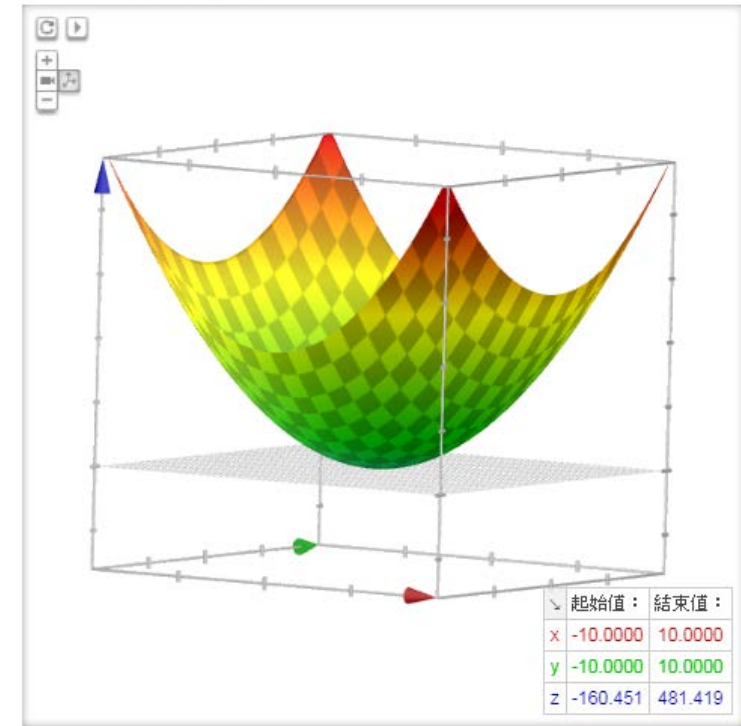
CONVOLUTIONAL NEURAL NETWORK

全連接層的運作



$$3x^2 + 2y^2$$

$3x^2 + 2y^2$ 的圖表

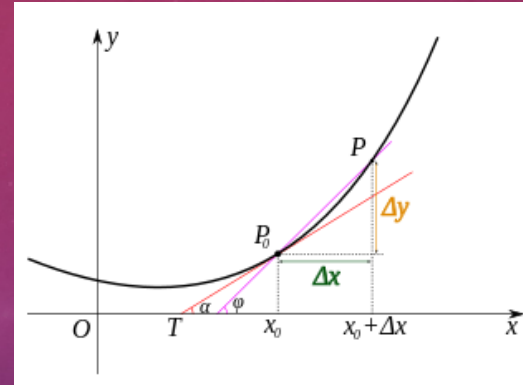
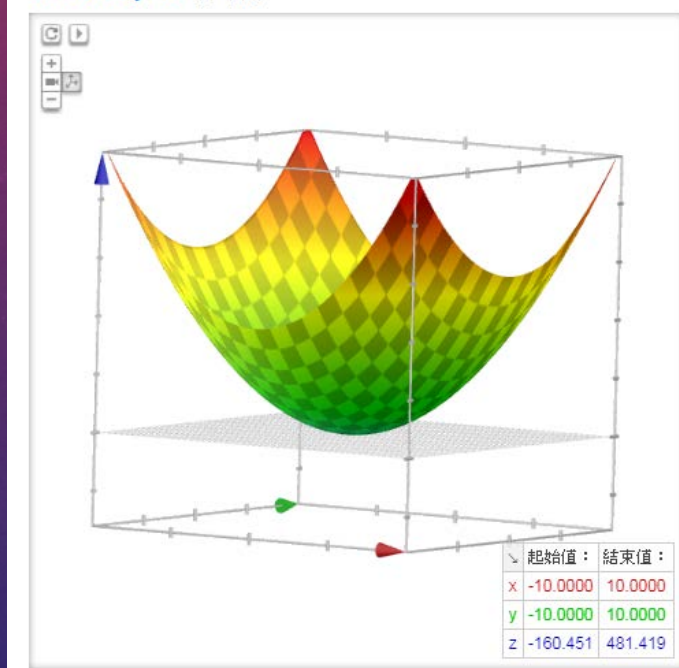


CONVOLUTIONAL NEURAL NETWORK

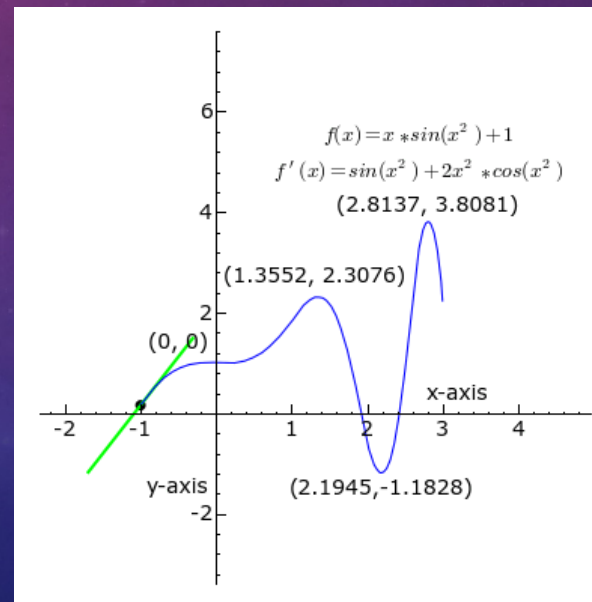
全連接層的運作

$$3x^2 + 2y^2$$

$3x^2 + 2y^2$ 的圖表



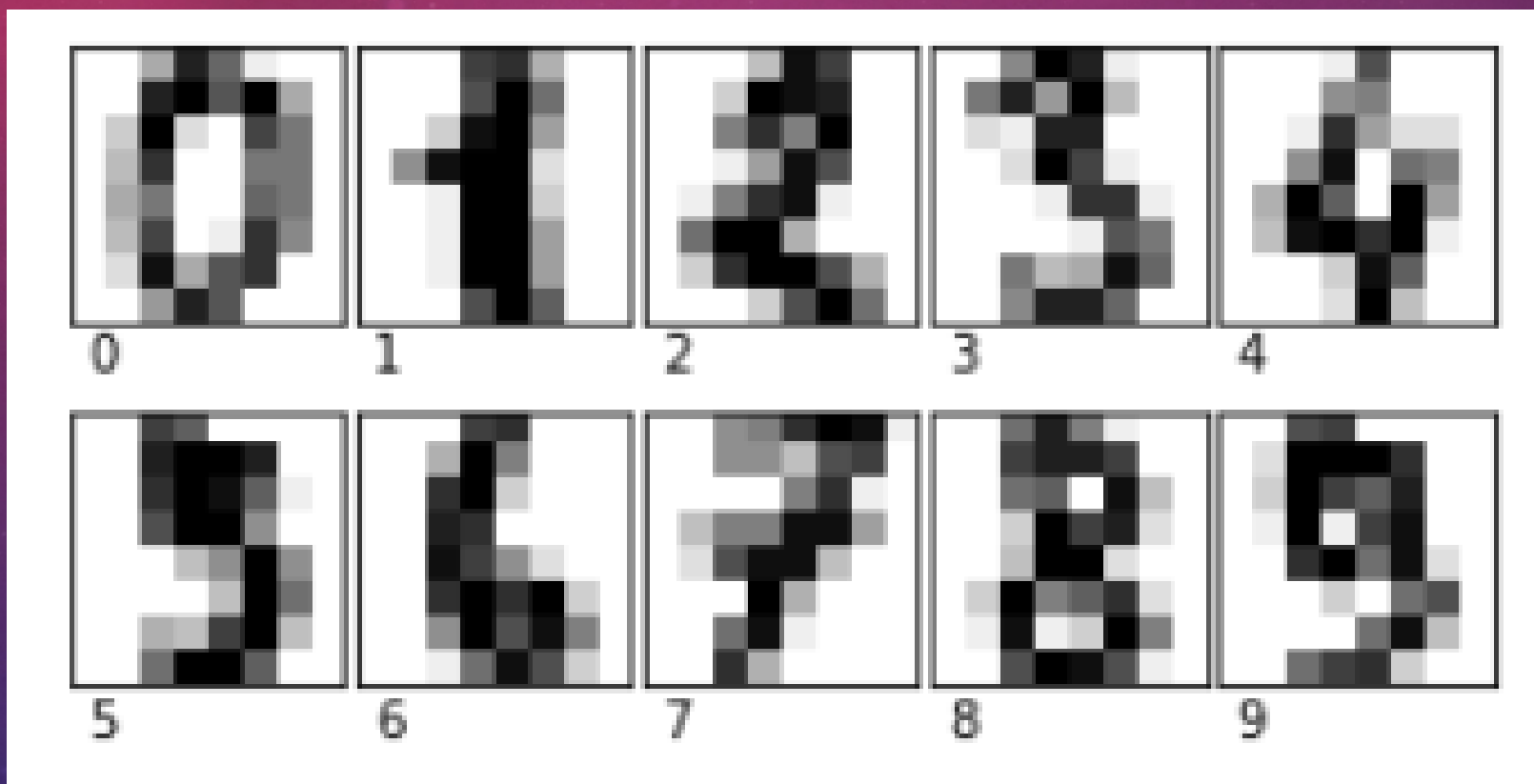
$$\tan \alpha = \lim_{\Delta x \rightarrow 0} \tan \varphi = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$



The background features a gradient from deep red at the top to dark blue at the bottom, speckled with white dots resembling stars. Overlaid on this are several faint, white circular and semi-circular lines, some with arrows indicating a clockwise direction. A prominent circular scale on the left side has numerical markings from 140 to 260 in increments of 10.

實際案例

手寫數字辨識

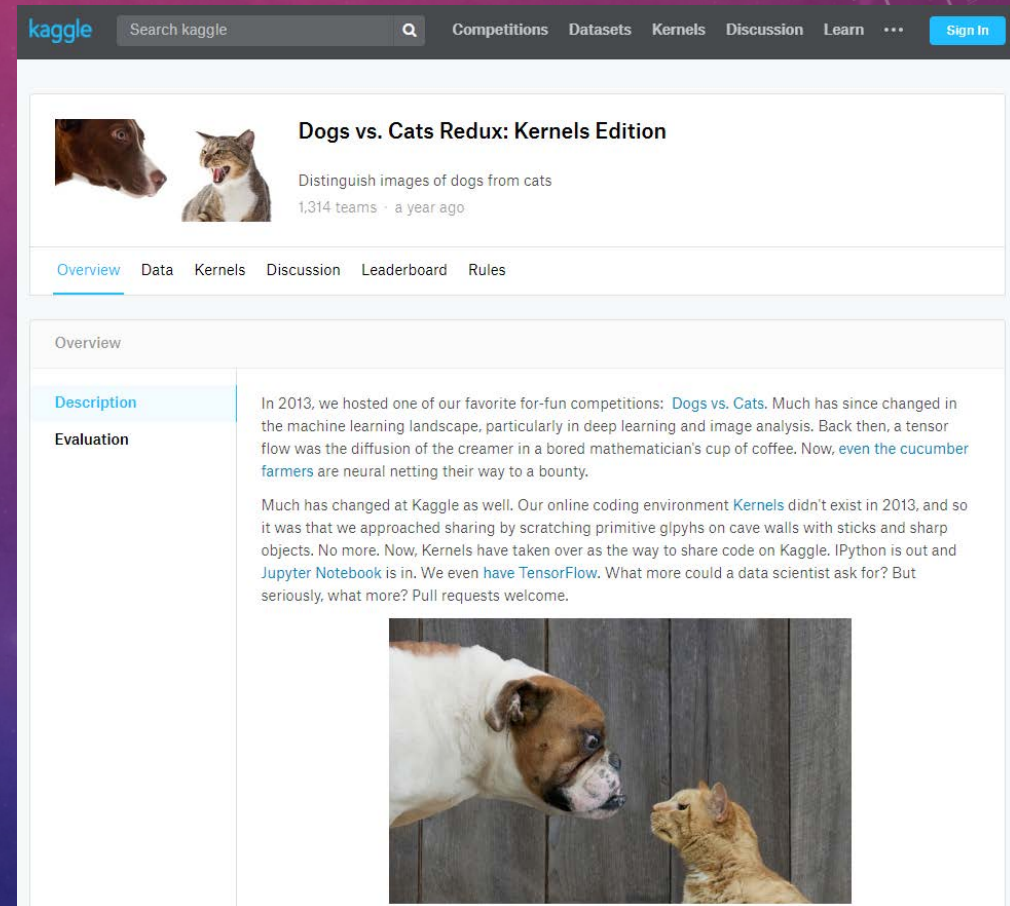


手寫數字辨識

	precision	recall	f1-score	support
0	1.00	0.98	0.99	41
1	0.91	0.95	0.93	42
2	0.94	0.92	0.93	37
3	0.89	0.95	0.92	44
4	0.96	1.00	0.98	49
5	0.96	0.91	0.93	47
6	0.98	1.00	0.99	45
7	0.98	0.98	0.98	60
8	0.87	0.85	0.86	40
9	0.98	0.91	0.94	45
avg / total	0.95	0.95	0.95	450

DOGS VS. CATS

- <https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition>
 - Training data: 25000 images
 - Test data: 12500 images
 - For each image in the test set, you should predict a probability that the image is a dog (1 = dog, 0 = cat).



The screenshot shows the Kaggle website interface for the 'Dogs vs. Cats Redux: Kernels Edition' competition. The header includes the Kaggle logo, a search bar, and navigation links for Competitions, Datasets, Kernels, Discussion, Learn, and a Sign In button. The competition title is 'Dogs vs. Cats Redux: Kernels Edition', with a subtitle 'Distinguish images of dogs from cats' and '1,314 teams · a year ago'. Below the title are tabs for Overview, Data, Kernels, Discussion, Leaderboard, and Rules. The 'Overview' tab is selected, showing a 'Description' section with text about the competition's history and the 'Evaluation' section. A large image of a dog and a cat is displayed at the bottom of the overview page.

Dogs vs. Cats Redux: Kernels Edition
Distinguish images of dogs from cats
1,314 teams · a year ago


[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [Rules](#)

Overview

Description

In 2013, we hosted one of our favorite for-fun competitions: [Dogs vs. Cats](#). Much has since changed in the machine learning landscape, particularly in deep learning and image analysis. Back then, a tensor flow was the diffusion of the creamer in a bored mathematician's cup of coffee. Now, [even the cucumber farmers](#) are neural netting their way to a bounty.

Much has changed at Kaggle as well. Our online coding environment [Kernels](#) didn't exist in 2013, and so it was that we approached sharing by scratching primitive glyphs on cave walls with sticks and sharp objects. No more. Now, Kernels have taken over as the way to share code on Kaggle. IPython is out and [Jupyter Notebook](#) is in. We even [have TensorFlow](#). What more could a data scientist ask for? But seriously, what more? Pull requests welcome.

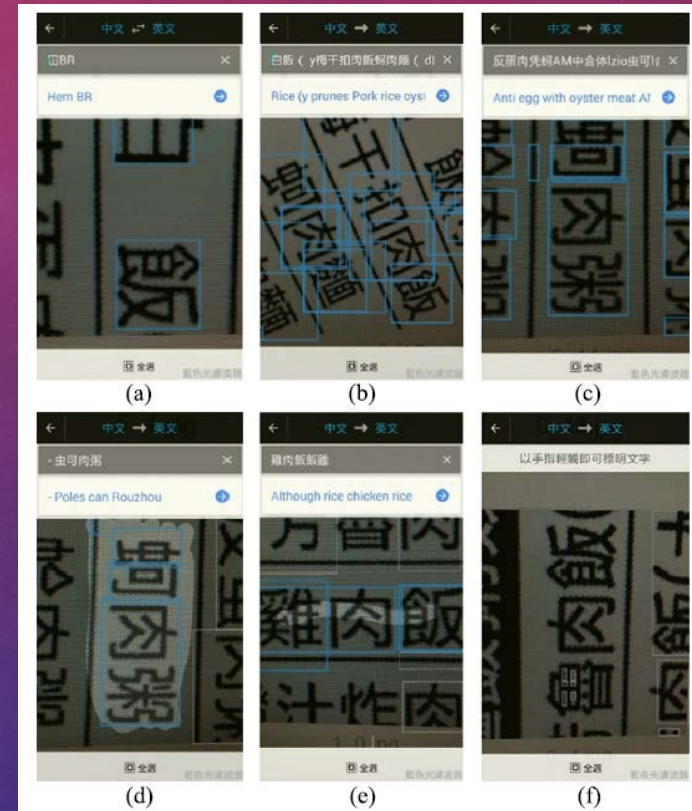


貓狗辨識的結果

[0.14723705	0.98256]
[0.04469623	0.99915826]	
[0.99998796	0.00834211]	
[0.99993527	0.01661933]	
[0.99877125	0.05415697]	
[0.43036035	0.6431105]
[0.20522958	0.95668554]	
[0.8880429	0.28058085]	
[0.08643436	0.9956601]
[0.99908304	0.04826429]	
[0.9999168	0.01841162]	
[0.1397599	0.98483086]	
[0.99558544	0.08876048]	
[0.43750185	0.6269166]
[0.10178897	0.9934009]
[0.10858331	0.9922093]



中文OCR辨識為例



- Lee, M. C., Chiu S. Y., & Chang, J. W. (2017) A Deep Convolutional Neural Network based Chinese Menu Recognition App. Information Processing Letters, 128, 14-20. <https://doi.org/10.1016/j.ipl.2017.07.010> (SCI, COMPUTER SCIENCE, INFORMATION SYSTEMS)

VIDEO TO VIDEO



VIDEO-TO-VIDEO SYNTHESIS

The paper "Video-to-Video Synthesis" and its source code is available here:
<https://tcwang0509.github.io/vid2vid/>
<https://github.com/NVIDIA/vid2vid>

視覺整合信號轉換的體感操控機械手臂



The background features a vertical gradient from deep blue at the bottom to magenta at the top. On the left side, there are several white circular and semi-circular patterns. A prominent circular scale with degree markings from 140 to 260 is visible. Other smaller circles and arcs, some with arrows, are scattered across the left half of the image. The text 'THANK YOU' is positioned on the right side in a white, bold, sans-serif font.

THANK YOU



NATURAL LANGUAGE PROCESSING

自然語言處理的原理與應用

自然語言處理的主要範疇

- 機器翻譯 (Machine Translation)
- 自然語言理解/語意分析 (Natural Language Understanding / Semantic Analysis)
 1. 問答系統 (Question Answering)
 2. 萃取式摘要 (Extractive Summarization)
 3. 文件分類 (Text Categorization)
- 自然語言生成 (Natural Language Generation)
 1. 進階問答系統 (Advanced Question Answering)
 2. 抽象式摘要 (Abstractive Summarization)
 3. 聊天機器人 (Chatbot)
- 語法分析 (Syntactic Parsing)
 1. 中文斷詞 (Chinese word segmentation)
 2. 詞性標註 (Part-of-speech Tagging)
 3. 實體辨識 (Named Entity Recognition)
 4. 詞彙依存 (Typed Dependencies)
 5. 文法樹 (Parse Tree)
- 語音辨識 (Speech Recognition)
- 文字轉語音 (Text to Speech)
- 語音轉文字 (Speech to Text)



機器翻譯

MACHINE TRANSLATION

GOOGLE 翻譯



翻譯 關閉即時翻譯 

英文 中文 日文 偵測語言 ▼

↔ 中文(繁體) 英文 中文(簡體) ▼ 翻譯

My dog also likes eating sausage.

   ▼

 33/5000

我的狗也喜歡吃香腸。

☆   

 提出修改建議

Wǒ de gǒu yě xǐhuān chī xiāngcháng.

BING 翻譯

英文 (已偵測)

↔

繁體中文

英文

義大利文

My dog also likes eating sausage.

33/5000

我的狗也喜歡吃香腸。

wǒ de gǒu yě xǐ huān chī xiāng cháng.

有道翻译

检测到：英语 » 中文

翻译

人工翻译

划词

My dog also likes eating sausage.

×

G

33/5000

我的狗也喜欢吃香肠。

☆

修改翻译结果

平行語料

Quiero ir a la playa más bonita.

I want to go to the beach more pretty.

We just replace each Spanish word with the matching English word.

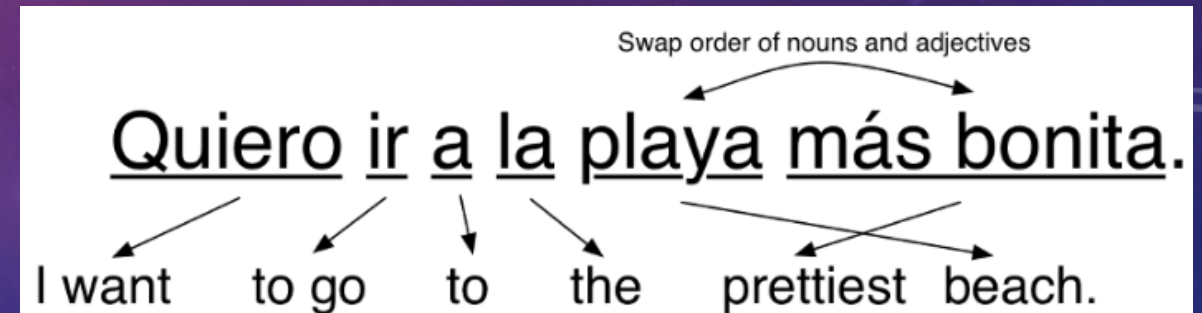


Swap order of nouns and adjectives

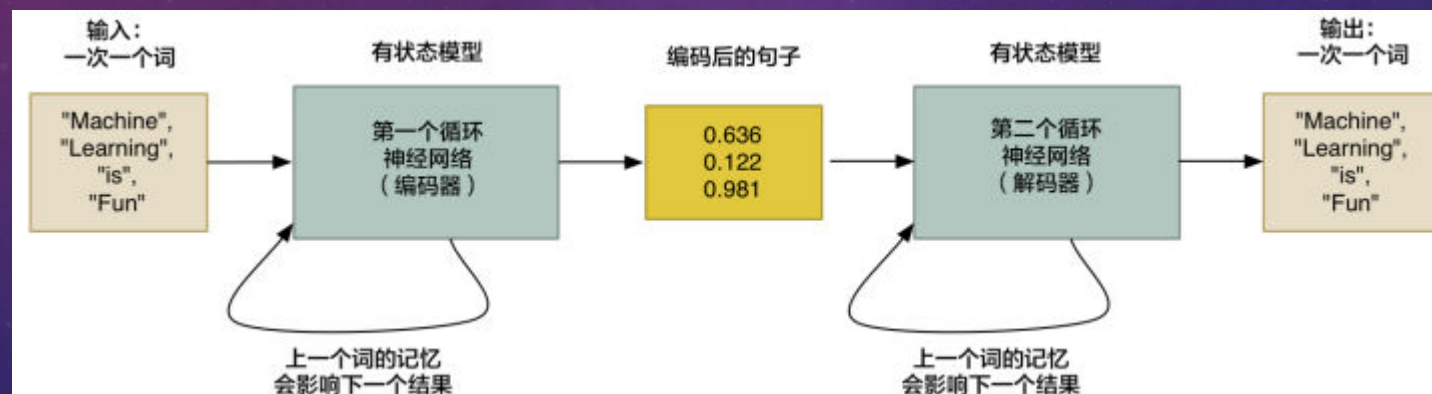
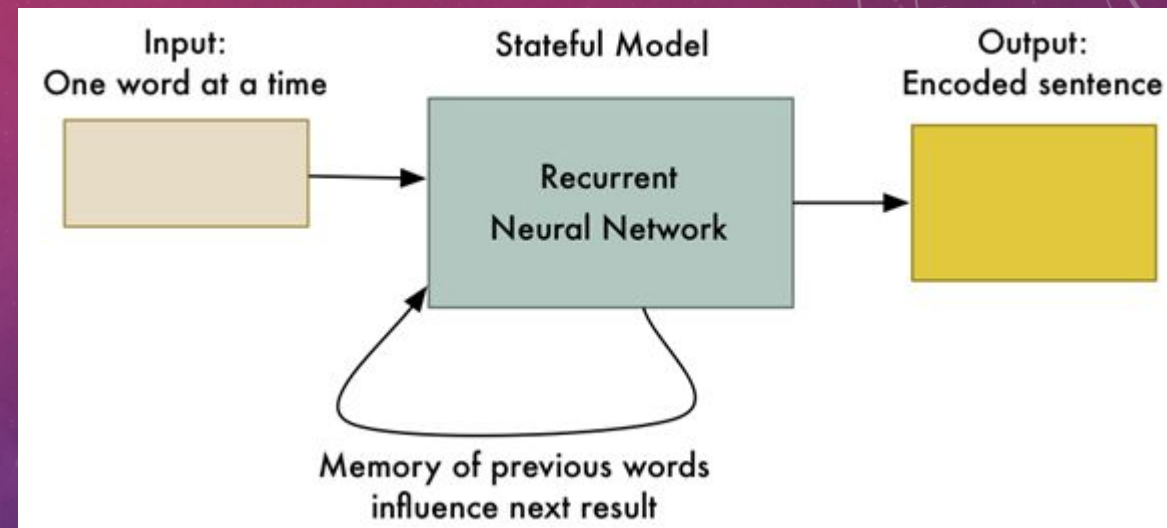
Quiero ir a la playa más bonita.

I want to go to the prettiest beach.

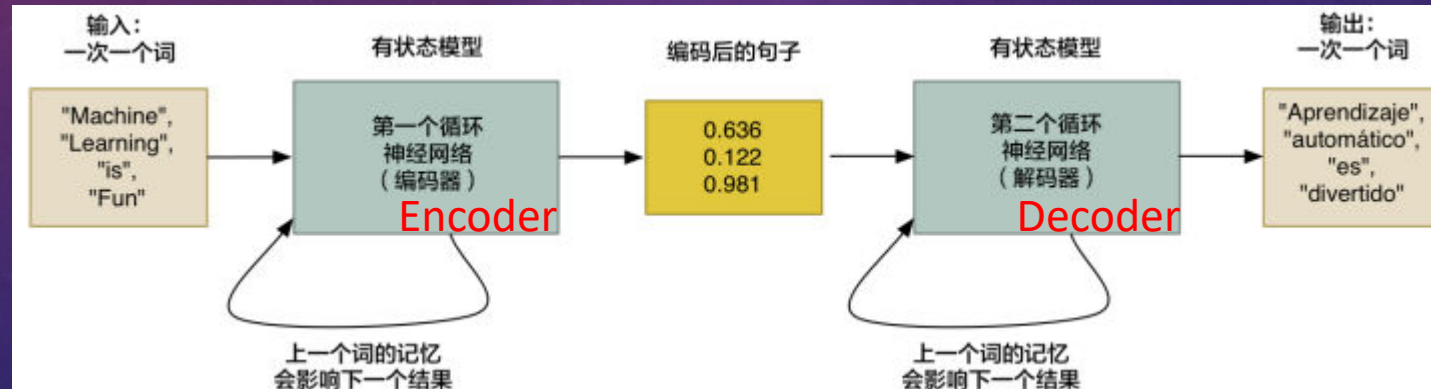
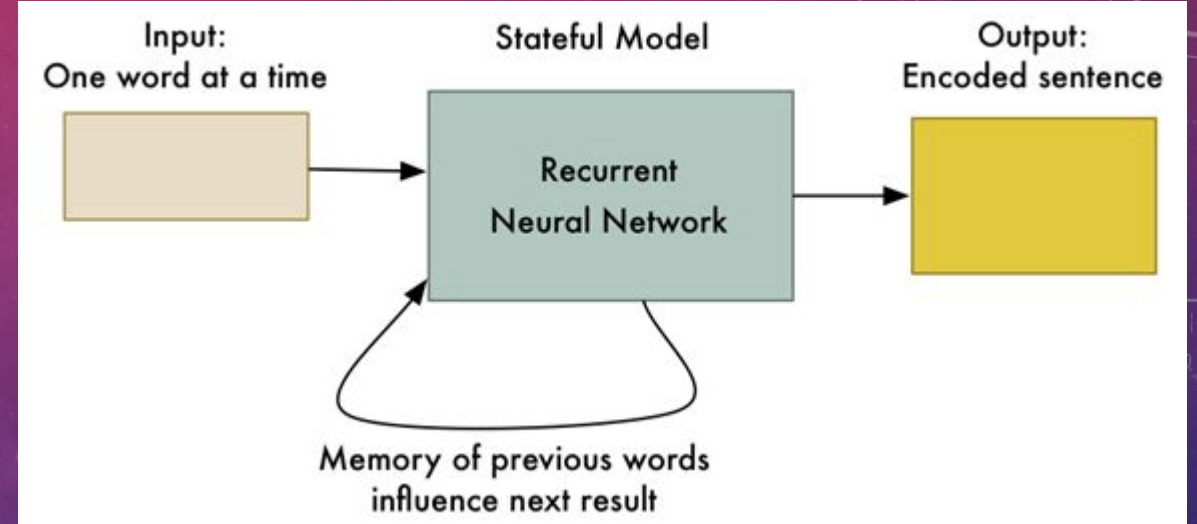
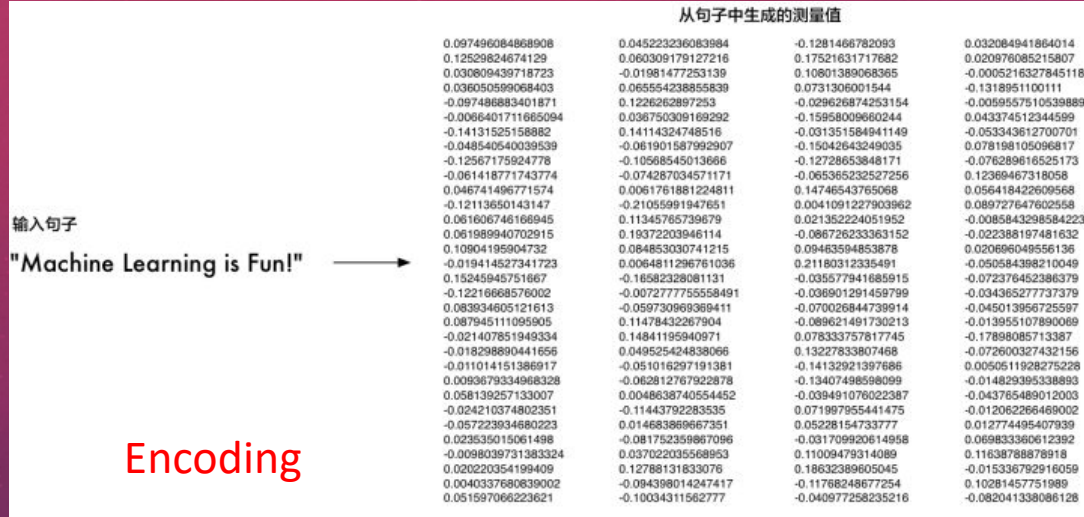
統計式機器翻譯之原理



深度學習於機器翻譯之原理



深度學習於機器翻譯之原理





自然語言理解

NATURAL LANGUAGE UNDERSTANDING

WORD-SENSE DISAMBIGUATION

- Ambiguity: a word or phrase with multiple meanings.
 1. "procure" (I will get the drinks)
 2. "become" (she got scared)
 3. "have" (I have got three dollars)
 4. "understand" (I get it)

WORDNET

WordNet Search - 3.1

- [WordNet home page](#) - [Glossary](#) - [Help](#)

Word to search for:

Display Options:

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

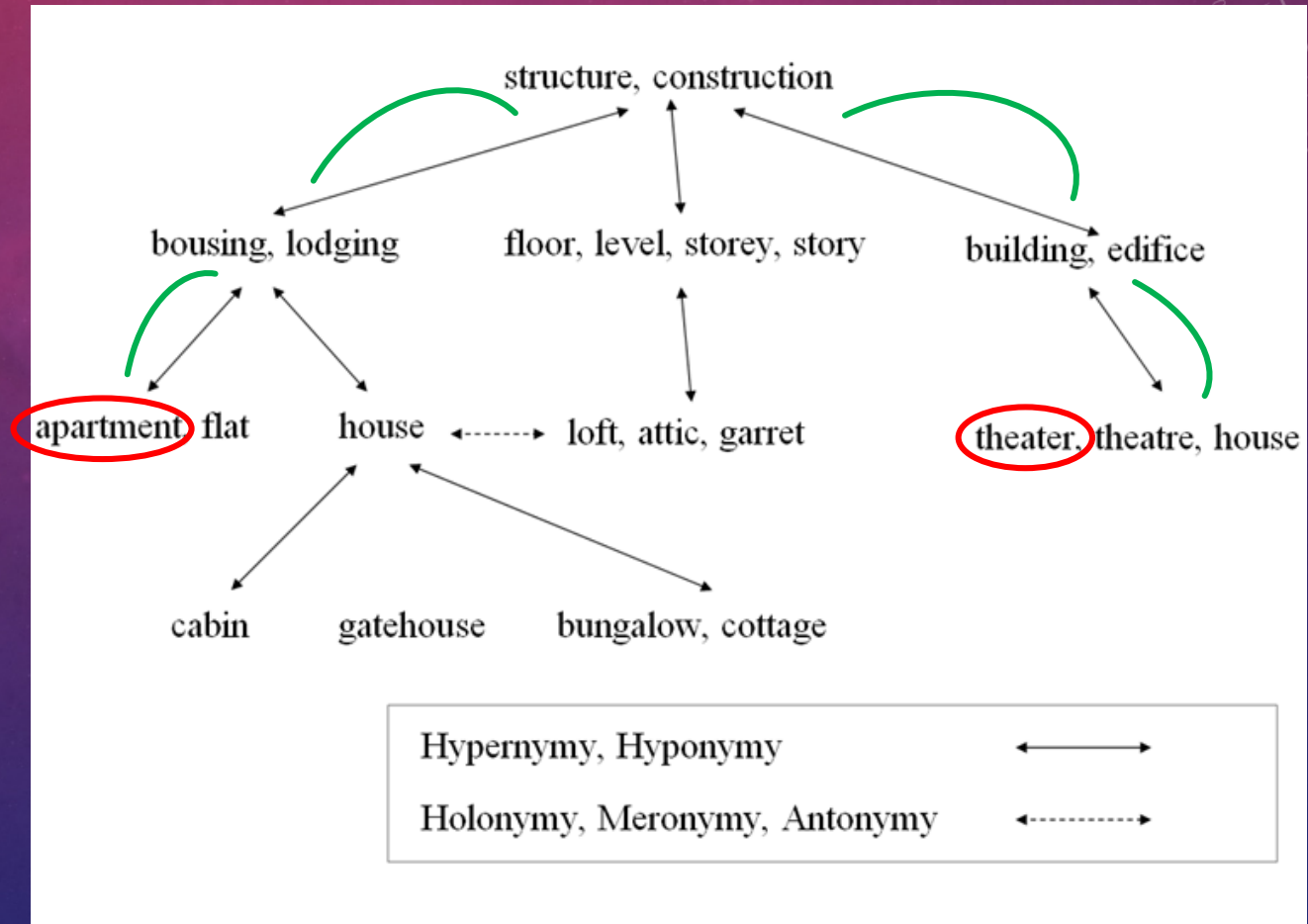
Noun

- [S:](#) [\(n\)](#) **apple** (fruit with red or yellow or green skin and sweet to tart crisp whitish flesh)
- [S:](#) [\(n\)](#) **apple**, [orchard apple tree](#), [Malus pumila](#) (native Eurasian tree widely cultivated in many varieties for its firm rounded edible fruits)

<http://wordnetweb.princeton.edu/perl/webwn>

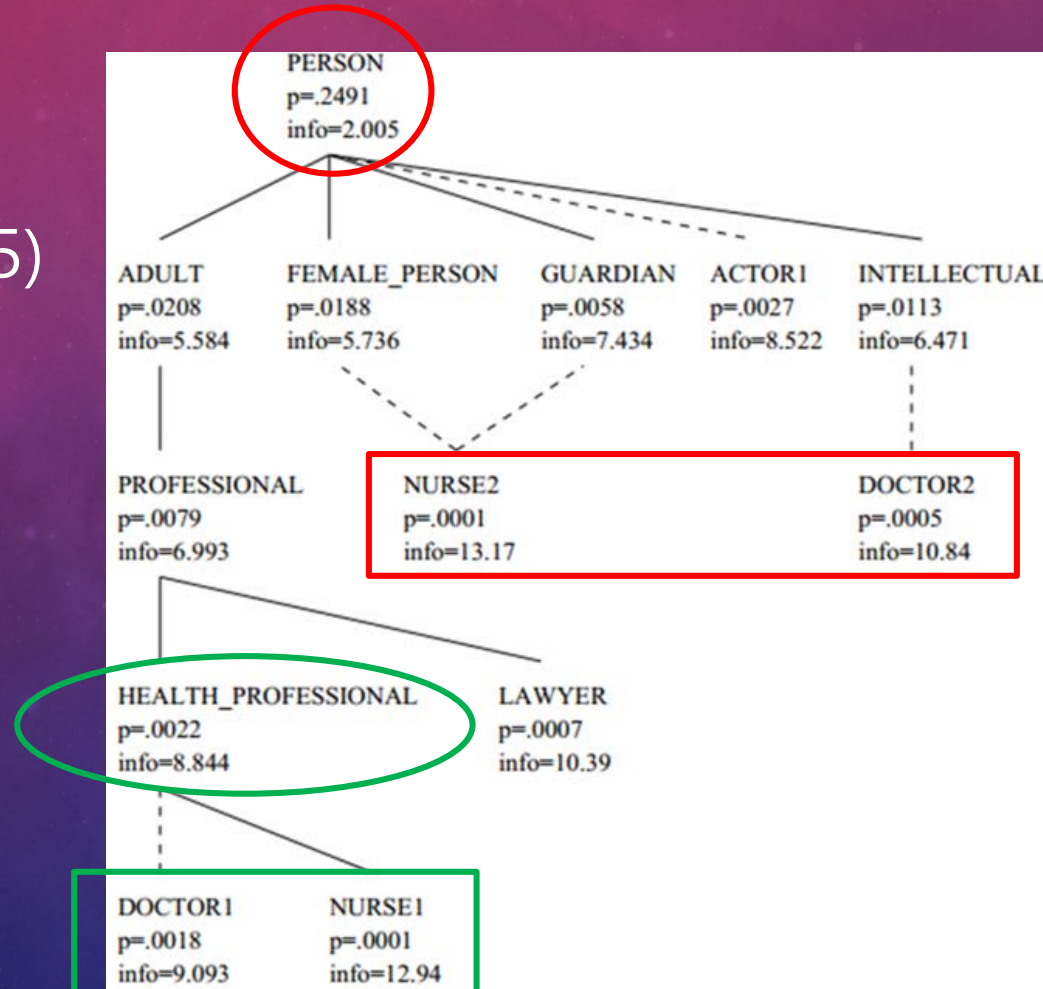
WORD-SENSE DISAMBIGUATION

- Distance-based: PATH (Rada, Mili, Bicknell, & Blettner, 1989)

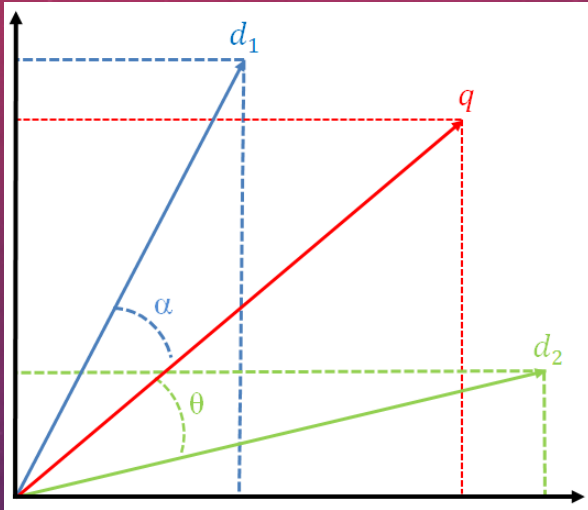


WORD-SENSE DISAMBIGUATION

- Information Content-based: RES (Resnik, 1995)



WORD-SENSE DISAMBIGUATION



- Gloss-based: VECTOR (Patwardhan, 2003)

Cute	Cunning
1. attractive especially by means of smallness or prettiness or quaintness	1. attractive especially by means of smallness or prettiness or quaintness
2. obviously contrived to charm	2. marked by skill in deception
	3. showing inventiveness and skill

廣義知網知識本體

The screenshot displays the EhowNet knowledge ontology interface. On the left, there is a sidebar with a search bar containing '成功大學' and a 'Go' button. Below the search bar, there are tabs for '查詢點', '查定義式', and '進階定義式查詢'. The main area shows a hierarchical tree structure of concepts, starting from 'TopNode' and branching into 'entity|事物', 'event|事件', 'object|物體', 'thing|萬物', 'physical|物質', 'animate|生物', 'inanimate|無生物', 'NaturalThing|天然物', 'artifact|人工物', 'clothing|衣物', 'edible|食物', 'medicine|藥物', 'addictive|嗜好物', 'building|建築物', 'house|房屋', 'facilities|設施', 'bridge|橋樑', 'route|道路', '布景|StageSettings', '看板|BulletinBoard', '陷阱|trap', '櫃臺|counter', '閘門|ThrottleValve', '槽|trough', '鞦韆|ASwing', '控制台|console', '柵欄|railings', '關隘|StrategicBorder', '籬笆|fence', '台子|platform', '機場|airport', '水庫|reservoir', '碼頭|wharf', '堤防|embankment', '棚子|shed', '軍營|MilitaryCamp', '崗哨|sentry', '教堂|church', '港口|port', '工廠|factory', '農場|farm', '車站|station', '醫院|hospital', '博物館|museum', '餐廳|restaurant', and '學校|school'.

<http://ehownet.iis.sinica.edu.tw/ehownet.php>

VECTOR REPRESENTATION

	w_1	w_2	w_3	w_{n-1}	w_n	label
D_1	0.11	0.23	0	0.57	0	0
D_2	0	0	0	0.29	0.7	1
D_3	0	0.81	0.44	0	0	0
D_4	0	0.37	0	0	0.16	1
..
D_k	1

TF-IDF

- TF: term frequency:
$$\text{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$
 - IDF: inverse document frequency:
$$\text{idf}_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$
- where:
- $|D|$: total number of documents in the corpus
 - $|\{j : t_i \in d_j\}|$: number of documents where term t_i appears

Then:

- $$\text{tfidf}_{i,j} = \text{tf}_{i,j} \times \text{idf}_i$$

Document 1		Document 2	
Term	Term Count	Term	Term Count
this	1	this	1
is	1	is	1
a	2	another	2
sample	1	example	3

- The calculation of tf-idf for the term "this" is performed as follows:

$$\begin{aligned} \text{tf}(\text{"this"}, d_1) &= \frac{1}{5} = 0.2 \\ \text{tf}(\text{"this"}, d_2) &= \frac{1}{7} \approx 0.14 \end{aligned}$$

$$\text{idf}(\text{"this"}, D) = \log\left(\frac{2}{2}\right) = 0$$

- So tf-idf is zero for the word "this", which implies that the word is not very informative as it appears in all documents.

$$\begin{aligned} \text{tfidf}(\text{"this"}, d_1) &= 0.2 \times 0 = 0 \\ \text{tfidf}(\text{"this"}, d_2) &= 0.14 \times 0 = 0 \end{aligned}$$

Document 1		Document 2	
Term	Term Count	Term	Term Count
this	1	this	1
is	1	is	1
a	2	another	2
sample	1	example	3

- A slightly more interesting example arises from the word "example", which occurs three times only in the second document:

$$\text{tf}(\text{"example"}, d_1) = \frac{0}{5} = 0$$

$$\text{tf}(\text{"example"}, d_2) = \frac{3}{7} \approx 0.429$$

$$\text{idf}(\text{"example"}, D) = \log\left(\frac{2}{1}\right) = 0.301$$

$$\begin{aligned} \text{tfidf}(\text{"example"}, d_1) &= \text{tf}(\text{"example"}, d_1) \times \text{idf}(\text{"example"}, D) = 0 \times 0.301 = 0 \\ \text{tfidf}(\text{"example"}, d_2) &= \text{tf}(\text{"example"}, d_2) \times \text{idf}(\text{"example"}, D) = 0.429 \times 0.301 \approx 0.13 \end{aligned}$$

潛藏語意分析(LSA)

- 奇異值分解
 - Singular Value Decomposition (SVD)

Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

=

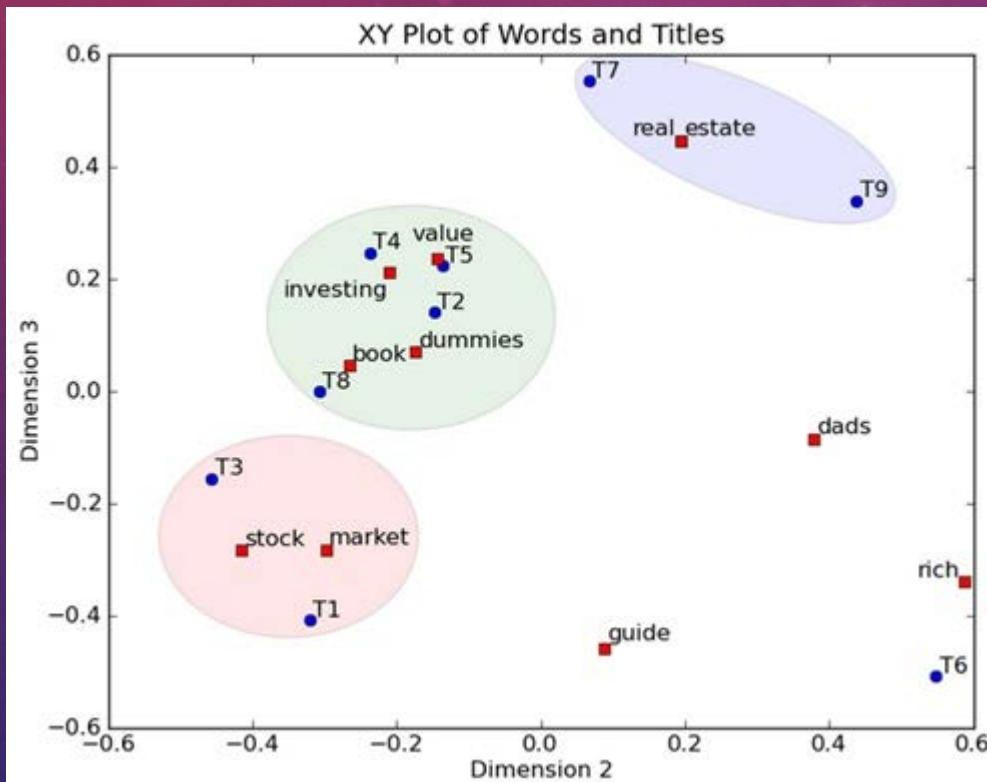
book	0.15	-0.27	0.04
dads	0.24	0.38	-0.09
dummies	0.13	-0.17	0.07
estate	0.18	0.19	0.45
guide	0.22	0.09	-0.46
investing	0.74	-0.21	0.21
market	0.18	-0.30	-0.28
real	0.18	0.19	0.45
rich	0.36	0.59	-0.34
stock	0.25	-0.42	-0.28
value	0.12	-0.14	0.23

3.91	0	0
0	2.61	0
0	0	2.00

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.35	0.22	0.34	0.26	0.22	0.49	0.28	0.29	0.44
-0.32	-0.15	-0.46	-0.24	-0.14	0.55	0.07	-0.31	0.44
-0.41	0.14	-0.16	0.25	0.22	-0.51	0.55	0.00	0.34

潛藏語意分析(LSA)

- 文件分類/主題探勘
- 語意分析



Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

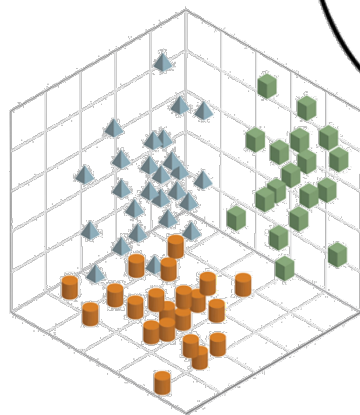
文字檔案

Input:
one document



Lorem ipsum dolor
sit amet, consete-
tur adipisicing elit,
sed diam nonumy
aliquid tempor
invidunt ut labore
et dolore magna
aliquam erat, sed
diam voluptua. At
vero eose et

word
vectors



word2vec

將被拆解成多個字元

Model:



vector space

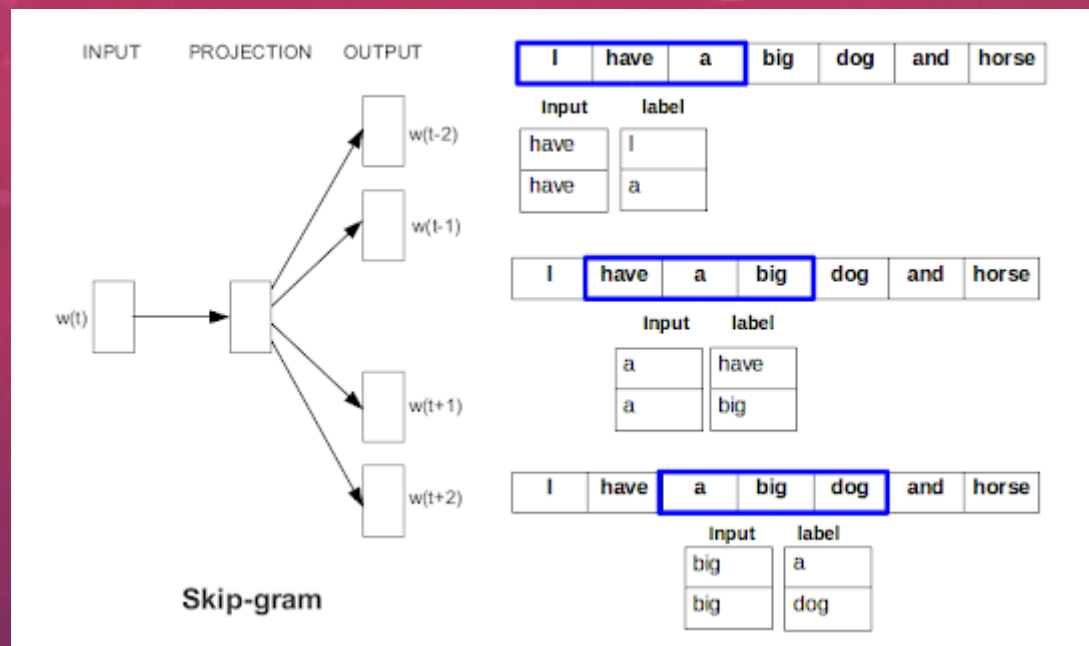
解析成多元維度的向量

透過向量比對
找出相似的資料

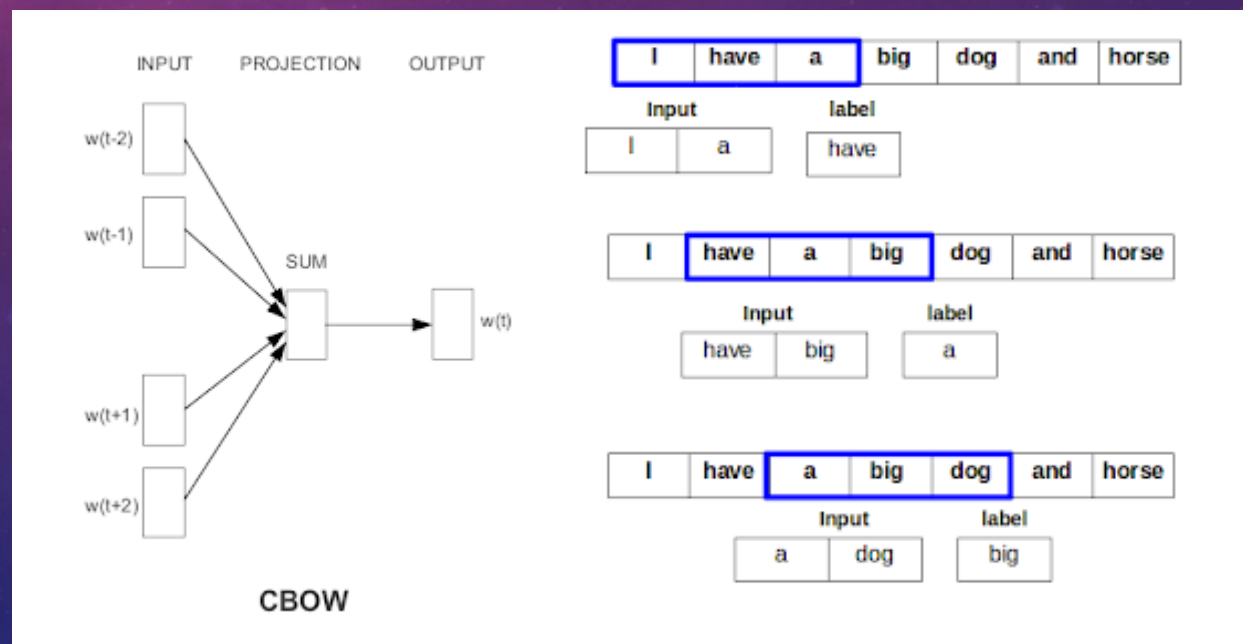
most_similar('france'):

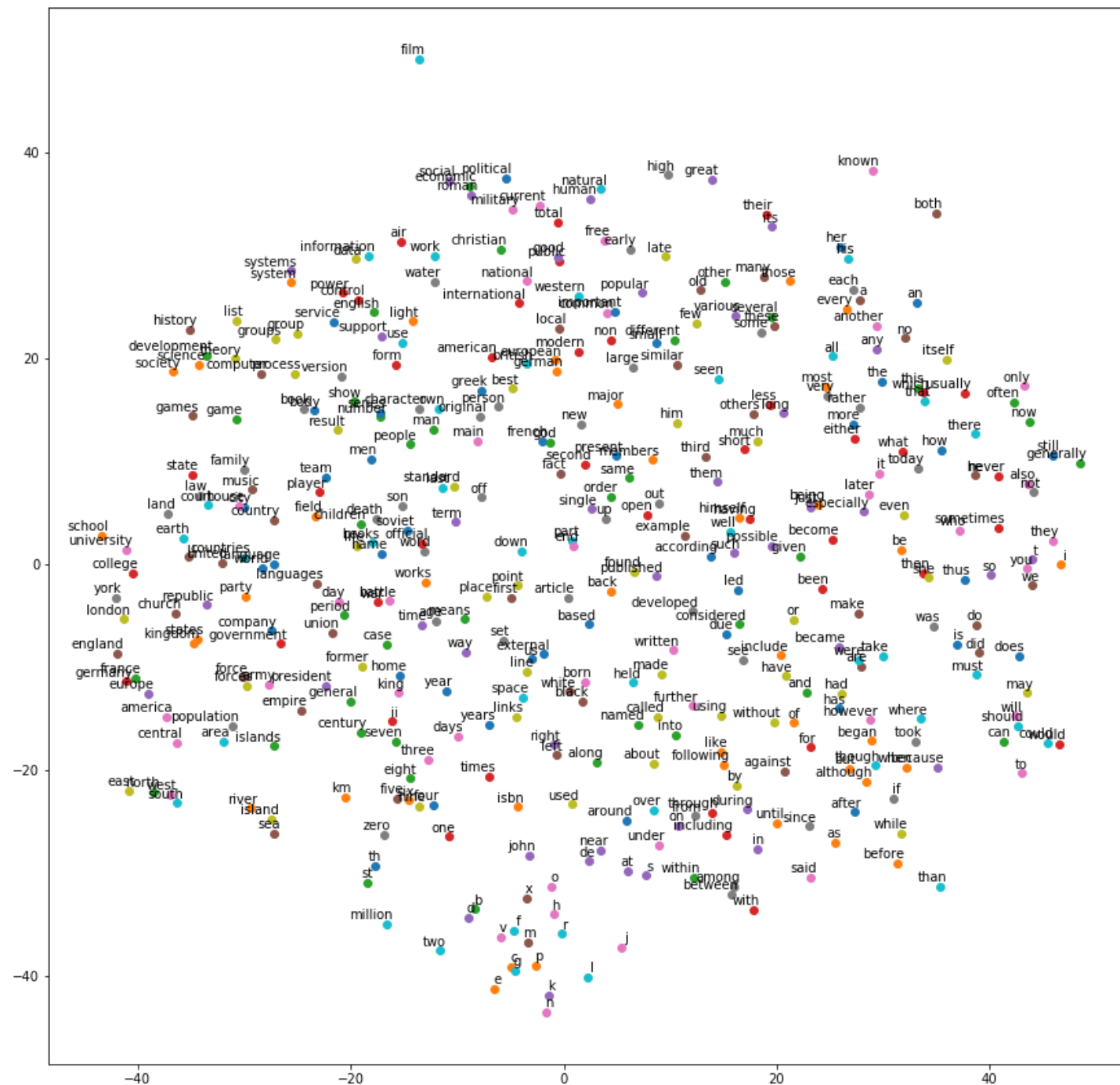
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130

highest cosine
distance values
in vector space
of the nearest
words



<http://zongsoftwarenote.blogspot.com/2017/04/word2vec-model-introduction-skip-gram.html>





語法分析

SYNTACTIC PARSING

STANFORD PARSER

A screenshot of the Stanford Parser website. The header features the Stanford NLP logo and the text 'The Stanford Natural Language Processing Group'. Navigation links include 'people', 'publications', 'research blog', 'software', 'teaching', and 'local'. A dark red banner contains the text 'Software > Stanford Parser'. Below this, the title 'The Stanford Parser: A statistical parser' is centered. A horizontal menu lists links: 'About | Citing | Questions | Download | Included Tools | Extensions | Release history | Sample output | Online | FAQ'. The 'About' section is active, showing a paragraph about natural language parsers and their development. Below this is the 'Package contents' section, which describes the Java implementation and its features. The final paragraph details the probabilistic parser's architecture and its GUI.

The Stanford Natural Language Processing Group

people publications research blog software teaching local

Software > Stanford Parser

The Stanford Parser: A statistical parser

[About](#) | [Citing](#) | [Questions](#) | [Download](#) | [Included Tools](#) | [Extensions](#) | [Release history](#) | [Sample output](#) | [Online](#) | [FAQ](#)

About

A natural language parser is a program that works out the grammatical **structure of sentences**, for instance, which groups of words go together (as "phrases") and which words are the **subject** or **object** of a verb. Probabilistic parsers use knowledge of language gained from hand-parsed sentences to try to produce the *most likely* analysis of new sentences. These statistical parsers still make some mistakes, but commonly work rather well. Their development was one of the biggest breakthroughs in natural language processing in the 1990s. You can try out our parser online.

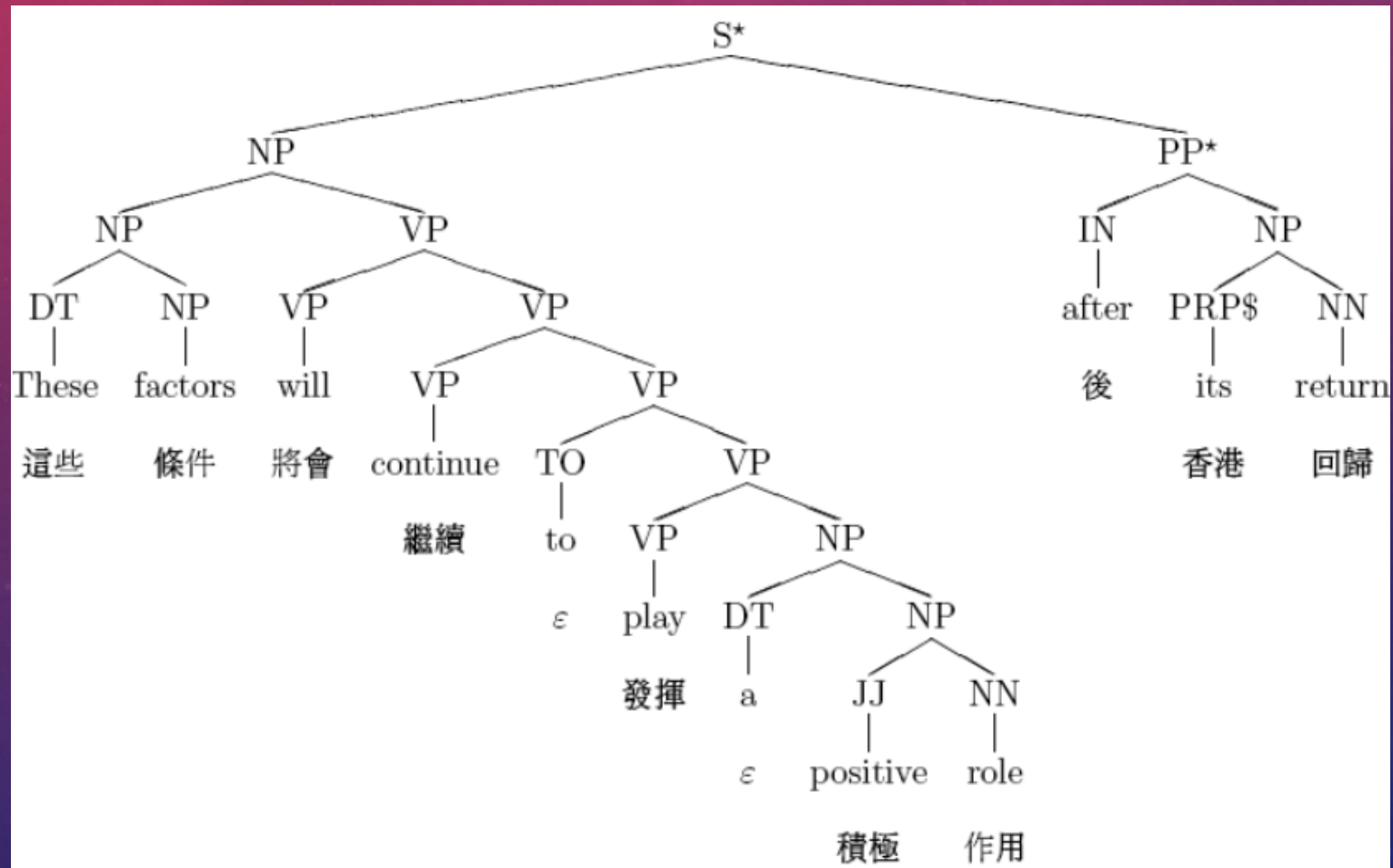
Package contents

This package is a Java implementation of probabilistic natural language parsers, both highly optimized PCFG and lexicalized dependency parsers, and a lexicalized PCFG parser. The original version of this parser was mainly written by Dan Klein, with support code and linguistic grammar development by Christopher Manning. Extensive additional work (internationalization and language-specific modeling, flexible input/output, grammar compaction, lattice parsing, *k*-best parsing, typed dependencies output, user support, etc.) has been done by Roger Levy, Christopher Manning, Teg Grenager, Galen Andrew, Marie-Catherine de Marneffe, Bill MacCartney, Anna Rafferty, Spence Green, Huihsin Tseng, Pi-Chuan Chang, Wolfgang Maier, and Jenny Finkel.

The lexicalized probabilistic parser implements a factored product model, with separate PCFG phrase structure and lexical dependency experts, whose preferences are combined by efficient exact inference, using an A* algorithm. Or the software can be used simply as an accurate unlexicalized stochastic context-free grammar parser. Either of these yields a good performance statistical parsing system. A GUI is provided for viewing the phrase structure tree output of the parser.

- <http://nlp.stanford.edu:8080/parser/>

PARSE TREE



中研院

傅達仁(Nb) 今(Nd) 將(D) 執行(VC) 安樂死(Na) , (COMMACATEGORY)

卻(D) 突然(D) 爆出(VJ) 自己(Nh) 20(Neu) 年前(Nd) 遭(P) 緯來(Nb) 體育台(Nc) 封殺(VC) , (COMMACATEGORY)

他(Nh) 不(D) 懂(VK) 自己(Nh) 哪裡(D) 得罪(VC) 到(P) 電視台(Nc) 。(PERIODCATEGORY)

實體辨識

傅達仁.PERSON今將執行安樂死，卻突然爆出自己20年前遭緯來體育台封殺，他不懂自己哪裡得罪到電視台。

指代消解

傅達仁今將執行安樂死，**NULL**傅達仁卻突然爆出自己20年前遭緯來體育台封殺，**他**傅達仁不懂自己哪裡得罪到電視台。

☒ show all ☐ show phrase head ☐ show word head

agent(執行_VC2)=傅達仁_NP
time(執行_VC2)=今_Ndabd
time(執行_VC2)=將_Dd
goal(執行_VC2)=安樂死_NP
theme(爆出_VJ3)=傅達仁_Nba
evaluation(爆出_VJ3)=卻_Dbb
time(爆出_VJ3)=突然_Dd
range(爆出_VJ3)=自己20年前遭緯來_NP
complement(爆出_VJ3)=來體育台封殺_VP
experiencer(懂_VK1)=傅達仁_NP
negation(懂_VK1)=不_Dc
goal(懂_VK1)=自己哪裡得罪到電視台_S

agent(執行_VC2)=theme(爆出_VJ3), 1
agent(執行_VC2)=experiencer(懂_VK1), 1
theme(爆出_VJ3)=experiencer(懂_VK1), 1

FUDANNLP

```
D:\fnlp>java -Xmx1024m -Dfile.encoding=UTF-8 -classpath "fnlp-core/target/fnlp-core-2.1-SNAPSHOT.jar;libs/trove-3.1a1.jar;libs/commons-cli-1.4.jar" org.fnlp.nlp.cn.tag.POSTagger -s models/seg.m models/pos.m "周杰伦出生于台湾，生日为79年1月18日，他曾经的绯闻女友是蔡依林。"
```

周杰伦/人名 出生于/动词 台湾/地名， /名词 生日/名词 为/介词 79年/时间短语 1月/时间短语 18日/时间短语， /动词 他/人称代词 曾经/形容词 的/结构助词 绯闻/名词 女友/名词 是/动词 蔡依林/人名 。/标点

http://blog.csdn.net/hhu_lyc

```
D:\fnlp>java -Xmx1024m -Dfile.encoding=UTF-8 -classpath "fnlp-core/target/fnlp-core-2.1-SNAPSHOT.jar;libs/trove-3.1a1.jar;libs/commons-cli-1.4.jar" org.fnlp.nlp.cn.tag.NERTagger -s models/seg.m models/pos.m "詹姆斯·默多克和丽贝卡·布鲁克斯鲁珀特·默多克旗下的美国小报《纽约邮报》的职员被公司律师告知，保存任何也许与电话窃听及贿赂有关的文件。"
```

<美国=地名, 纽约=地名, 詹姆斯·默多克=人名, 鲁珀特·默多克=人名, 丽贝卡·布鲁克斯=人名>

http://blog.csdn.net/hhu_lyc

復旦NLP - 簡體中文剖析工具

https://blog.csdn.net/hhu_lyc/article/details/79179619

以語言學習輔助工具為例

Collocation online suggestion v1.0
英語搭配詞線上檢索系統

[介紹](#) [常用搭配詞查詢](#) [整句搭配詞查詢與推薦](#)

整句搭配詞查詢與推薦

輸入句子：

清除

送出

輸入的句子為
We commonly use a small cell for medical research.

副詞修飾(V/Adv/Adj組合)
commonly + V/Adv/Adj

#	collocation	freq(%)
1	commonly use	46.5
2	commonly used	4.7
3	commonly find	4.4
4	commonly know	3.3
5	commonly employ	2.4
6	commonly refer	2.2
7	commonly observe	1.9
8	commonly report	1.9
9	commonly encounter	1.4
10	commonly available	1.3

commonly與use的搭配字同義組合
commonly + 搭配同義字

#	collocation	freq(%)	
1	commonly use	46.5	
2	commonly employ	2.4	
3	commonly apply	0.5	

同義詞搭配詞組搜尋結果
commonly的同義字 + use 的同義字

#	collocation	count	
1	commonly use	296	
2	often use	140	
3	frequently use	68	
4	commonly employ	15	
5	frequently employ	9	
6	often employ	6	
7	frequently apply	5	
8	repeatedly use	5	
9	routinely use	5	
10	frequently utilize	4	
11	routinely employ	3	
12	commonly apply	3	

查詢總時間:0.52sec

以語言學習輔助工具為例

	computer	data	pinch	result	sugar
aprocot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$$P(x = \text{information}, y = \text{data}) = \frac{6}{19} = 0.32$$

$$P(x = \text{information}) = \frac{6 + 4 + 1}{19} = \frac{11}{19} = 0.58$$

$$P(y = \text{data}) = \frac{6 + 1}{19} = \frac{7}{19} = 0.37$$

$$\begin{aligned} & \text{pmi}(x = \text{information}, y = \text{data}) \\ &= \log \frac{P(x = \text{information}, y = \text{data})}{P(x = \text{information}) \times P(y = \text{data})} \\ &= \log 1.49 \\ &= 0.57 \end{aligned}$$

The background features a vertical gradient from red at the top to blue at the bottom, overlaid with a fine white speckle pattern. On the left side, there are several white circular and semi-circular graphic elements. A prominent circular scale with degree markings (40, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260) is visible. Other elements include concentric circles, dashed lines, and arrows, some of which are partially cut off by the frame edges.

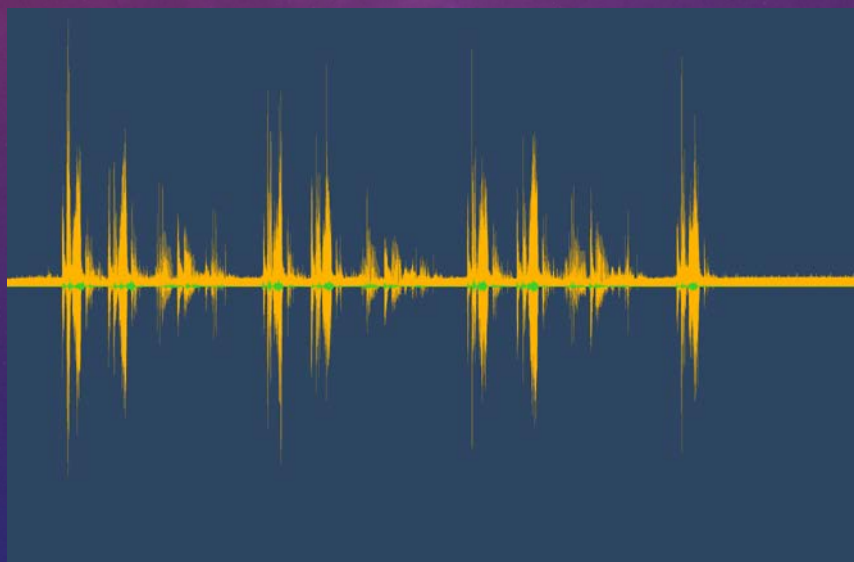
THANK YOU

AUDIO SIGNAL PROCESSING

語音處理

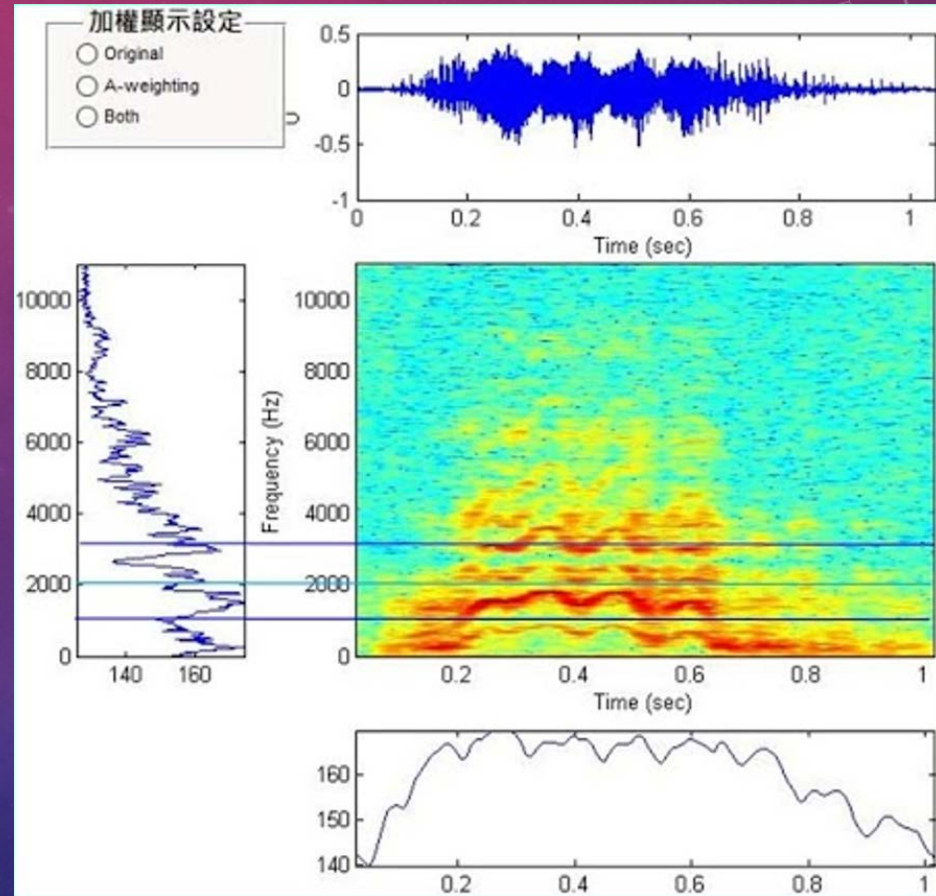
聲學特徵 (BASIC ACOUSTIC FEATURES)

- **音量(Volume/Loudness)**：聲音的大小聲(響度)，與訊號的振幅相關，可視為音頻訊號的能量或強度。
- **音高(Pitch)**：聲音的頻率，音頻訊號的振動速率，可以用基音頻率來表示。
- **音色(Timbre)**：聲音頻率的組合，表示音頻訊號中有意義的內容，為語音訊號基本週期內的波形。



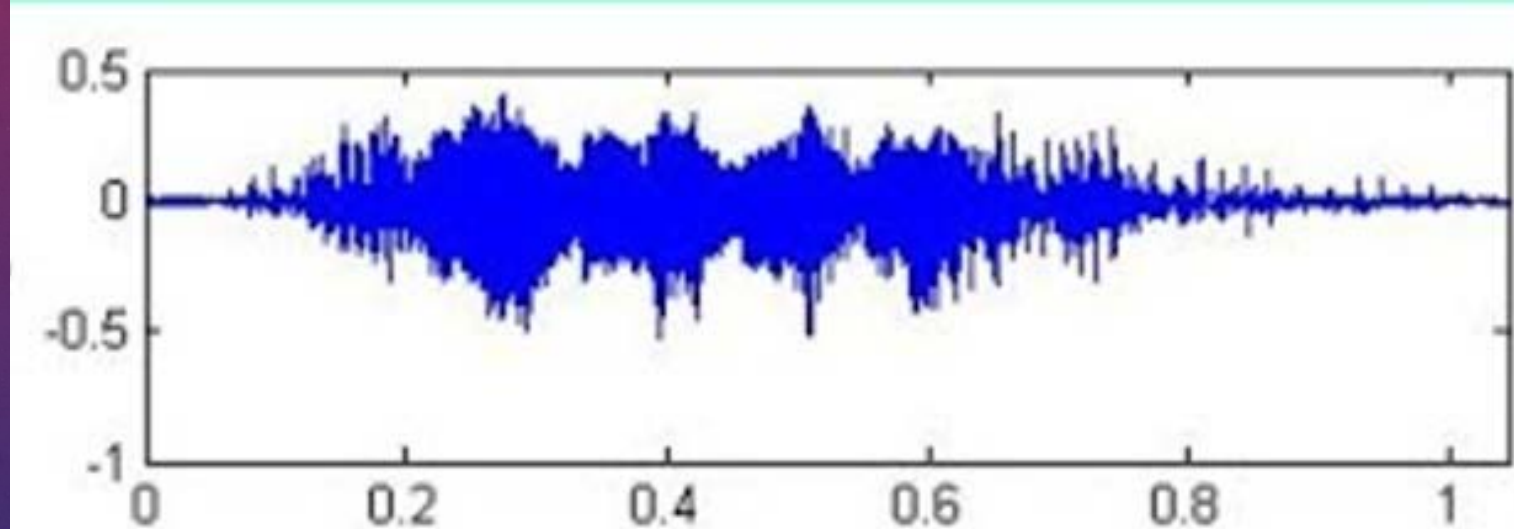
語音的特徵

- 聲音壓力時間域波形圖
(Time Waveform)
- 時間域聲音壓力位準(響度)圖
(Sound Pressure Level)
- 時頻圖
(Spectrogram)
- 頻譜圖
(Spectrum)



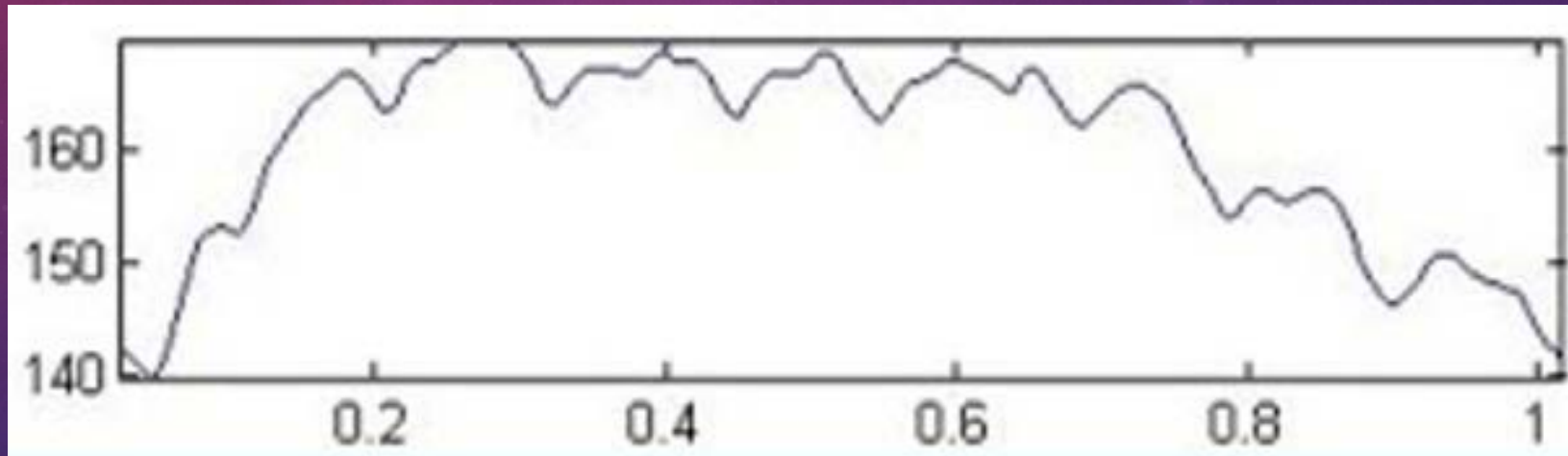
語音的特徵

- 聲音壓力時間域波形圖 (Time Waveform)：呈現所分析聲音波形的原始「時間域波形」訊號。代表的物理意義是「聲音壓力」(Sound Pressure)。此波形圖呈現了聲音的「波動」現象，不過，除了大小聲外，似乎觀察不出甚麼特別的現象。



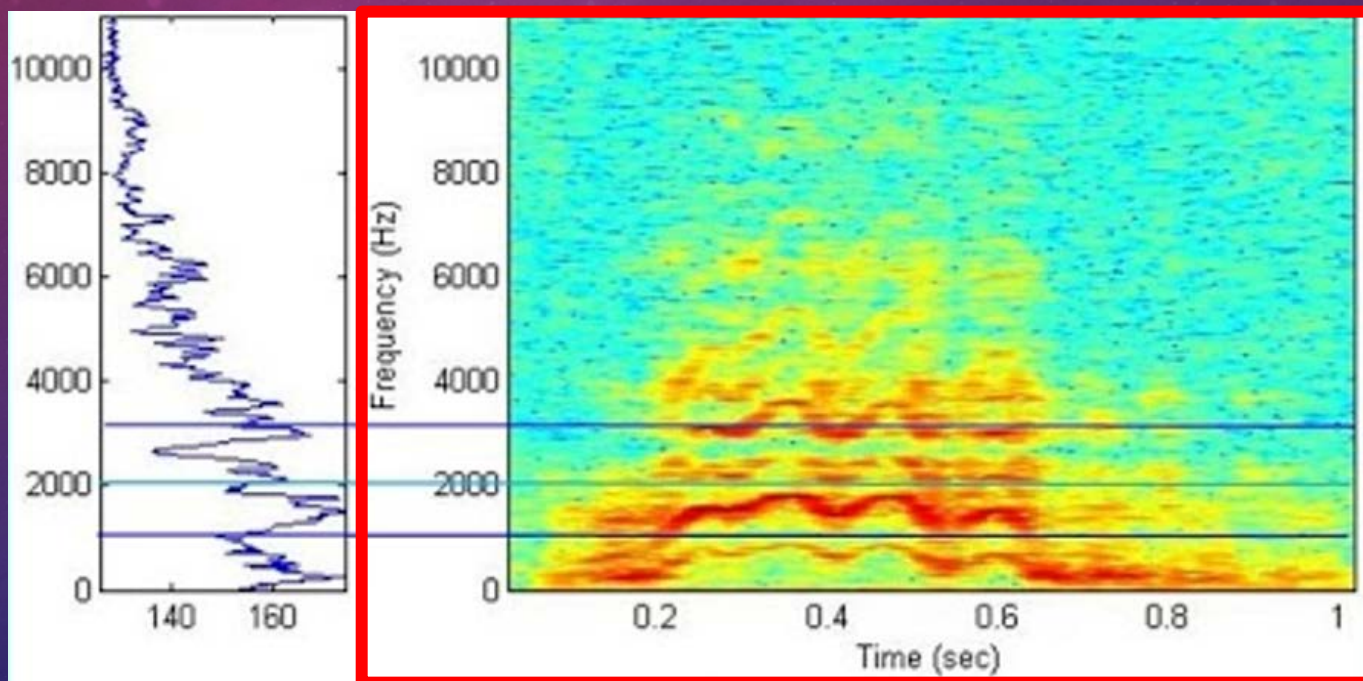
語音的特徵

- 時間域聲音壓力位準(響度)圖 (Sound Pressure Level)：可以簡稱「聲音響度圖」，呈現此聲音隨時間變化的「聲音振幅大小」(dB)，此「聲音振幅」的物理意義是「聲音壓力位準」(Sound Pressure Level)。



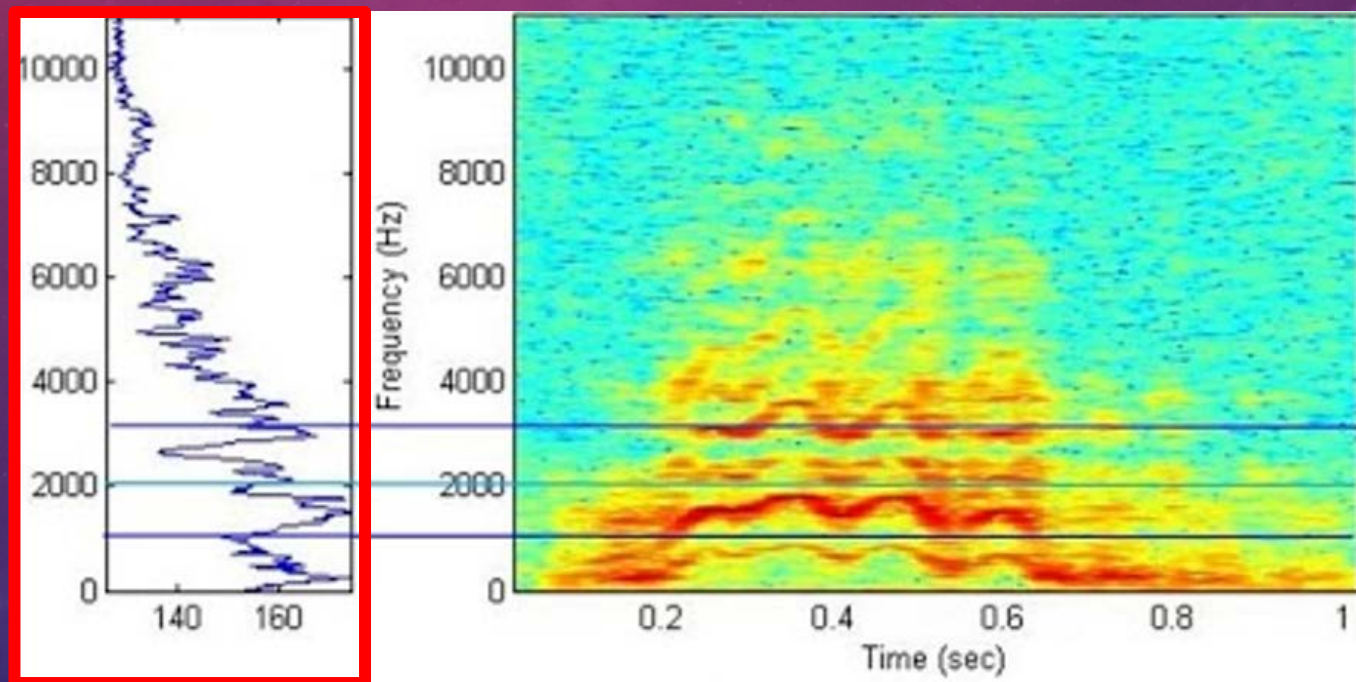
語音的特徵

- 頻譜圖 (Spectrum)：水平軸是對應「時間域波形」的時間(sec)，垂直軸是頻率(Hz)，彩色的分布圖示為聲音振幅大小(dB)。可以觀察隨時間變動時，「聲音頻譜」的分布狀態與趨勢。
- 以快速傅立葉變換 (Fast Fourier Transform, FFT) 取得。



語音的特徵

- 時頻圖 (Spectrogram)：因為解析的訊號是「聲音」，所以稱為「聲音頻譜圖」，也就是頻率與其振幅的組成。
- 以短時距傅立葉變換(Short-time Fourier transform, STFT)取得。



- 圖示的垂直軸為頻率，單位：Hz，水平軸是聲音的振幅大小，單位：dB。可以觀察「聲音頻譜」的分佈特徵。

語音的特徵

- 梅爾頻率倒譜係數 (Mel-Frequency Cepstral Coefficients, MFCC) 是一組用來建立梅爾倒頻譜 (Mel-Frequency Cepstrum, MFC) 的關鍵係數。梅爾倒頻譜最大的特色在於，於梅爾倒頻譜上的頻帶是均勻分布於梅爾刻度上的，也就是說，這樣的頻帶相較於一般所看到、線性的倒頻譜表示方法，**和人類非線性的聽覺系統更為接近**。
 1. 將一訊號進行傅利葉轉換
 2. 利用三角窗函式 (triangular overlapping window) ，將頻譜對映 (mapping) 至梅爾刻度
 3. 取對數
 4. 取離散餘弦轉換
 5. MFCC是轉換後的頻譜

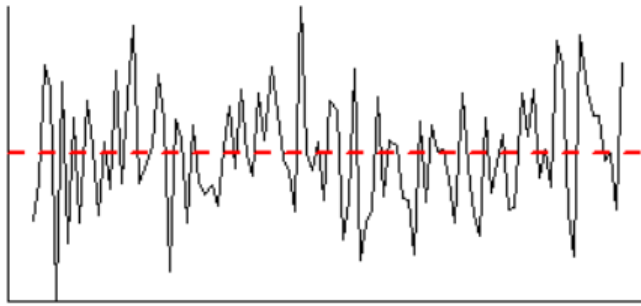


THANK YOU

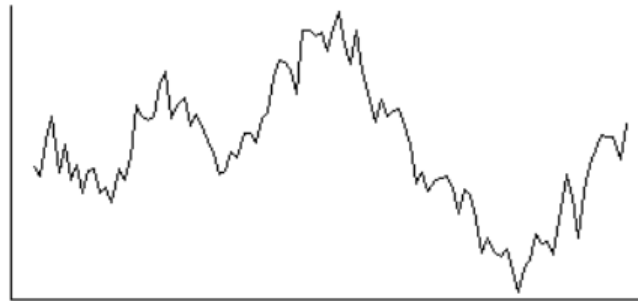
Definition of Time Series

Categories of Time Series

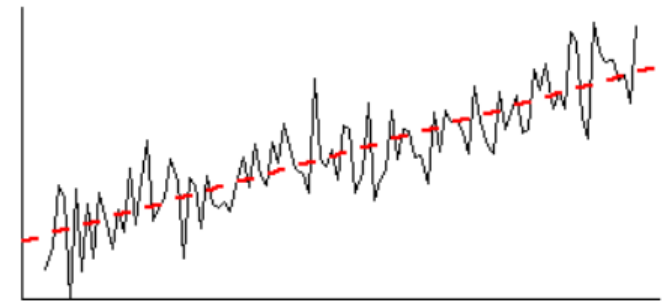
(A) 平穩型



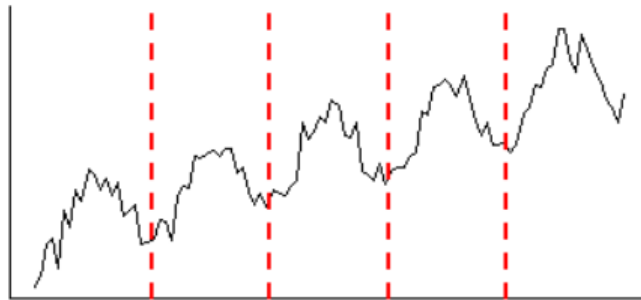
(B) 無定向型



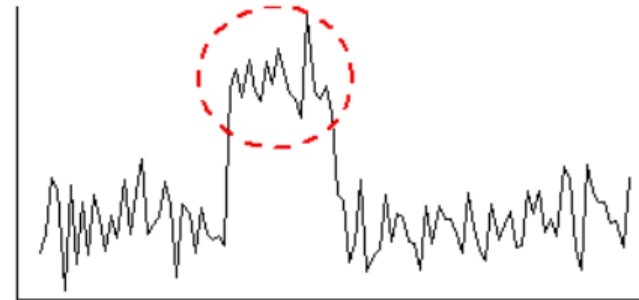
(C) 趨勢型



(D) 季節型(趨勢型)



(E) 介入事件型



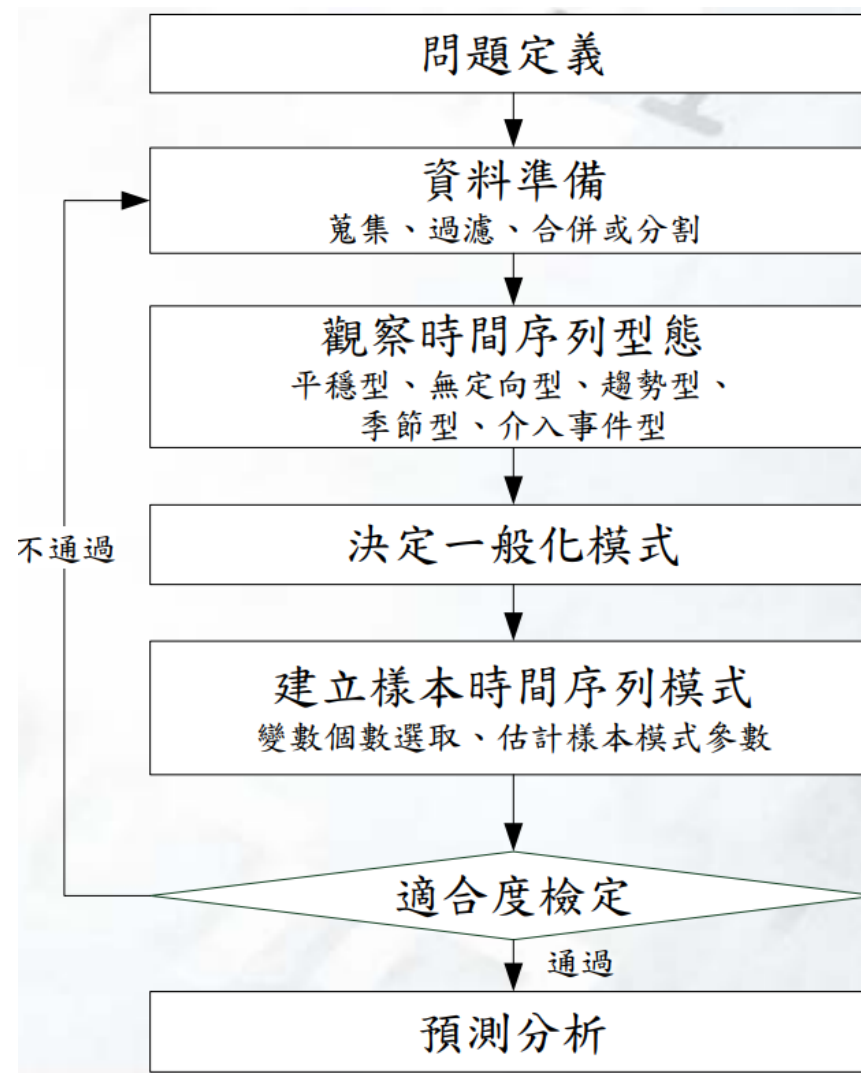
Process of Time Series Analysis

Process

➤ Analysis in Time Domain

- Autocorrelation Function
- 主要探討同一變數於不同時期的相關程度。
- 假設 $Z_i, i=1, \dots, n$ 為時間序列之 n 項觀測值，相隔 k 期的兩個觀測值的自我相關函數。

$$\rho_k = \frac{Cov(Z_t, Z_{t+k})}{\sqrt{Var(Z_t)} \times \sqrt{Var(Z_{t+k})}}$$

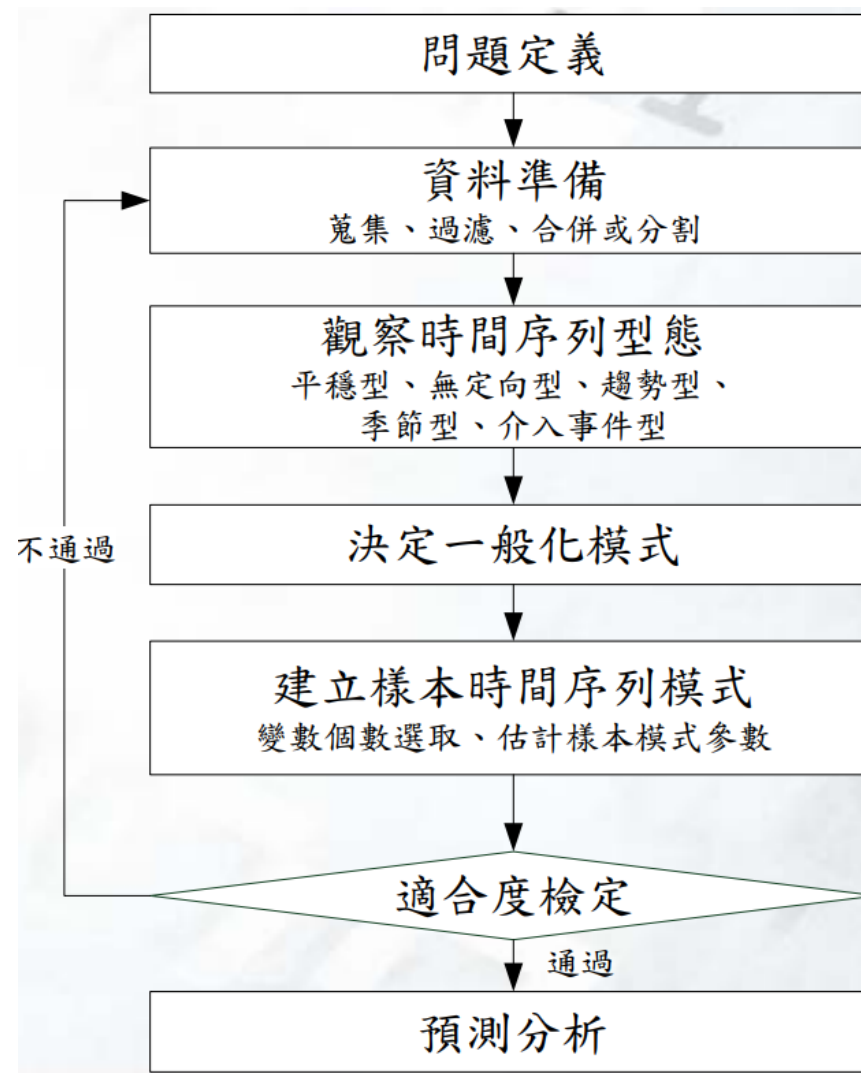


Process

➤ Analysis in Time Domain

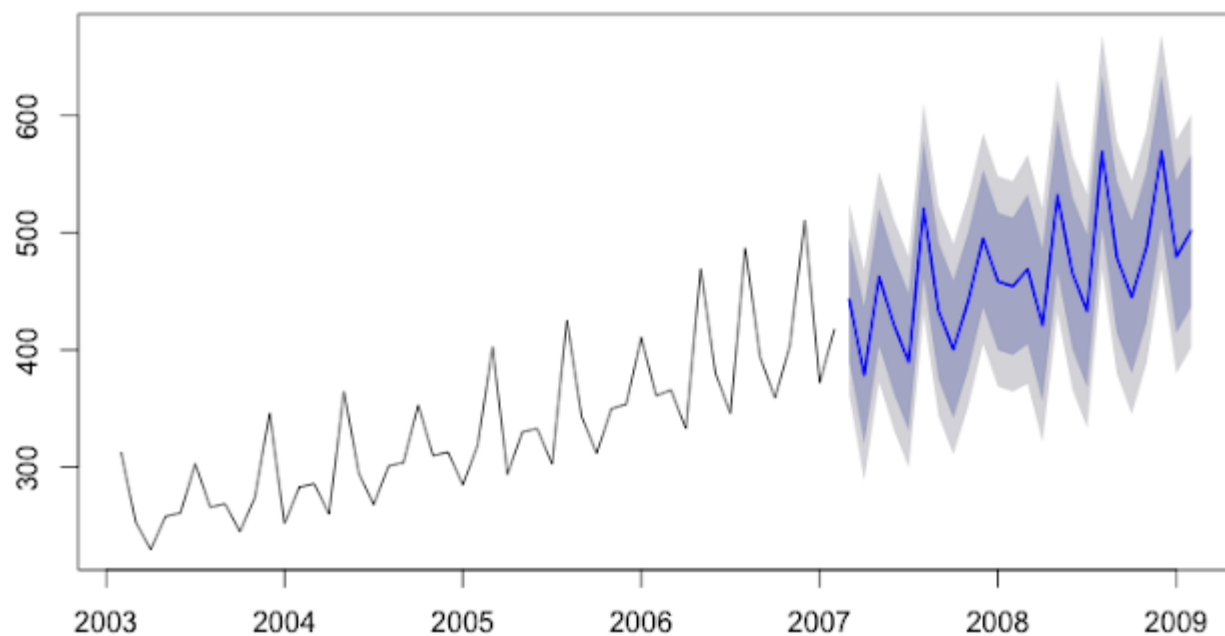
- Autocorrelation Function
- 主要探討同一變數於不同時期的相關程度。
- 假設 $Z_i, i=1, \dots, n$ 為時間序列之 n 項觀測值，相隔 k 期的兩個觀測值的自我相關函數。

$$\rho_k = \frac{Cov(Z_t, Z_{t+k})}{\sqrt{Var(Z_t)} \times \sqrt{Var(Z_{t+k})}}$$



ARIMA

Forecasts from ARIMA(0,0,1)(1,1,0)[12] with drift



- 時間序列模型，AR、MA、ARMA、ARIMA模型等基礎知識，在訊號、金融時序分析通用。

ARIMA

1. 自迴歸模型 (AR, Autoregressive Model)

- 自身迴歸用同一變數例如 x 的之前各期，亦即 x_1 至 x_{t-1} 來預測本期 x_t 的表現，並假設它們為一線性關係。

2. 移動平均模型 (MA, Moving Average Model)

- 以近期實際數值之平均做為最新預測值的參考。另外，也可以以愈近期數值乘以愈大權數的方法來估算預測值。

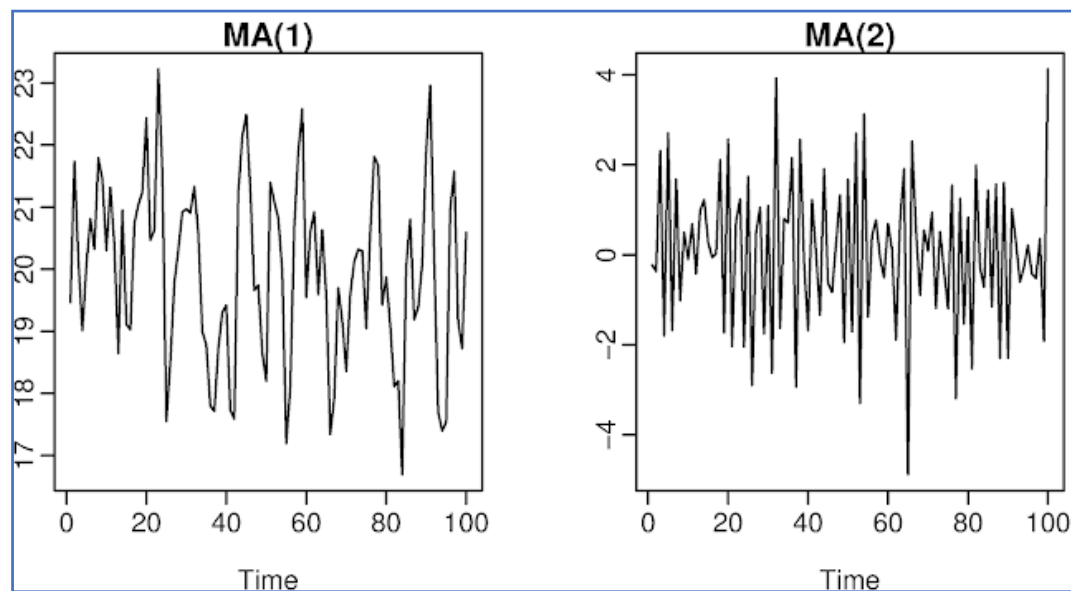
3. ARMA模型 (AR, MA 兩者的混合)

4. ARIMA (AR+MA+差分)

令 $x_k = x_0 + kh, (k = 0, 1, \dots, n)$
 $\Delta f(x_k) = f(x_{k+1}) - f(x_k)$

- 一階差分的差分為二階差分，二階差分的差分為三階差分。

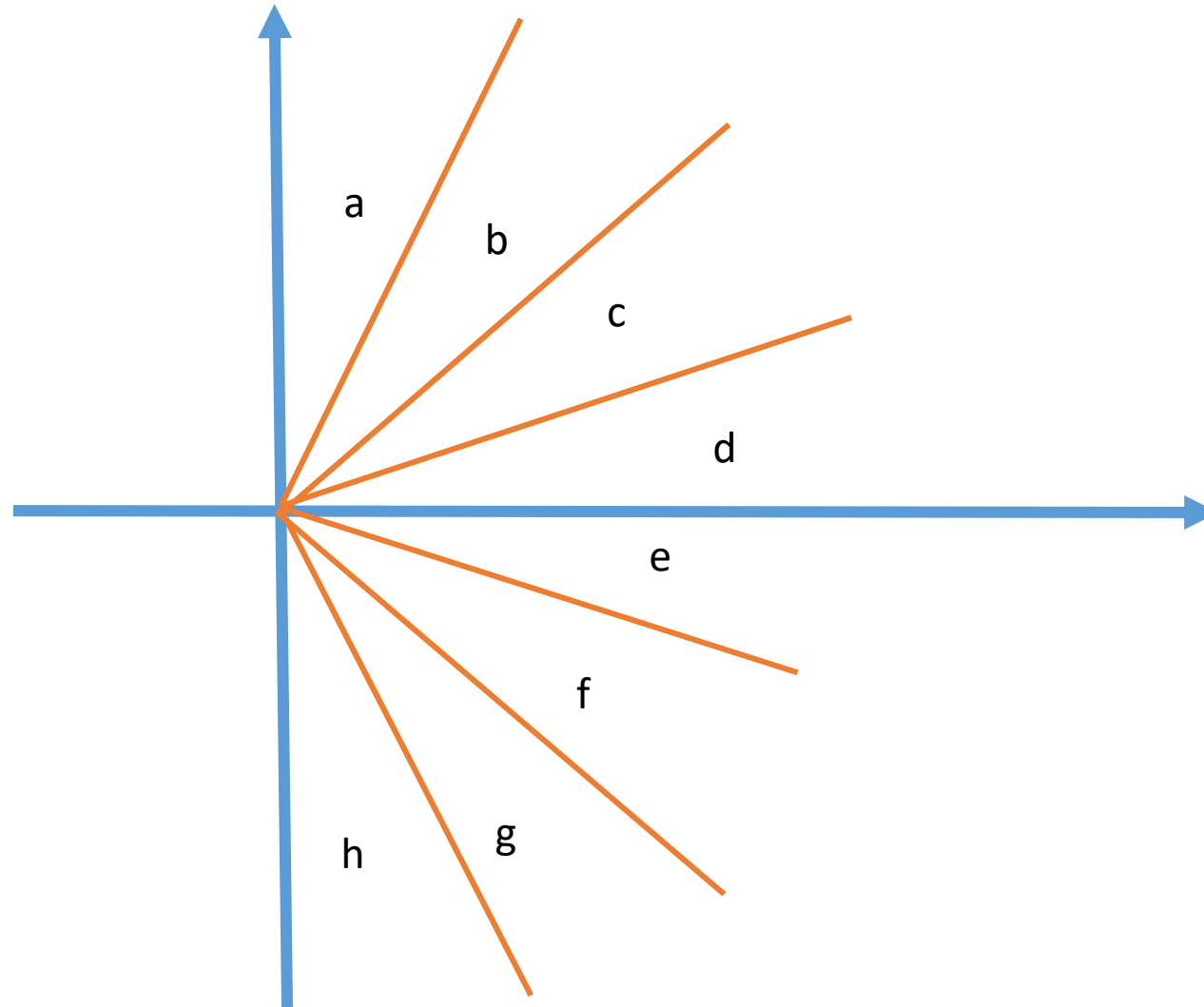
$$F_t = MA_3 = \frac{A_{t-1} + A_{t-2} + A_{t-3}}{3}$$



Time Series Encoding

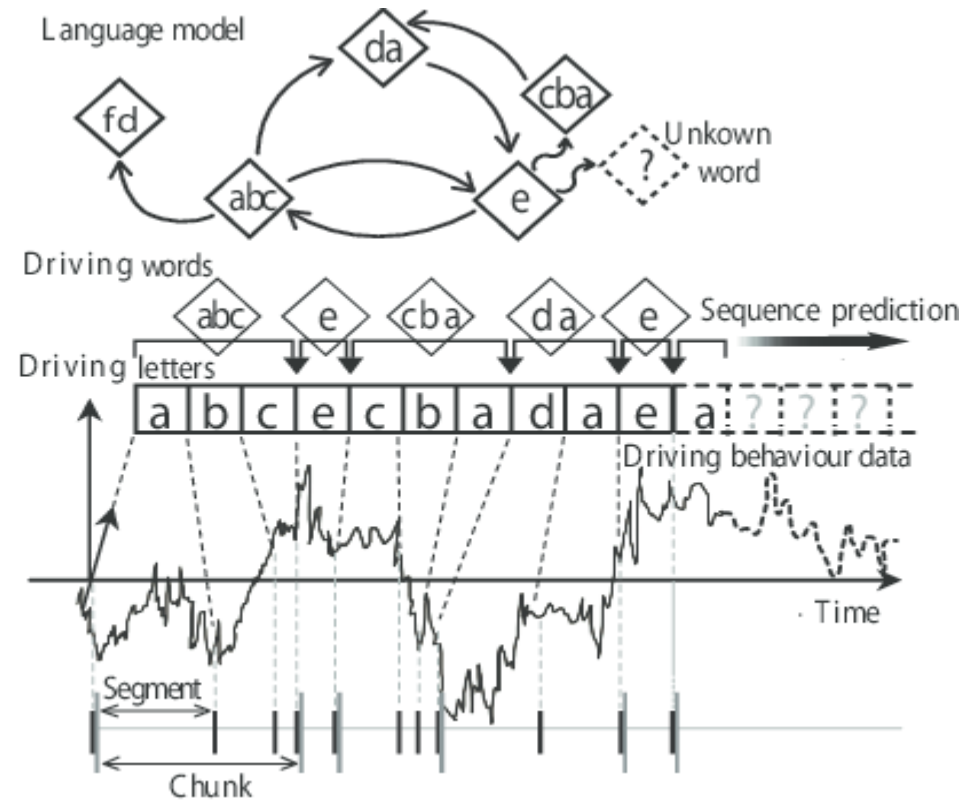
Time Series Encoding

- RNN
- LSTM
- GRU



Prediction Model

- RNN
- LSTM
- GRU



Thank you

Reference

1. https://dalab.ie.nthu.edu.tw/DMclass/file/%E8%B3%87%E6%96%99%E6%8C%96%E7%A4%A6%E8%88%87%E5%A4%A7%E6%95%B8%E6%93%9A%E5%88%86%E6%9E%90_Ch10%E6%99%82%E9%96%93%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90_2014.pdf
2. <https://mropengate.blogspot.com/2015/11/time-series-analysis-ar-ma-arma-arima.html>
3. http://web.ydu.edu.tw/~alan9956/doc100/100-02.ec/om_chap03.pdf