自然語言處理

Justin.Lee

講師介紹

李錦和

大數據與開源專案的熱好者

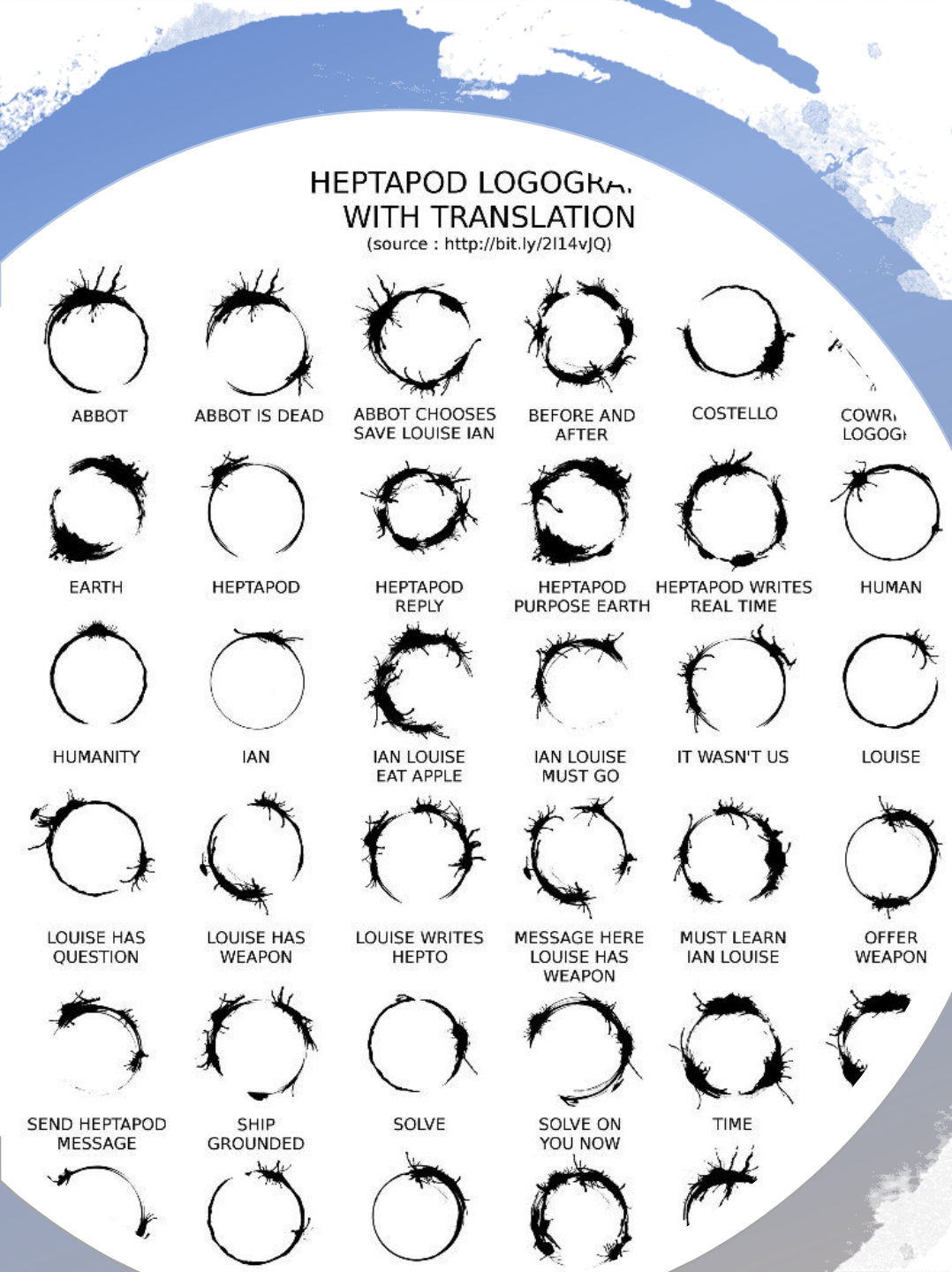
- **學　　歷**：東吳資科所
- **專長領域**：大數據平台架構研究與實作
- **專案經歷**：國內大型電商、遠傳電信與癮科技專案合作經驗
- **講師經歷**：資策會、企業內訓

課程概要

- Day1
  - Text mining 理論基礎
  - 實作：Text mining 文字雲
- Day2
  - 自然語言處理介紹
  - 實作：WordNet 初體驗
  - Word2Vec 介紹
  - 實作：Word2vec 理解內容
- Day3
  - Attention 介紹
  - 讓Attention 看懂日期
  - BERT 與 GPT-2 介紹
  - GPT-2實作

語言是武器

圖片來源：
http://www.impawards.com/2016/arrival.html



文字是思想的載體，圖象是意念聚合成像的輸出

文字是思想

來源:https://www.facebook.com/lffcalligraphy/posts/6704215397258O9/

自然語言處理

中文的博大精深

- 有關係就沒關係，沒關係就有關係
- 冬天能穿多少算多少，夏天能穿多少算多少
- 以前我喜歡一個人，現在我喜歡一個人

自然語言處理介紹

- 自然語言處理(Natural Language Processor)
  - 讓電腦了解人類語言

第一回：宴桃園豪傑三結義，斬黃巾英雄首立功

詞曰：

滾滾長江東逝水,浪花淘盡英雄。是非成敗轉頭空：青山依舊在,幾度夕陽紅。白髮漁樵江渚上

話說天下大勢：分久必合，合久必分：周末七國分爭，并入於秦。及秦滅之後，楚、漢分爭
桓帝崩，靈帝即位，大將軍竇武、太傅陳蕃，共相輔佐。時有宦官曹節等弄權，竇武、陳蕃

建寧二年四月望日，帝御溫德殿。方陞座，殿角狂風驟起，只見一條大青蛇，從梁上飛將下
海水泛溢，沿海居民，盡被大浪捲入海中。光和元年，雌雞化雄。六月朔，黑氣十餘丈，飛

帝下詔問群臣以災異之由，議郎蔡邕上疏，以為蛻墮雞化，乃婦寺干政之所致，言頗切直。
比為奸，號為「十常侍」。帝尊信張讓，呼為「阿父」，朝政日非，以致天下人心思亂，盜

時鉅鹿郡有兄弟三人：一名張角，一名張寶，一名張梁。那張角本是個不第秀才。因入山採
姓名。老人曰：「吾乃南華老仙也。」言訖，化陣清風而去。

萃取文字

盧植, 劉焉, 鄒靖, 賊眾, 雲長, 黃巾, 程遠志, 五百, 董卓, 潁川, 中郎, 身長, 封謚, 公將軍, 廣宗, 涿縣, 五萬, 桃園, 為兄, 二客

自然語言處理介紹

- 自然語言處理(Natural Language Processor)
  - 電腦了解人類語言之後...

第一回：宴桃園豪傑三結義，斬黃巾英雄首立功

詞曰：

滾滾長江東逝水,浪花淘盡英雄。是非成敗轉頭空：青山依舊在,幾度夕陽紅。白髮漁樵江渚上

話說天下大勢，分久必合，合久必分：周末七國分爭，并入於秦。及秦滅之後，楚、漢分爭
桓帝崩，靈帝即位，大將軍竇武、太傅陳蕃，共相輔佐。時有宦官曹節等弄權，竇武、陳蕃

建寧二年四月望日，帝御溫德殿。方陞座，殿角狂風驟起，只見一條大青蛇，從梁上飛將下
海水泛溢，沿海居民，盡被大浪捲入海中。光和元年，雌雞化雄。六月朔，黑氣十餘丈，飛

帝下詔問群臣以災異之由，議郎蔡邕上疏，以為蛻墮雞化，乃婦寺干政之所致，言頗切直。
比為奸，號為「十常侍」。帝尊信張讓，呼為「阿父」，朝政日非，以致天下人心思亂，盜

時鉅鹿郡有兄弟三人：一名張角，一名張寶，一名張梁。那張角本是個不第秀才。因入山採
姓名。老人曰：「吾乃南華老仙也。」言訖，化陣清風而去。

文章生成

盧植, 劉焉, 鄒靖, 賊眾, 雲長, 黃巾, 程遠志, 五百, 董卓, 潁川, 中郎, 身長, 封謚, 公將軍, 廣宗, 涿縣, 五萬, 桃園, 為兄, 二客

自然語言處理介紹

NLP

- Processing: text into structure data

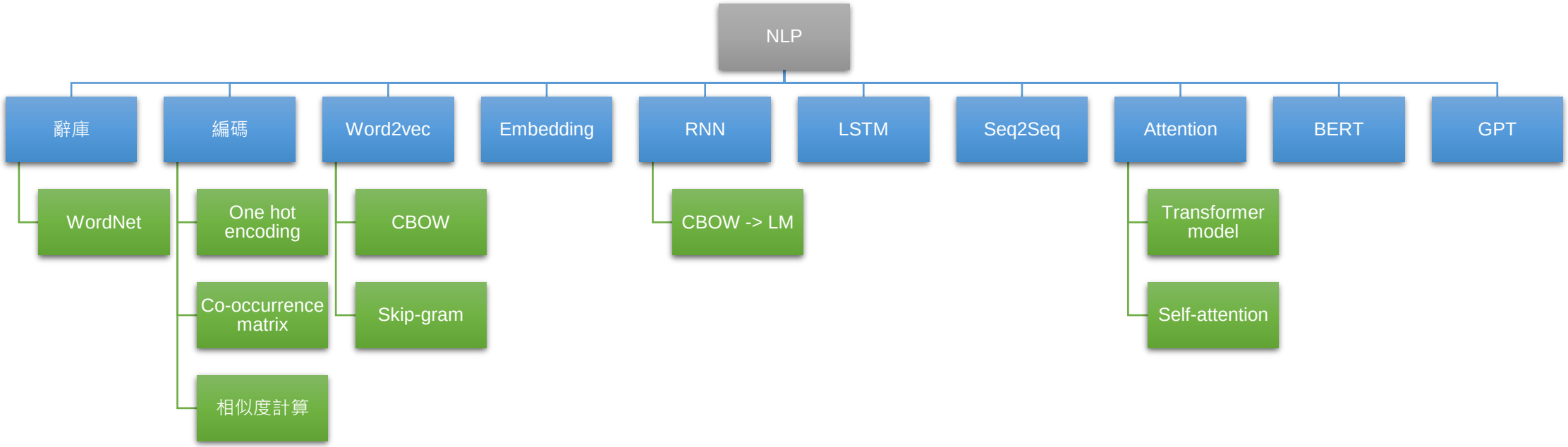
NLU

- Understanding: sentiment detection, topic classification, entity detection.

NLG

- Generation: structure data into text

自然語言處理介紹


NLP
辭庫
編碼
Word2vec
Embedding
RNN
LSTM
Seq2Seq
Attention
BERT
GPT
WordNet
One hot encoding
Co-occurrence matrix
相似度計算
CBOW
Skip-gram
CBOW -> LM
Transformer model
Self-attention

自然語言處理介紹

- 我們學習字詞一般的方式是利用字典 / 辭典

http://dict.revised.moe.edu.tw/cgi-bin/cbdic/gsweb.cgi

自然語言處理介紹

- 要讓電腦可以分析字詞，一種是以詞庫(thesaurus)的方式理解內容
- 詞庫 => 同義詞辭典，把意義相同的詞分類在同一個群

https://www.moedict.tw/%E6%B1%BD%E8%BB%8A

自然語言處理介紹

- 詞庫(thesaurus)
  - 也稱為敘詞表或類語辭典，同義詞辭典，是主題分析的一種實作方法。
- WordNet
  - 有名的詞庫，WordNet 是由普林斯頓大學從1985 年開始開發的詞庫。
- 詞庫的問題
  - 需要維護
  - 無法表現字詞的細微差異

自然語言處理介紹 - WordNet 簡介

- WordNet 是由普林斯頓大學從 1985 年開始開發的詞庫
- 既是一個字典，也是辭典
- 可自由下載使用

- 中文也有一個中文詞彙網路Chinese Wordnet

自然語言處理介紹 - NLTK 簡介

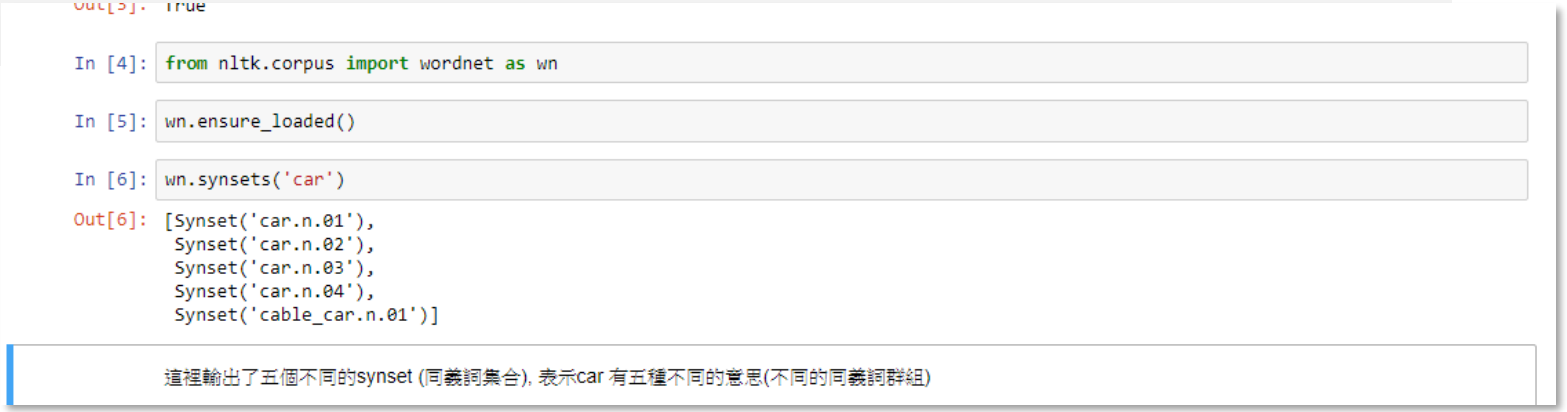
- 在 python 中使用 WordNet ，必須使用NLTK (Natural Language Toolkit)
- NLTK 是處理自然語言的python 函式庫
- NLTK 可以做詞性標記、語法分析、資料擷取等功能。

WordNet 初體驗

WordNet 初體驗 - 安裝 NLTK

請在 jupyter 下執行以下程式碼

```
1. !pip install nltk
2. import nltk
3. nltk.download('wordnet')
4. from nltk.corpus import wordnet as wn
5. wn.ensure_loaded()
6. wn.synsets('car')
```



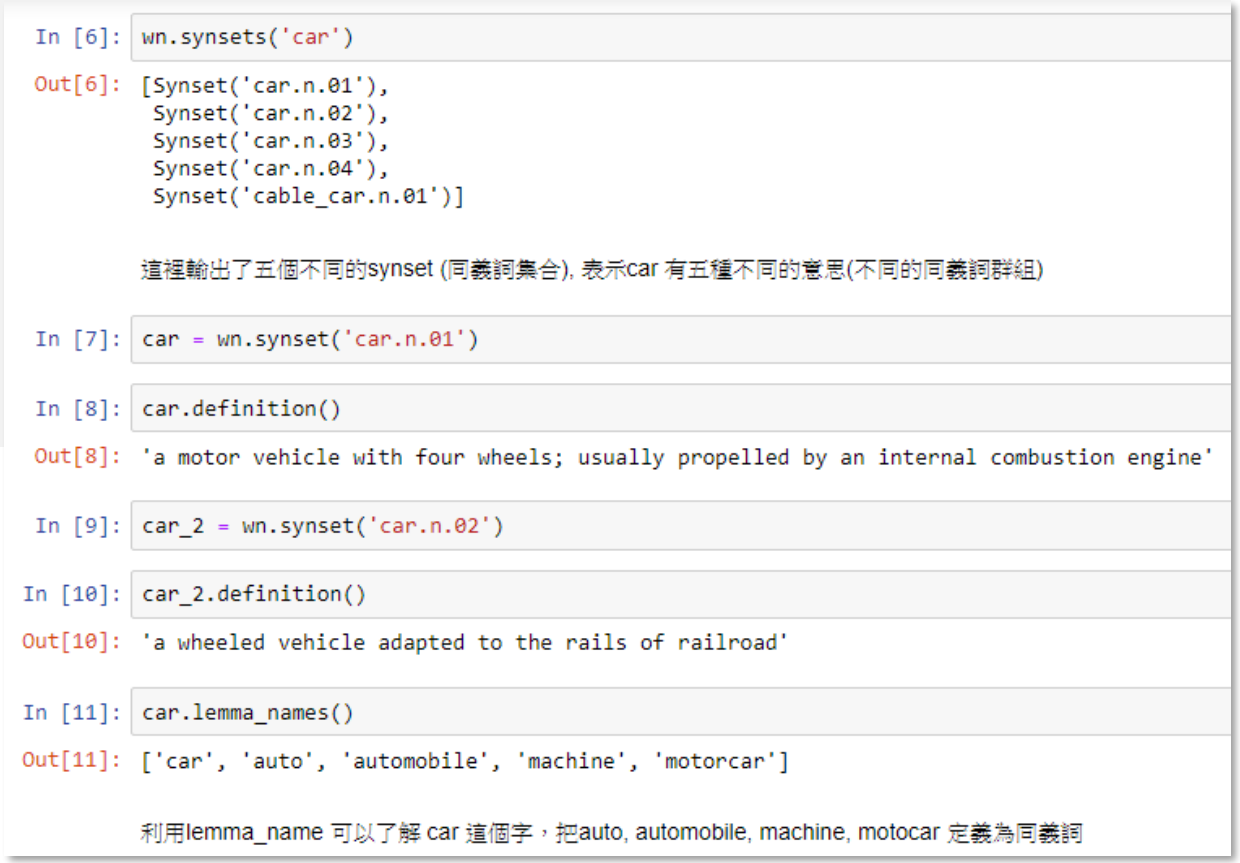
DEMO

這裡輸出了五個不同的synset (同義詞集合), 表示car 有五種不同的意思(不同的同義詞群組)

WordNet 初體驗 - 取得同義詞

請在 jupyter 下執行以下程式碼

```
1. car = wn.synset('car.n.01')
2. car.definition()
3. car_2 = wn.synset('car.n.02')
4. car_2.definition()
5. car.lemma_names()
```



WordNet 初體驗 - 詞意相似度

請在 jupyter 下執行以下程式碼

```
1. car.hypernym_paths()[0] #"上位詞"
# 愈往上，愈抽象；愈往下，愈具體。
# 接下來看 Wrodnet的詞意相似度
2. car = wn.synset('car.n.01')
3. novel = wn.synset('novel.n.01')
4. dog = wn.synset('dog.n.01')
5. motorcycle = wn.synset('motorcycle.n.01')
6. car.path_similarity(novel)
7. car.path_similarity(dog)
8. car.path_similarity(motorcycle)
9. cat = wn.synset('cat.n.01')
10.car.path_similarity(cat)
```

WordNet 初體驗 – 關於詞庫

- 詞庫(thesaurus)
  - 也稱為敘詞表或類語辭典，同義詞辭典，是主題分析的一種實作方法。
- WordNet
  - 有名的詞庫，WordNet 是由普林斯頓大學從1985 年開始開發的詞庫。
- 詞庫的問題
  - 需要維護
  - 無法表現字詞的細微差異

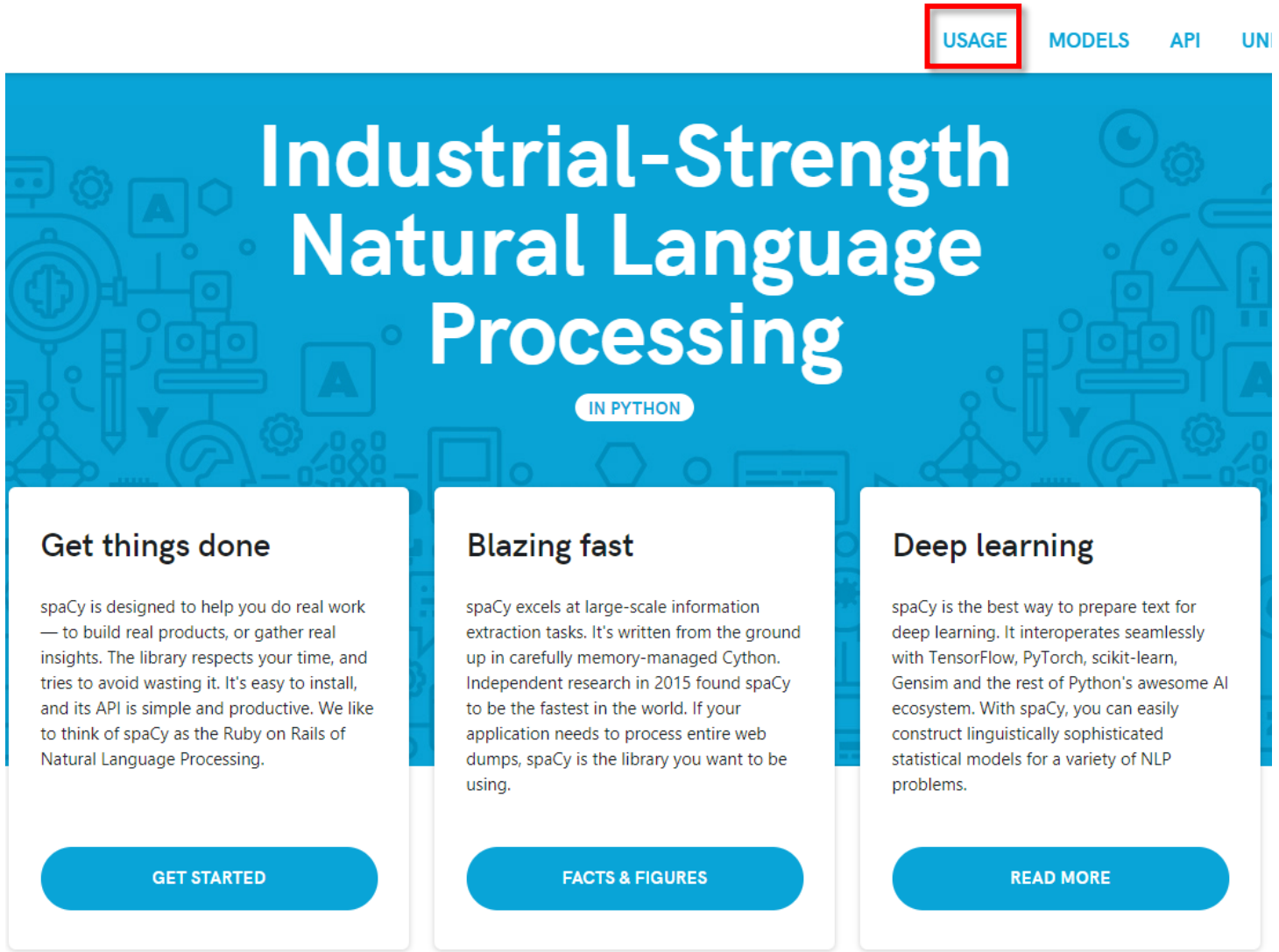
spaCy

spaCy 介紹

spaCy是用於高級自然語言處理的開源程式庫，它使用Python和Cython編程語言編寫。該程式庫是在MIT許可下發布的，其主要開發人員是軟體公司Explosion的創始人Matthew Honnibal和Ines Montani。與廣泛用於教學和研究的NLTK不同，spaCy專注於提供用於生產等級的軟體。(取自維基百科)

spaCy 安装

https://spacy.io/

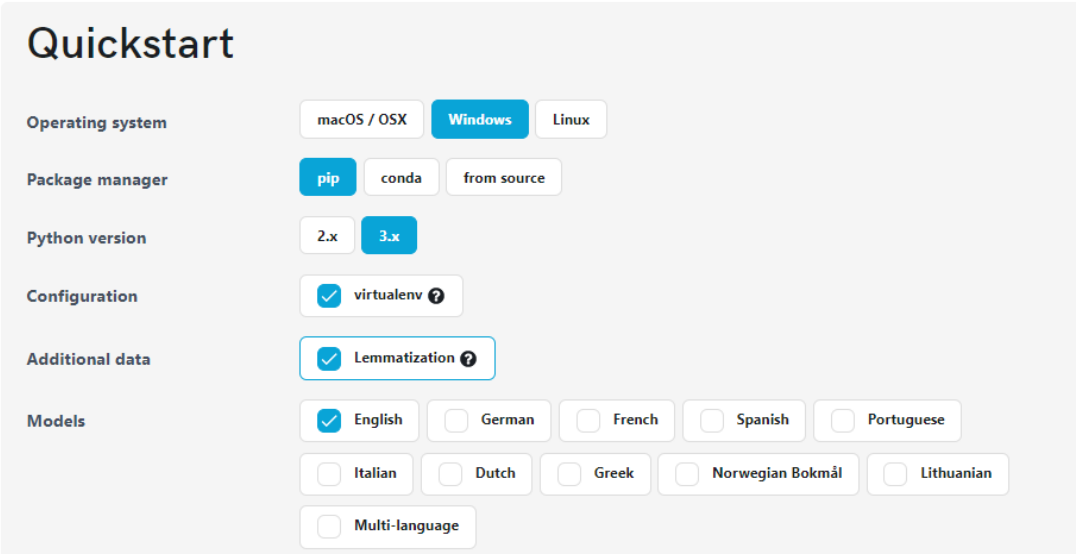


spaCy 安装

Install spaCy

spaCy is compatible with **64-bit CPython 2.7 / 3.5+** and runs on **Unix/Linux**, **macOS/OS X** and **Windows**. The latest spaCy releases are available over pip and conda.

Quickstart




```
$ python -m venv .env
$ .env\Scripts\activate
$ pip install -U spacy
$ pip install -U spacy-lookups-data
$ python -m spacy download en_core_web_sm
```

spaCy NLP

Text: The original word text.
Lemma: The base form of the word.
POS: The simple UPOS part-of-speech tag.
Tag: The detailed part-of-speech tag.
Dep: Syntactic dependency, i.e. the relation between tokens.
Shape: The word shape – capitalization, punctuation, digits.
is alpha: Is the token an alpha character?
is stop: Is the token part of a stop list, i.e. the most common words of the language?

- Part-of-speech tagging

TEXT	LEMMA	POS	TAG	DEP	SHAPE	ALPHA	STOP
Apple	apple	PROPN	NNP	nsubj	Xxxxx	True	False
is	be	VERB	VBZ	aux	xx	True	True
looking	look	VERB	VBG	ROOT	xxxx	True	False
at	at	ADP	IN	prep	xx	True	True
buying	buy	VERB	VBG	pcomp	xxxx	True	False
U.K.	u.k.	PROPN	NNP	compound	X.X.	False	False
startup	startup	NOUN	NN	dobj	xxxx	True	False
for	for	ADP	IN	prep	xxx	True	True
$	$	SYM	$	quantmod	$	False	False
1	1	NUM	CD	compound	d	False	False
billion	billion	NUM	CD	pobj	xxxx	True	False

https://spacy.io/usage/spacy-101

spaCy NLP

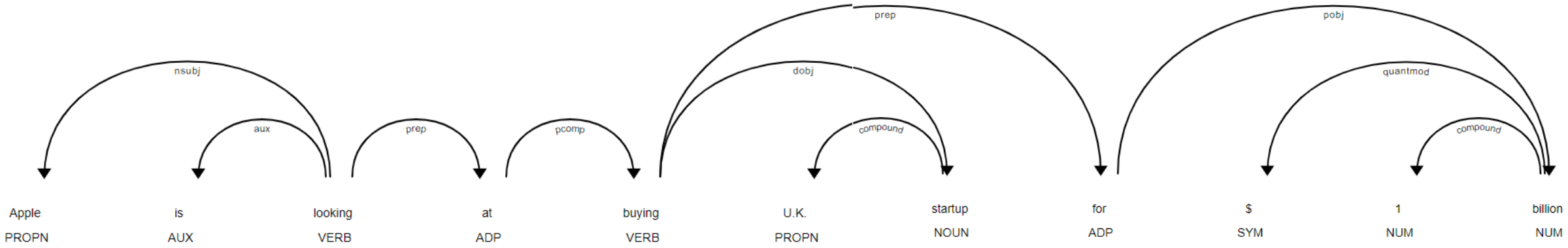
- Tokenization

0	1	2	3	4	5	6	7	8	9	10
Apple	is	looking	at	buying	U.K.	startup	for	$	1	billion

https://spacy.io/usage/spacy-101

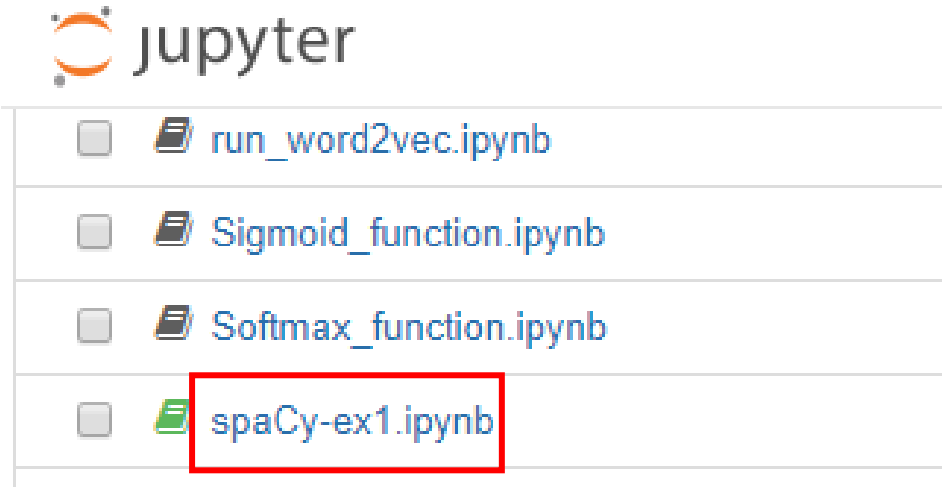
spaCy NLP

- Dependency



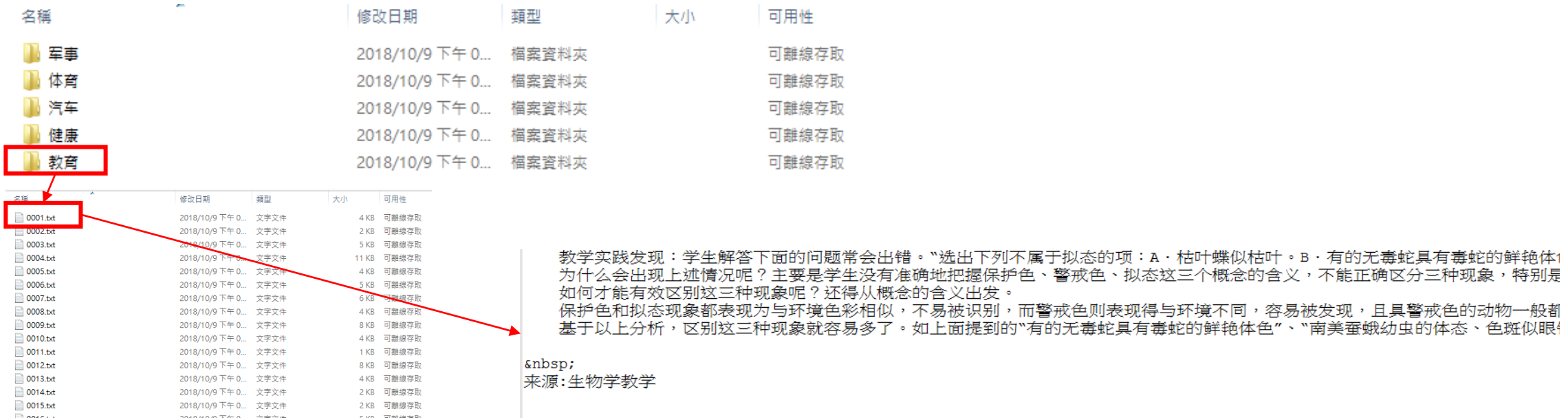
spaCy NLP 實作

- 利用瀏覽器開啟我們在docker 中準備好的 lab
- 網址： [你的電腦或VM的IP]:9999



自然語言處理介紹

- 除了詞庫之外，語料庫(corpus)也用於NLP中
- 語料庫就是大量的文本資料，這些資料是由人工產生，為了研究NLP而產生的資料



自然語言處理介紹

- 語料庫(corpus)
  - 透過蒐集得到的，大量的文本資料
  - **在講義中，我們簡化大量的文本資料，利用簡單的句子作為文本的語料庫。**

You say YES but I say No.

- 上下文(context)
  - 指的是某個字附近的字詞，例如 say 的上下文為 you, YES 兩個字
  - Window size 指的是上下文的大小，包含周圍多少字

自然語言處理介紹

- 不管是詞庫還是語料庫，進行自然語言處理還是會引入數學模型
- 前面提到的詞矩陣(BoW) 就是向量化的一種方式，另外還有
- One-hot encoding
- Co-occurrence matrix

自然語言處理介紹

- BoW(Bag of Words)
- 假設目前有X篇文件(document)，在X篇文件中總共有Y個詞項(term)。在X=5, Y=6的情況可以表述為下列的表格(矩陣)：

文件/詞	詞1	詞2	詞3	詞4	詞5	詞6
文件1	0	3	7	0	1	1
文件2	0	2	8	4	0	2
文件3	1	7	0	1	1	0
文件4	8	0	1	0	0	0
文件5	10	0	1	2	0	0

一個詞一個值

自然語言處理介紹

- **利用Bow，可以計算 TF, IDF**
- **TF -> term frequency 詞頻**
  - 特定詞項在該文件中出現的頻率，例如詞t在第i篇文件中出現的頻率，記為tf(t, i)
  - EX：文件 a 總共有100個字，而詞 i 在文件a 出現的次數是12次，因此 tf (i,a)=12/100，**以頻率而不是次數來看待文字的重要性，來比對文章與文章之間的相似程度。**
- **IDF -> inverse document frequency 逆文件頻率**
  - 特定詞t在多篇文章出現的頻率的倒數再取對數，記為idf(t)
  - idf(t) = log S(totoal_doc_count / doc_count(t))
  - **IDF 最常用在處理常用字的部分**
- **TF-IDF = TF x IDF**，某一特定檔案內的高詞語頻率，以及該詞語在整個檔案集合中的低檔案頻率，可以產生出高權重的tf-idf。**因此，tf-idf傾向於過濾掉常見的詞語，保留重要的詞語。**

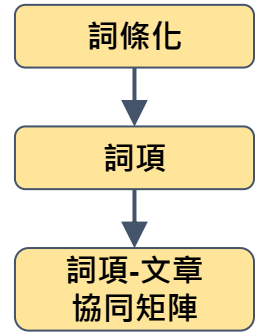
自然語言處理介紹

- **TF-IDF 利用餘弦相似性（cosine similarity）在向量空間模型中可以解釋與判斷兩份檔案之間的相似性。**
- **TF-IDF 認為在文章中出現頻率小的詞就越重要，頻率大的詞就愈不重要，這可能會出現不太正確的結果：**
  - 被忽略的高頻率詞；高率頻詞不等於無意義的詞。 IDF 原先是想忽視無意義的高頻詞（通常是 stopwords）的影響，在專業領域的文本中很容易發生。
  - 某些需要的低頻詞被忽視。

stopword = { "的", "了", "在", "是", "我", "有", "和", "就", "不", "人", "都", "一", "一个", "上", "也", "很", "到", "说", "要", "去", "你", "会", "着", "没有", "看", "好", "自己", "这" };

自然語言處理介紹

複數 -> 單數
大寫 -> 小寫
動詞還原



#id	Document (in one field)	word
#0	One shoe, two shoe, the red shoe, the blue shoe, lots of shoes.	one two shoe the red blue lots of
#1	The blue dress shoe is the best shoe.	the blue dress shoe is best
#2	The best dress is the one red dress.	the best dress is one red

term

One -> one	shoes -> shoe	The -> the	red
blue	lots -> lot	of	dress
is	best	two	

BOW

	one	two	shoe	the	red	blue	lot	of	dress	is	best
#0	1	1	5	2	1	1	1	1	0	0	0
#1	0	0	2	2	0	1	0	0	1	1	1
#2	1	0	0	2	1	0	0	0	2	1	1

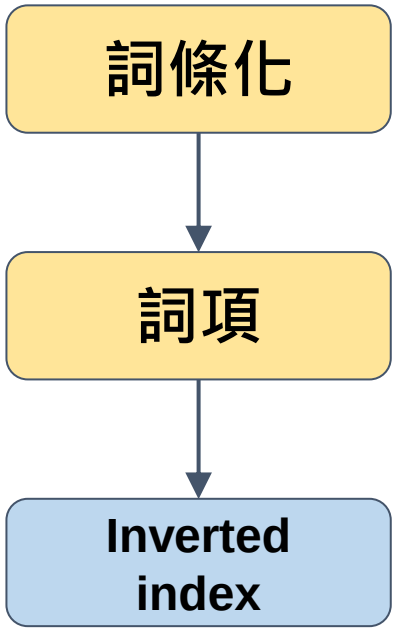
自然語言處理介紹

- 詞袋模型（英語：Bag-of-words model）是個在自然語言處理和信息檢索(IR)下被簡化的表達模型。此模型下，像是句子或是文件這樣的文字可以用一個袋子裝著這些詞的方式表現，這種表現方式不考慮文法以及詞的順序。最近詞袋模型也被應用在電腦視覺領域。
- 詞袋模型被廣泛應用在文件分類，詞出現的頻率可以用來當作訓練分類器的特徵。

(以上來源：維基百科)

	one	two	shoe	the	red	blue	lot	of	dress	is	best
#0	1	1	5	2	1	1	1	1	0	0	0
#1	0	0	2	2	0	1	0	0	1	1	1
#2	1	0	0	2	1	0	0	0	2	1	1

自然語言處理介紹



term	doc
one	0,2
two	0
shoe	0,1
the	0,1,2
red	0,2
blue	0,1
lot	0
of	0
dress	1,2
is	1,2
best	1,2

自然語言處理介紹

tf(“one” , #0) = 1/13

tf(“one” , #1) = 0/8 = 0

Idf(“one” ,D) = total doc number / (term in doc number+1) = 3 / 3 = 1

tf-idf(“one” , #0) = 1/13 * 1 = 0.08

tf-idf(“one” , #1) = 0 * 1 = 0

	one	two	shoe	the	red	blue	lot	of	dress	is	best	<total>
#0	1	1	5	2	1	1	1	1	0	0	0	13
#1	0	0	2	2	0	1	0	0	1	1	1	8
#2	1	0	0	2	1	0	0	0	2	1	1	8
df=	*2*	*1*	*2*	*3*	*2*	*2*	*1*	*1*	*2*	*2*	*2*	
idf=	*1.00*	*1.50*	*1.00*	*0.75*	*1.00*	*1.00*	*1.50*	*1.50*	*1.00*	*1.00*	*1.00*	
#0 tf	0.08	0.08	0.38	0.15	0.08	0.08	0.08	0.08	0.00	0.00	0.00	
#1 tf	0.00	0.00	0.25	0.25	0.00	0.13	0.00	0.00	0.13	0.13	0.13	
#2 tf	0.13	0.00	0.00	0.25	0.13	0.00	0.00	0.00	0.25	0.13	0.13	
#0 tfidf	0.08	0.12	0.38	0.12	0.08	0.08	0.12	0.12	0.00	0.00	0.00	
#1 tfidf	0.00	0.00	0.25	0.19	0.00	0.13	0.00	0.00	0.13	0.13	0.13	
#2 tfidf	0.13	0.00	0.00	0.19	0.13	0.00	0.00	0.00	0.25	0.13	0.13	

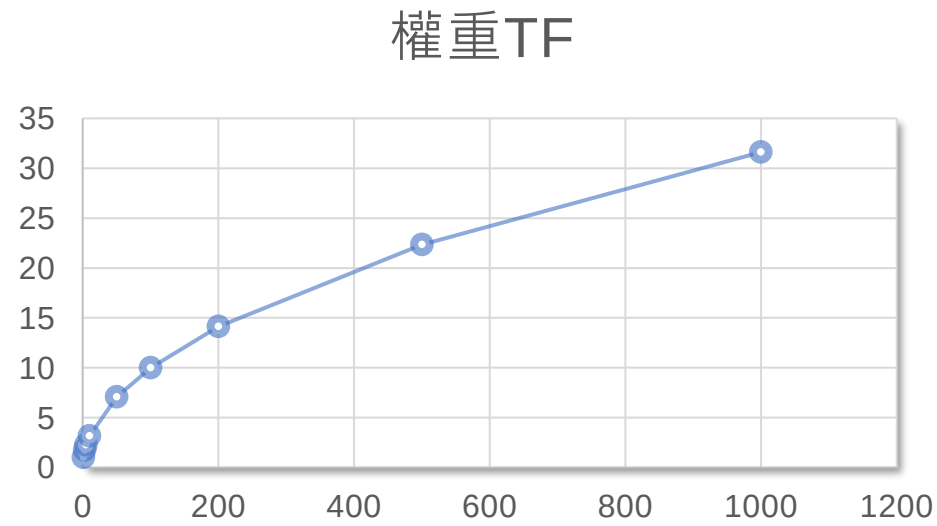
自然語言處理介紹

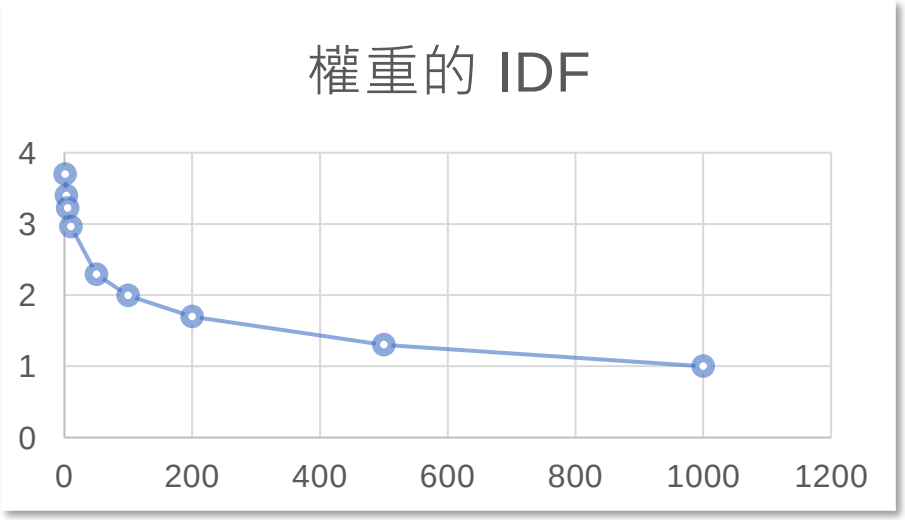
抑制TF-IDF的方式：

TF weight = sqrt(tf)

IDF weight = log(numDocs / (df+1)) +1

TF	權重TF		DF	權重的 IDF
1	1		1	3.698970004
2	1.414214		2	3.522878745
5	2.236068		5	3.22184875
10	3.162278		10	2.958607315
50	7.071068		50	2.292429824
100	10		100	1.995678626
200	14.14214		200	1.696803943
500	22.36068		500	1.300162274
1000	31.62278		1000	0.999565923





40

自然語言處理介紹

Query: “two shoes”

	one	two	shoe	the	red	blue	lot	of	dress	is	best
#0	1	1	5	2	1	1	1	1	0	0	0
#1	0	0	2	2	0	1	0	0	1	1	1
#2	1	0	0	2	1	0	0	0	2	1	1
Q	0	1	1	0	0	0	0	0	0	0	0

41

自然語言處理介紹

- 分布假說(distributional hypothesis)
  - 詞意是由周圍的字詞所形成

I see.

I understand.

I see you.

I watch you.

I drink wine.

We drink wine.

I booze wine.

自然語言處理介紹

- One-hot 編碼
  - 每個字 / 詞都用一個向量表示
  - 該向量長度為不重複字/詞的數量
  - 向量中只有一個 1 ，其他都是 0

自然語言處理介紹

You say YES but I say No.

- One-hot 編碼
  - you : **1 0 0 0 0 0 0**
  - say : **0 1 0 0 0 0 0**
  - YES: **0 0 1 0 0 0 0**
  - but: **0 0 0 1 0 0 0**
  - I: **0 0 0 0 1 0 0**
  - No: **0 0 0 0 0 1 0**
  - . : **0 0 0 0 0 0 1**

自然語言處理介紹

- Co-occurrence matrix
  - 共生矩陣
  - 考慮字/詞附近的周圍進行統計，形成向量
  - 字/詞附近(一個字的距離)向量值為 1 ，其他都是 0

You say YES but I say No.

	you	say	YES	but	I	No	.
you	0	1	0	0	0	0	0

自然語言處理介紹

- Co-occurrence matrix

You say YES but I say No.

	you	say	YES	but	I	No	.
say	1	0	1	0	1	1	0

You say YES but I say No.

	you	say	YES	but	I	No	.
YES	0	1	0	1	0	0	0

自然語言處理介紹

共生矩陣

You say YES but I say No.

	you	say	YES	but	I	No	.
you	0	1	0	0	0	0	0
say	1	0	1	0	1	1	0
YES	0	1	0	1	0	0	0
but	0	0	1	0	1	0	0
I	0	1	0	1	0	0	0
NO	0	1	0	0	0	0	1
.	0	0	0	0	0	1	0

自然語言處理介紹

- 向量之間的相似度，常用的方法是餘弦相似度

$$Similarity(x, y) = \frac{x \cdot y}{\sqrt{x_1^2 + \cdots + x_n^2}\sqrt{y_1^2 + \cdots + y_n^2}}$$

Word2vec

Word2vec

- **Word2Vec** 是Google 在2013年發表的詞向量工具，想深入了解可以參考這篇論文 Distributed Representations of Words and Phrases and their Compositionality
- 兩種問題，兩種模型

Word2vec – model

Word2vec – CBOW

對於一串文字，考慮第t個字，上下文觀察視窗為 1 的上下文：

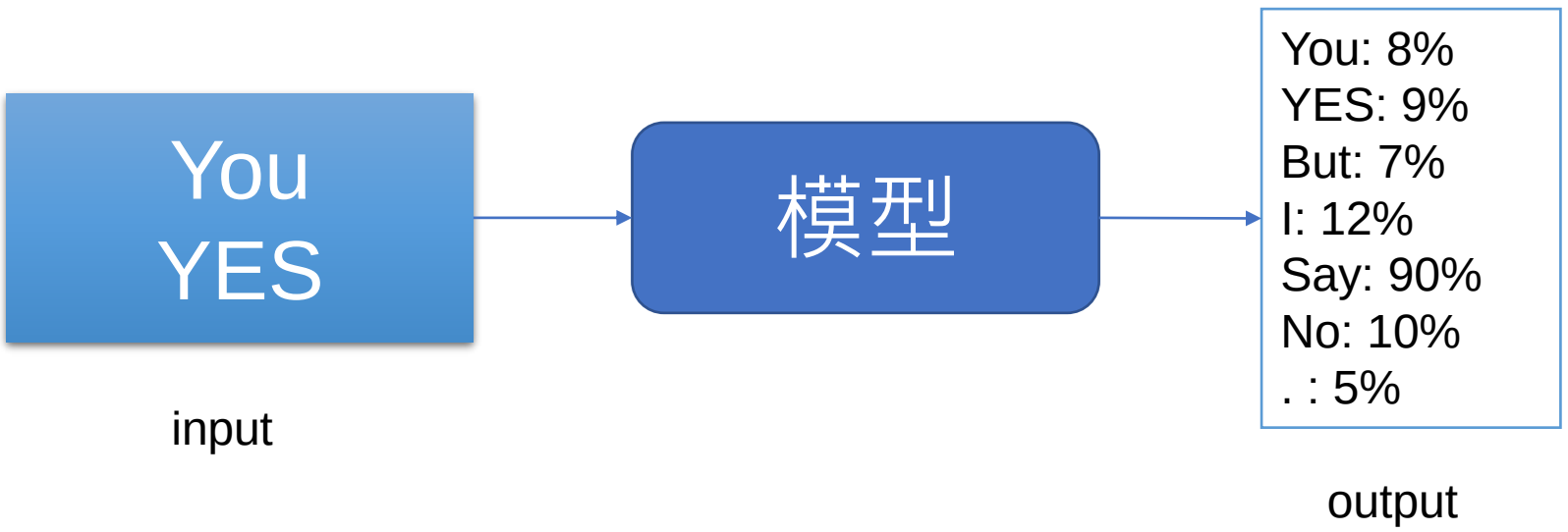
$$w_1, w_2, w_3 \dots\dots, w_{t-1}, w_t\ , w_{t+1}, \dots\dots, w_{T-1}, w_T$$

第t個字發生的機率為

$$P(w_t | w_{t-1}, w_{t+1})$$

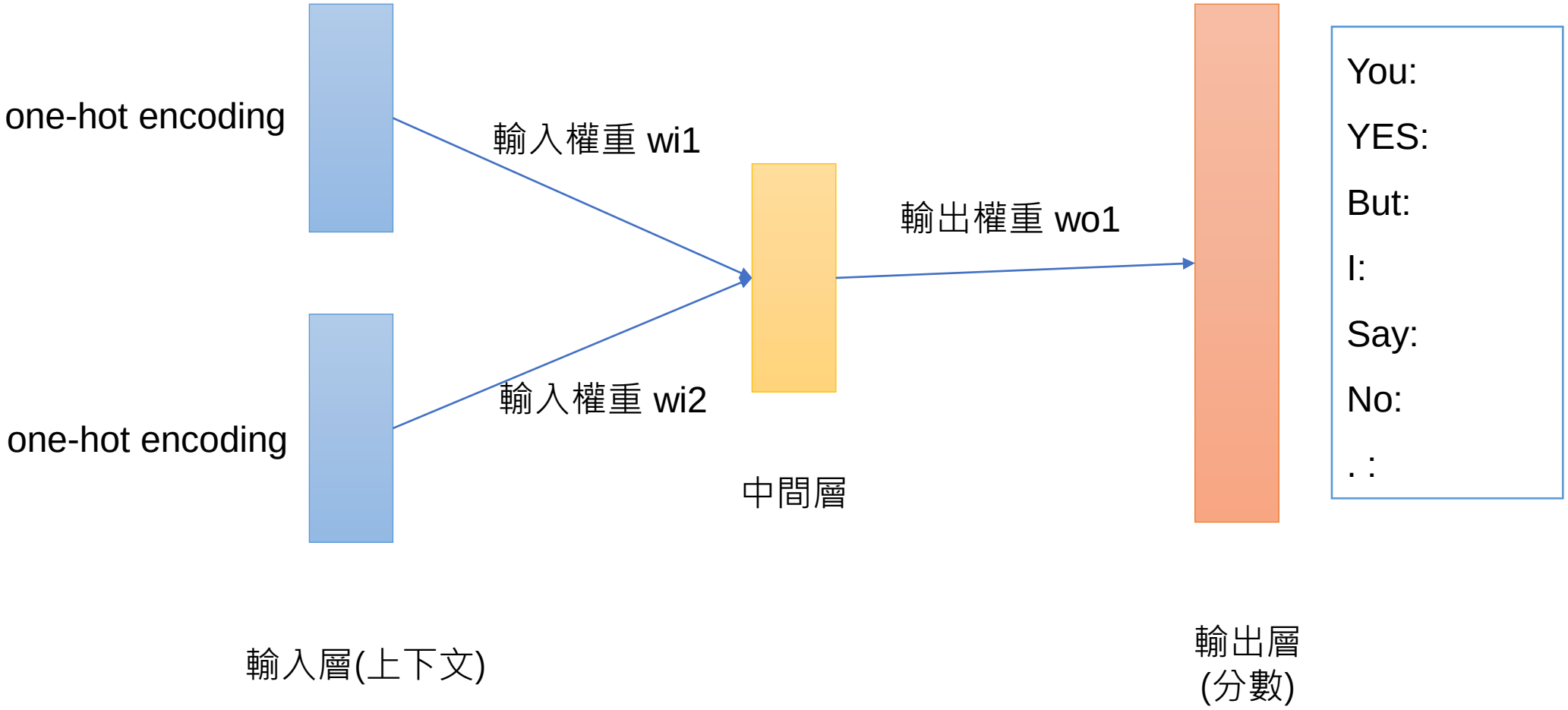
Word2vec - CBOW

You _?_ YES but I say No.



Word2vec - CBOW

CBOW模型的類神經網路架構



Word2vec - CBOW

- 上下文與目標對象

語料庫 corpus	上下文 context	目標 target
You say YES but I say No.	you, YES	say
You say YES but I say No.	say, but	YES
You say YES but I say No.	____, ____	but
You say YES but I say No.	____, ____	I
You say YES but I say No.	____, ____	say
You say YES but I say No.	____, ____	No

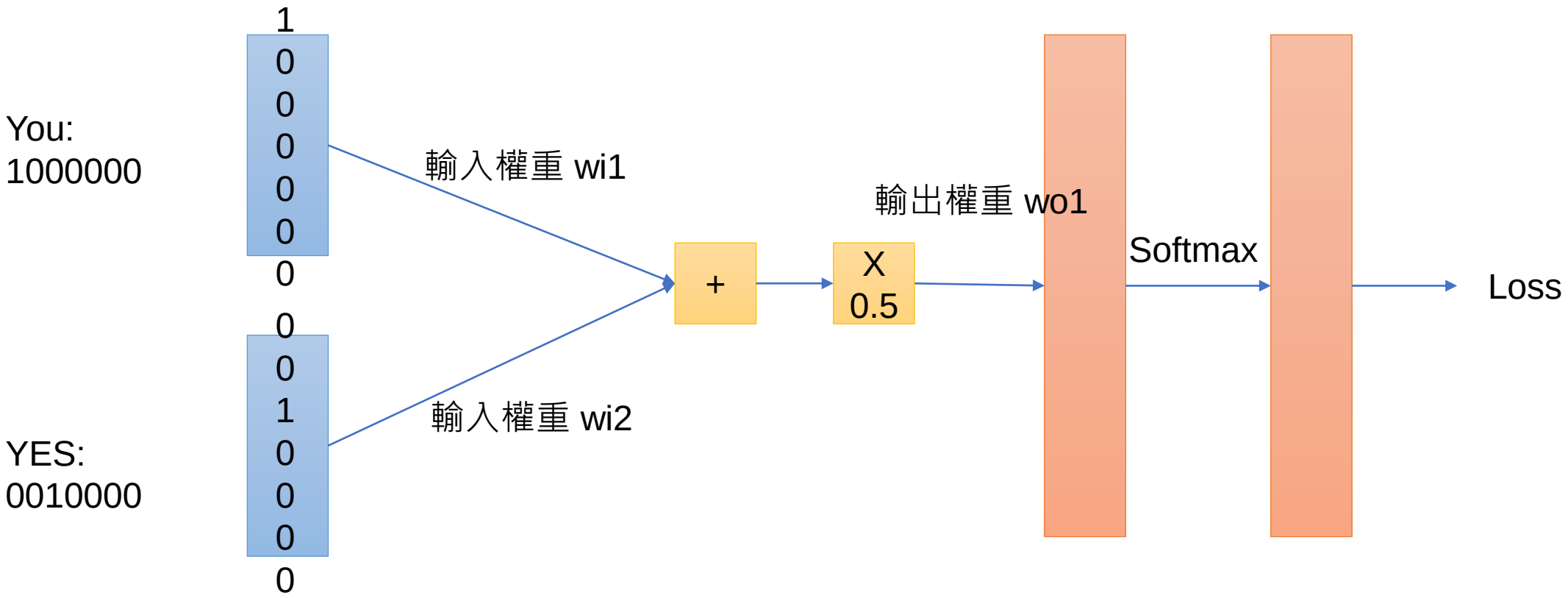
Word2vec - CBOW

- 轉成one-hot 編碼

context	target
you, YES	say
say, but	YES
YES, I	but
but, say	I
I, No	say
say, .	No

context	target
1000000,0010000	0100000
0100000,0001000	0010000
0010000,0000100	0001000
0001000,0100000	0000100
0000100,0000010	0100000
0100000,0000001	0000010

Word2vec - CBOW



DEMO

Word2vec – Skip-gram

對於一串文字，考慮第t個字，上下文觀察視窗為 1 的上下文：

$$w_1, w_2, w_3 \dots\dots, w_{t-1}, w_t\ , w_{t+1}, \dots\dots, w_{T-1}, w_T$$

給定第t個字的時候，同時發生第t-1 個字與t+1 個字的機率為

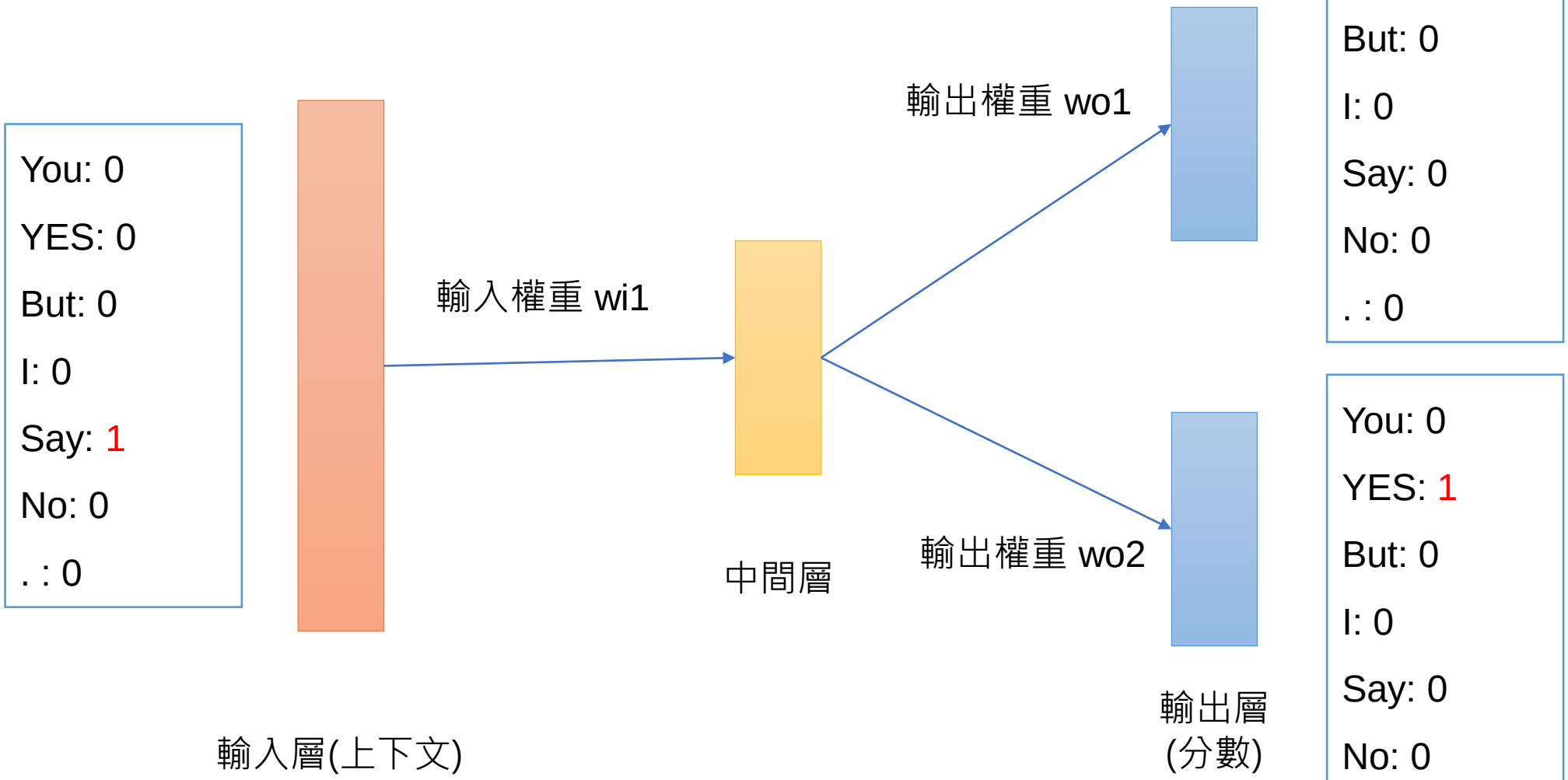
$$P(w_{t-1}, w_{t+1} | w_t)$$

Word2vec – Skip-gram

- 例如

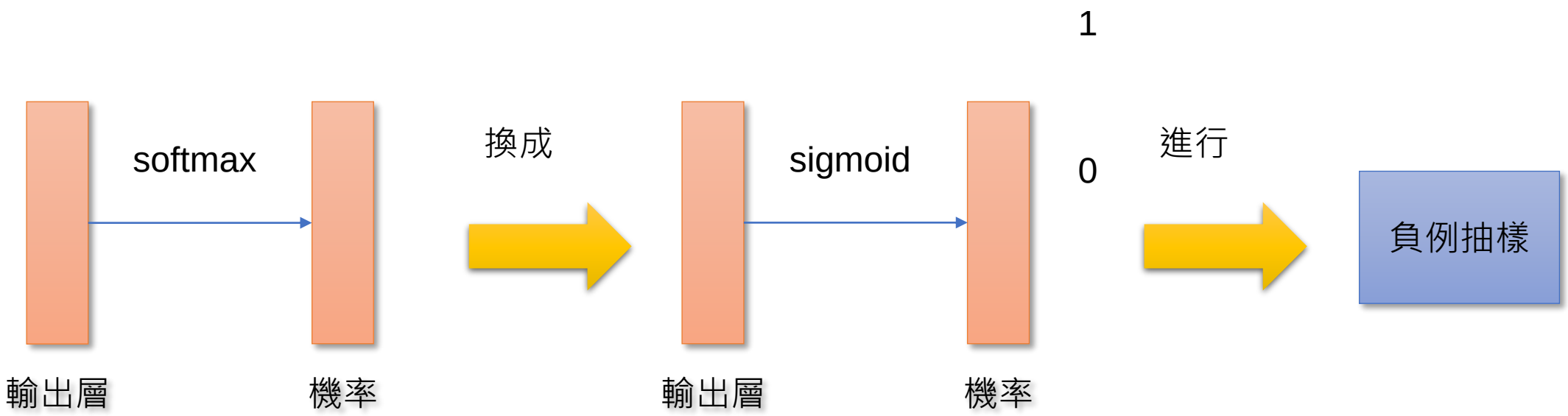
_?_ say _?_ but I say No.

Word2vec – Skip-gram



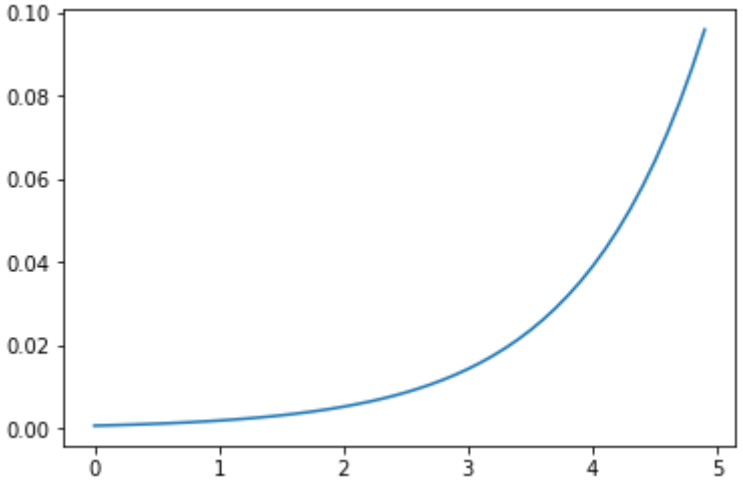
Negative sampling

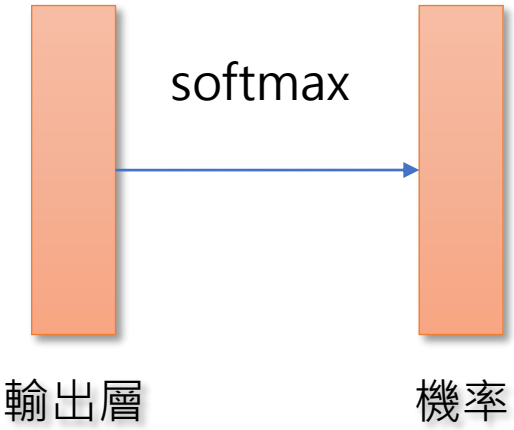
- 負例抽樣
- 將輸出調整為二元分類



Negative sampling

- Softmax 的性質
- 適合多值分類





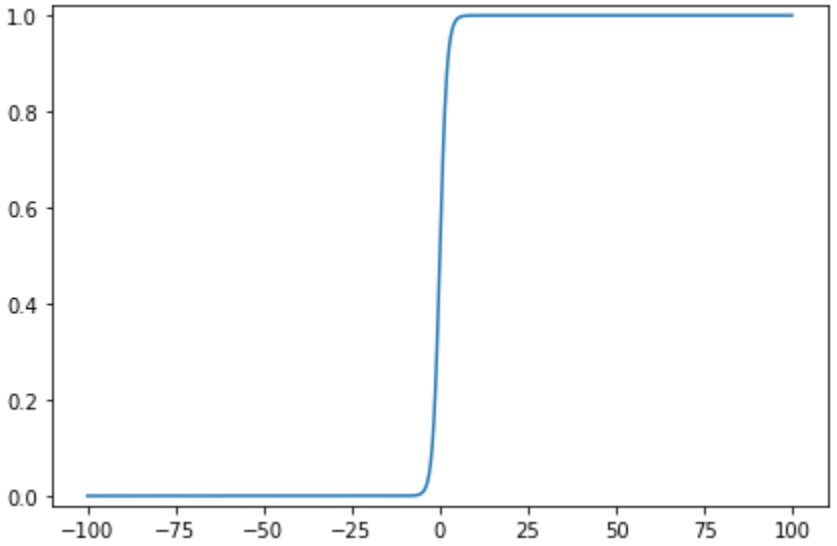
$$y_k = \frac{\exp(a_k)}{\sum_{1^1=1}^{n} \exp(a_i)}$$

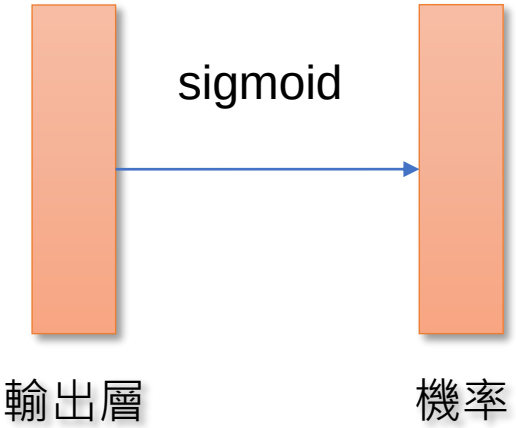
```
def softmax(x):
    c = np.max(x)
    exp_ac = np.exp(x-c)
    y = exp_ac / np.sum(exp_ac)
    return y
```

DEMO

Negative sampling

- Sigmoid 特性
- 適合二元分類





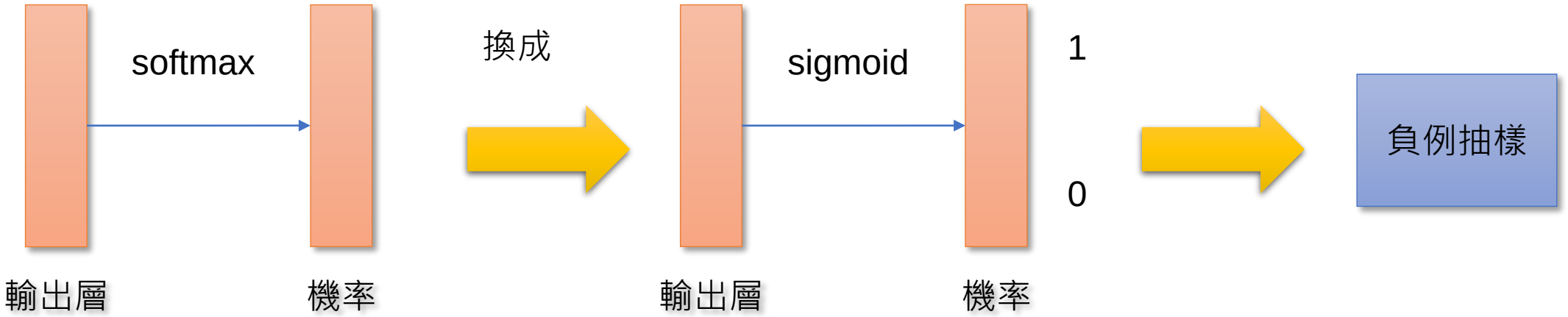
$$h(x) = \frac{1}{1 + \exp(-x)}$$

```
def sigmoid(x):
    s = 1 / (1 + np.exp(-x))
    return s
```

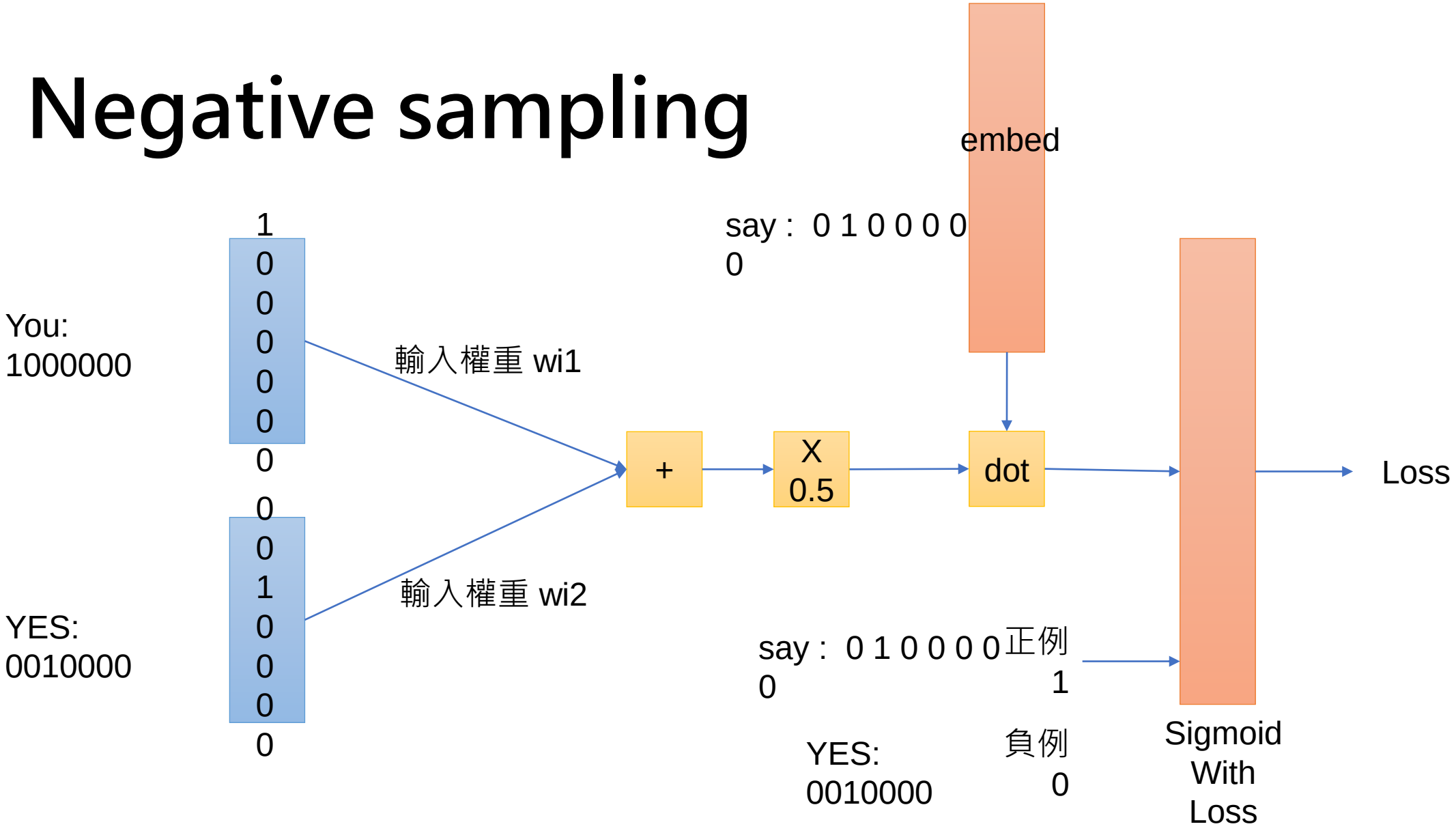
DEMO

Negative sampling

- 將輸出調整為二元分類
- 模型在訓練的時候，除了正例之外，也需要輸入負例來進行學習，在詞彙數量愈來愈大的情況下，全部的負例都進去學習會需要很多時間，因此在負例的部分就用取樣的方式取出一定數量的負例去做模型的學習。



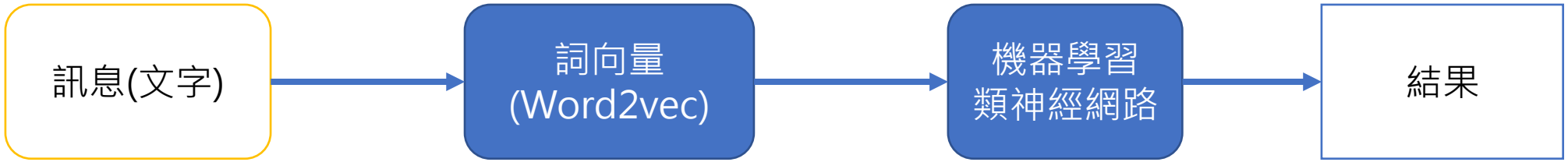
Negative sampling



DEMO

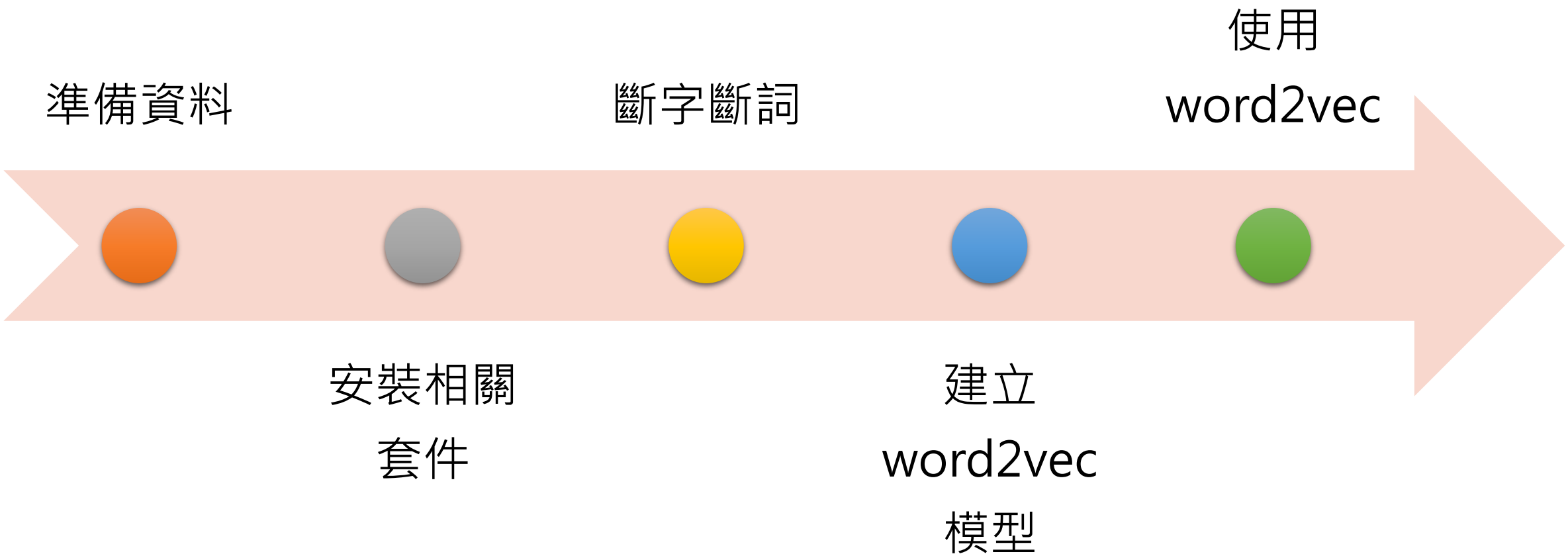
Word2vec 應用

- 問答系統
- 文件自動分類系統



Word2vec 實作

Word2vec 實作



DEMO

Word2vec 實作 - 準備資料

- 在網路上取得純文字檔的 三國演義 小說
- 檢查檔案編碼，並調整為UTF-8
- 上傳到 Jupyter 所在的server

Word2vec 實作 -安裝相關套件

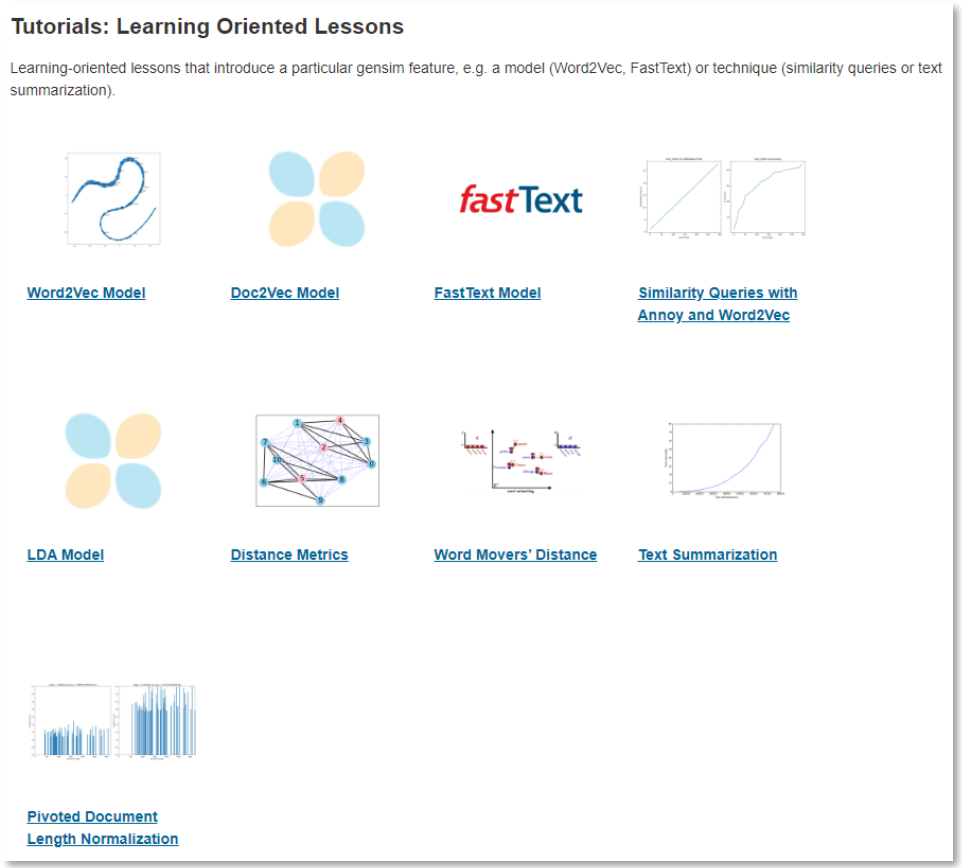
- 安裝 jieba 套件
- 安裝 gensim 套件
- 在 jupyter 中可以利用 !pip 來安裝

```
1. ! pip install jieba
2. ! pip install gensim
```

```
In [2]: !pip install jieba

        Collecting jieba
          Downloading https://files.pythonhosted.org/packages/71/46/c6f9179f73b818d5827202ad1c4a94e371a29473b7f043b736b4dab6b8cd/jieba-
        0.39.zip (7.3MB)
             |████████████████████████████████| 7.3MB 29.2MB/s eta 0:00:01
        Building wheels for collected packages: jieba
          Building wheel for jieba (setup.py) ... done
          Created wheel for jieba: filename=jieba-0.39-cp37-none-any.whl size=7282593 sha256=e0c52e50334c25d0085879264fa013a0e0435e0f96
        f7aed9d862448354bc239e
          Stored in directory: /home/jovyan/.cache/pip/wheels/c9/c7/63/a9ec0322ccc7c365fd51e475942a82395807186e94f0522243
        Successfully built jieba
        Installing collected packages: jieba
        Successfully installed jieba-0.39

In [ ]: !pip install gensim
```

Word2vec 實作 – genism 介紹

- Gensim 是topic modeling 的函式庫
- 提供Corpus 相關的功能可以處理文本
- 字詞向量化
- 建模(tf-idf, word2vec, Doc2vec, FastText, LDA… etc)



Word2vec 實作 -斷字斷詞

- 載入 jieba 函式庫

```
1. import jieba
```

- 開啟準備好的檔案

```
# 開檔
1. fileAllLines = []
2. with open('Three Kingdoms.txt') as fileLine:
3.     for line in fileLine:
4.         #print(line)
5.         fileAllLines.append(line)
```

Word2vec 實作 -斷字斷詞

- 斷字詞

```
1. seg = []
2. for i in range(len(fileAllLines)):
3.     seg.append([' '.join(list(jieba.cut(fileAllLines[i])))])
4. get_ipython().system('rm segDone.txt')
```

- 斷詞結果存檔

```
1. segSaveFile = 'segDone.txt'
2. with open(segSaveFile, 'wb') as saveFile:
3.     for i in range(len(seg)):
4.         saveFile.write(seg[i][0].encode('utf-8'))
5.         saveFile.write('\n'.encode())
```

Word2vec 實作 -建立word2vec模型

- 建立模型

```
1. from gensim.models import word2vec
2. sentences = word2vec.LineSentence("segDone.txt")
3. model = word2vec.Word2Vec(sentences, size=300, iter=100, sg=0, workers=3)
```

size: 向量維度

iter: 迭代次數

sg: 0(CBOW), 1(Skip-gram)

Word2vec 實作 –儲存word2vec模型

- 儲存訓練好的word2vec 模型

```
1. model.save("word2vec.model")
2. model.corpus_total_words
```

儲存訓練好的word2vec 模型

```
In [11]: model.save("word2vec.model")

In [12]: model.corpus_total_words
Out[12]: 392015
```

Word2vec 實作 -使用word2vec

- 相似度

```
1. model.most_similar('赤壁')
2. model.similar_by_word('赤壁')
```

詞相似度

```
In [13]: model.most_similar('赤壁')
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `most_similar` (Method w
ill be removed in 4.0.0, use self.wv.most_similar() instead).
  """Entry point for launching an IPython kernel.
```

```
Out[13]: [('文王', 0.39981240034103394),
 ('楚', 0.3953494131565094),
 ('七年', 0.39043542742729187),
 ('幕府', 0.37336504459381104),
 ('渭橋', 0.36399832367897034),
 ('在位', 0.36285775899887085),
 ('宜', 0.35630089044570923),
 ('韓信', 0.3513275384902954),
 ('疲', 0.3441169261932373),
 ('業', 0.3367699384689331)]
```

Word2vec 實作 -使用word2vec

- 詞預測

```
1. model.predict_output_word('赤壁')
```

詞預測

```
In [13]: model.predict_output_word('赤壁')

Out[13]: [('下', 0.6779435),
          ('無', 0.12980454),
          ('幘', 0.047978833),
          ('奔走', 0.03580733),
          ('太守', 0.023490814),
          ('黃', 0.018670587),
          ('取', 0.005650897),
          ('、', 0.0031672297),
          ('赤', 0.0026711996),
          ('主公', 0.001182011)]
```

Word2vec 實作 -使用word2vec

- 類推 (analogy)

```
1. ans1 = model.most_similar(positive=['劉備','諸葛亮'], negative=['曹操'])
2. print("%s: %s"%(ans1[0]))
```

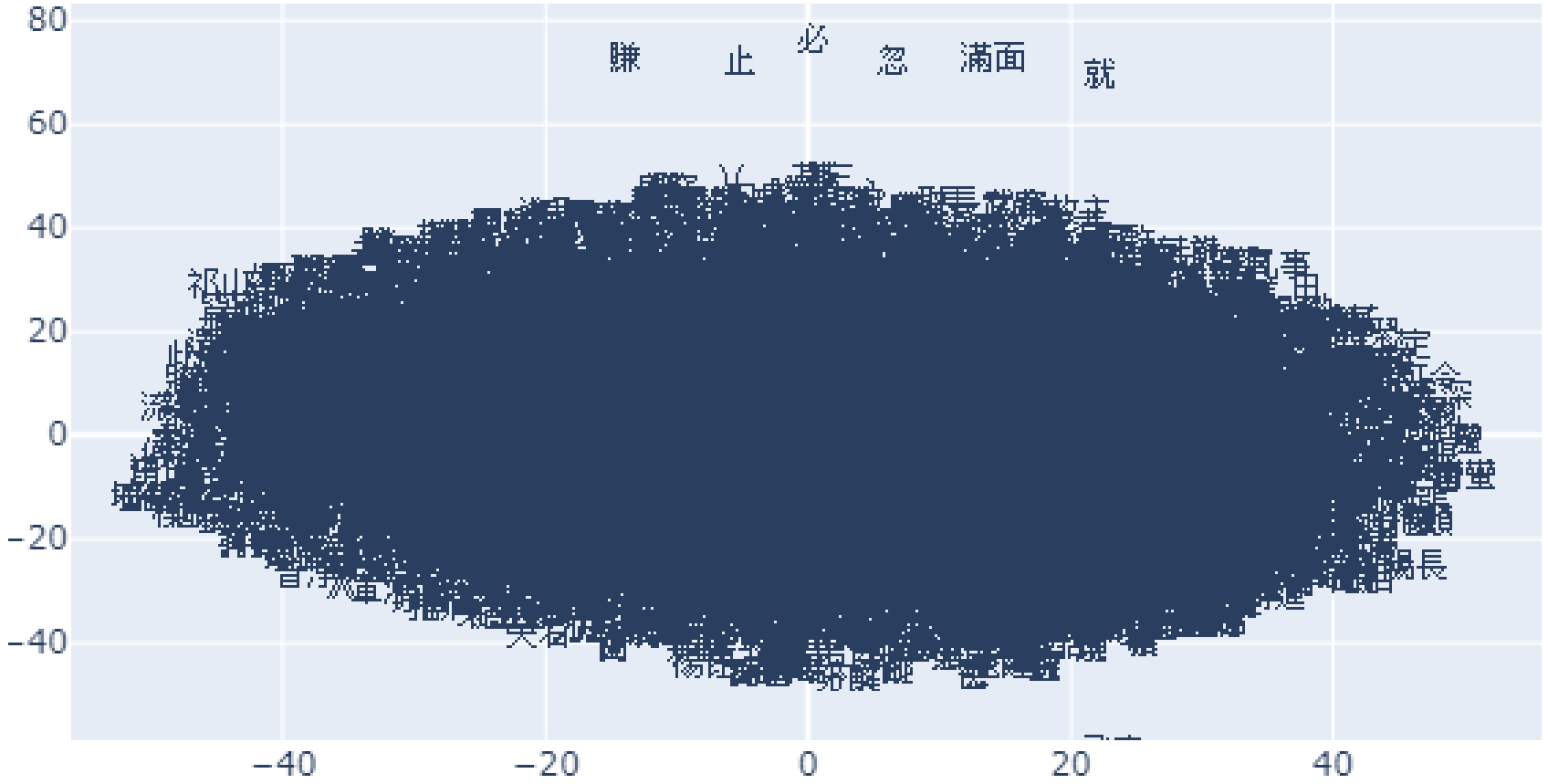
In [44]:
```
ans1 = model.most_similar(positive=['劉備','諸葛亮'], negative=['曹操'])
print("%s: %s"%(ans1[0]))
```

```
用兵: 0.32446613907814026
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `most_similar` (Method w
ill be removed in 4.0.0, use self.wv.most_similar() instead).
  """Entry point for launching an IPython kernel.
```

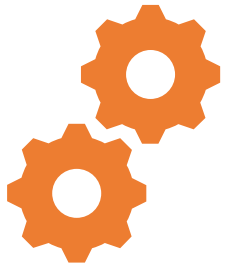
Word2vec 實作 -使用word2vec

- 向量視覺化



淺談語言模型

語言模型



模型

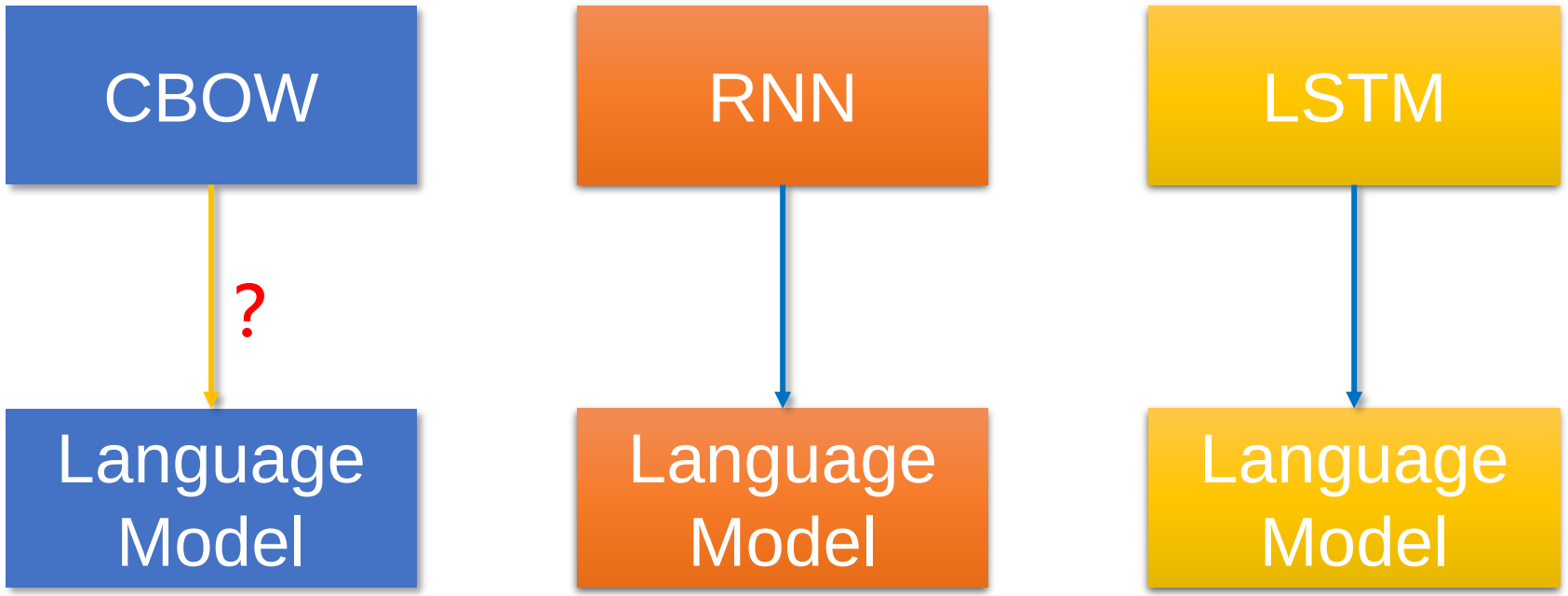
以抽象的方式表示複雜的事物

語言模型

用機率評估字詞發生的程度

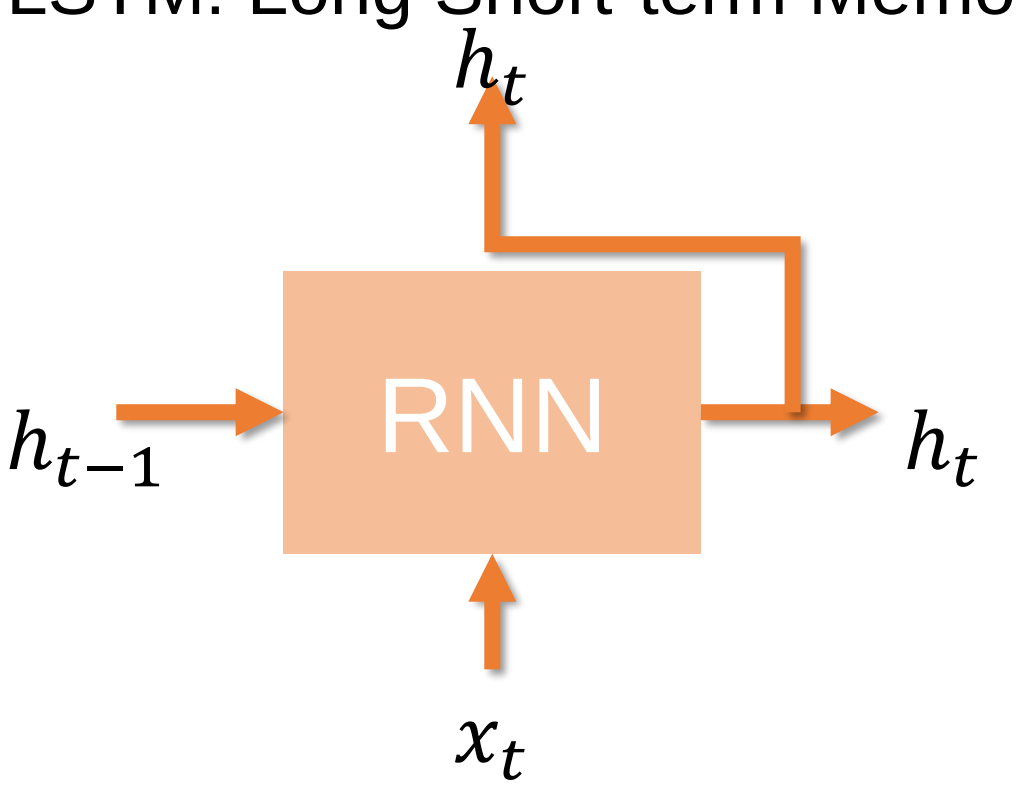
語言模型

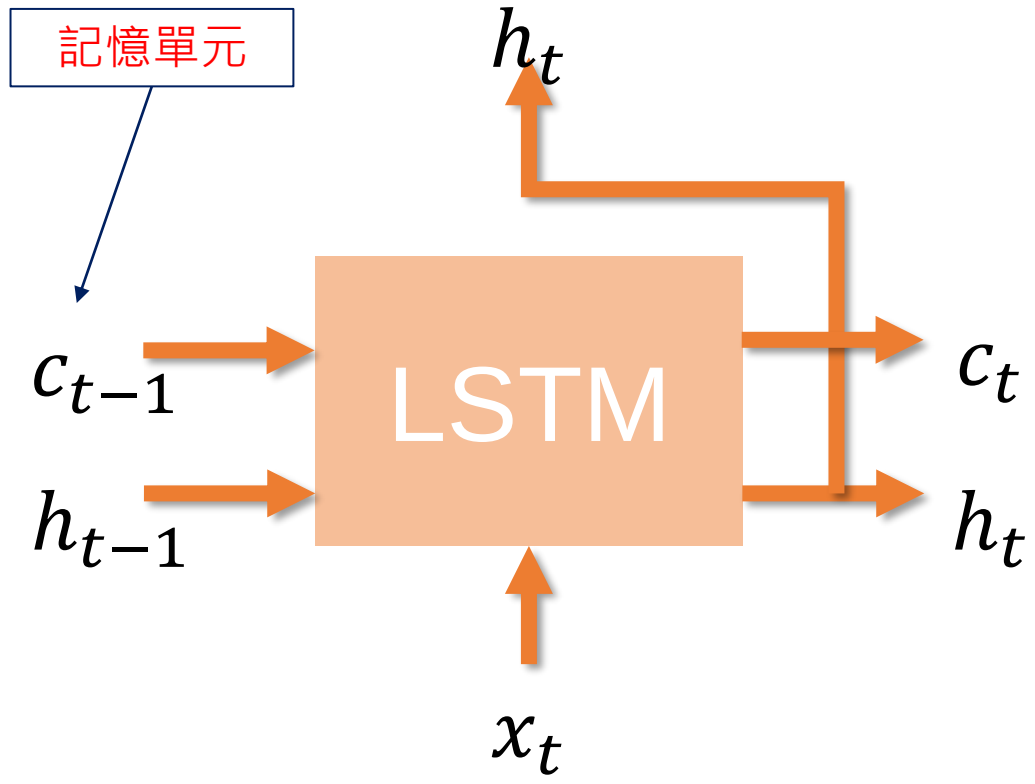
- 用機率評估字詞發生的程度



語言模型

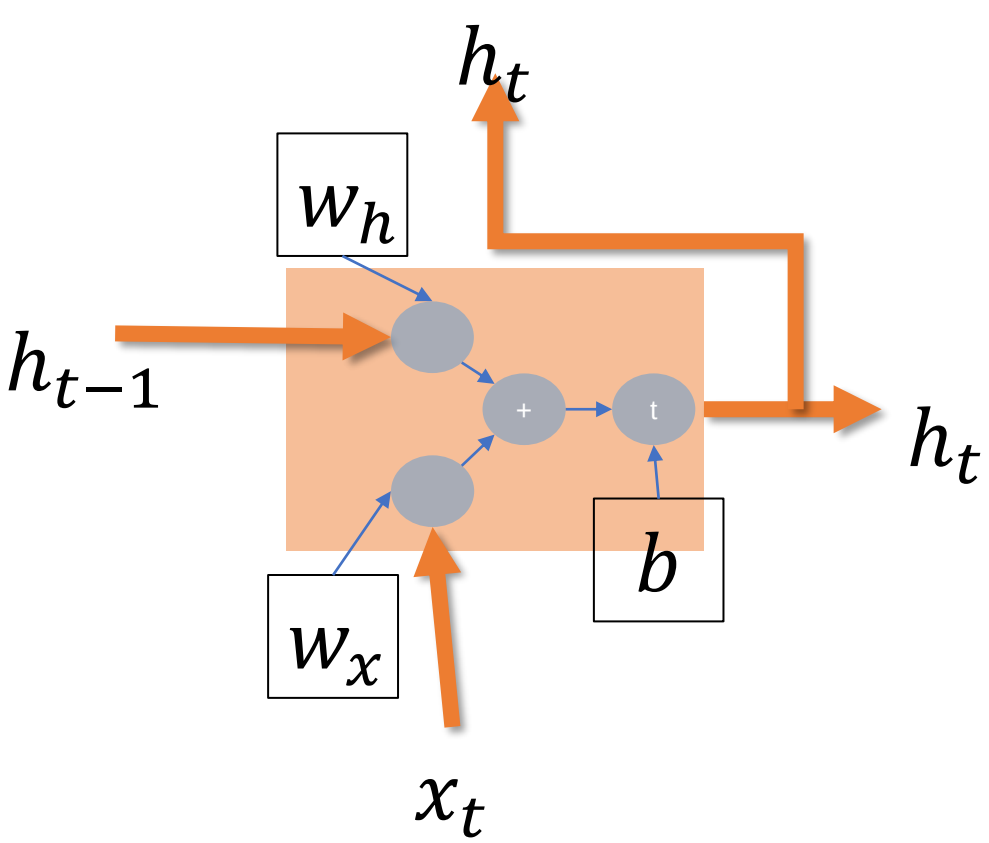
- RNN & LSTM
- RNN: Recurrent Neural Network
- LSTM: Long Short-term Memory

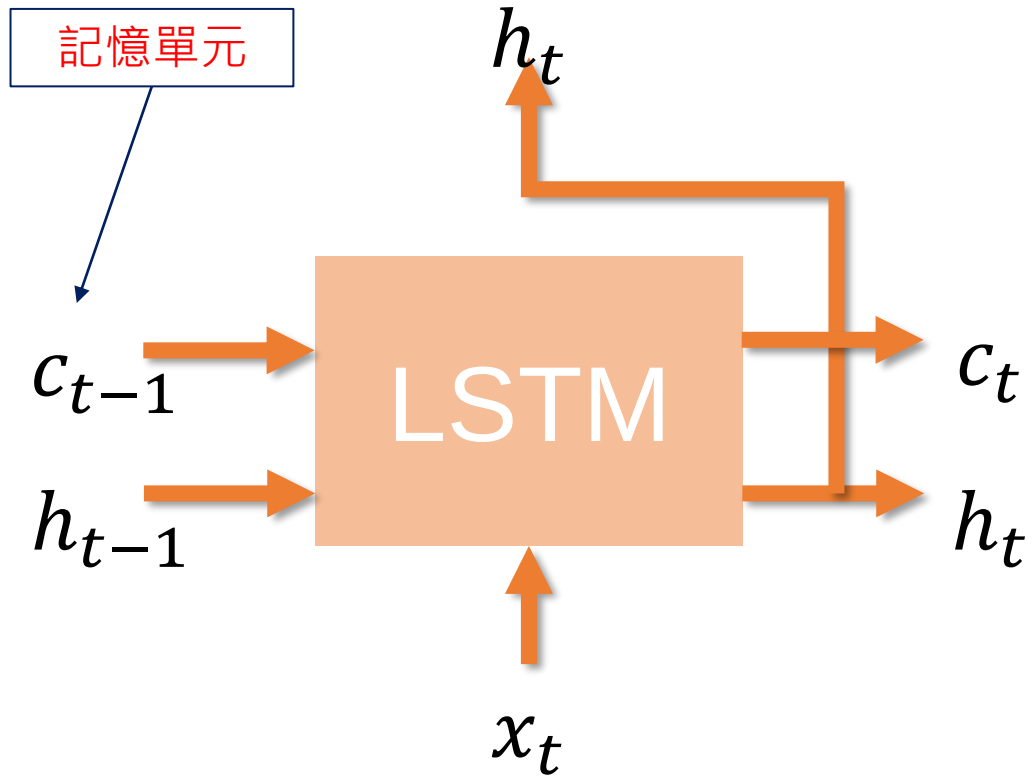




語言模型

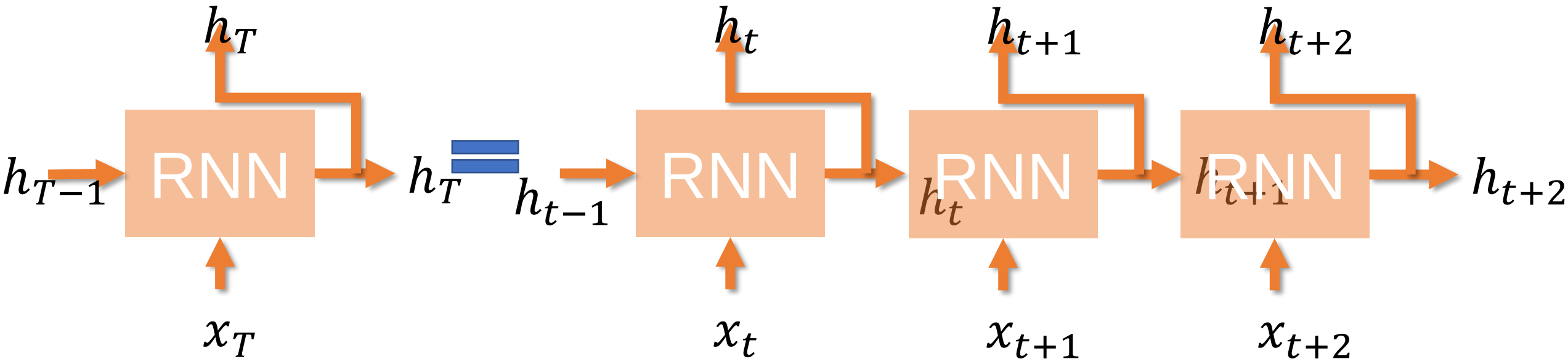
- RNN & LSTM





語言模型

- RNN: Recurrent Neural Network



語言模型

- Preceptron

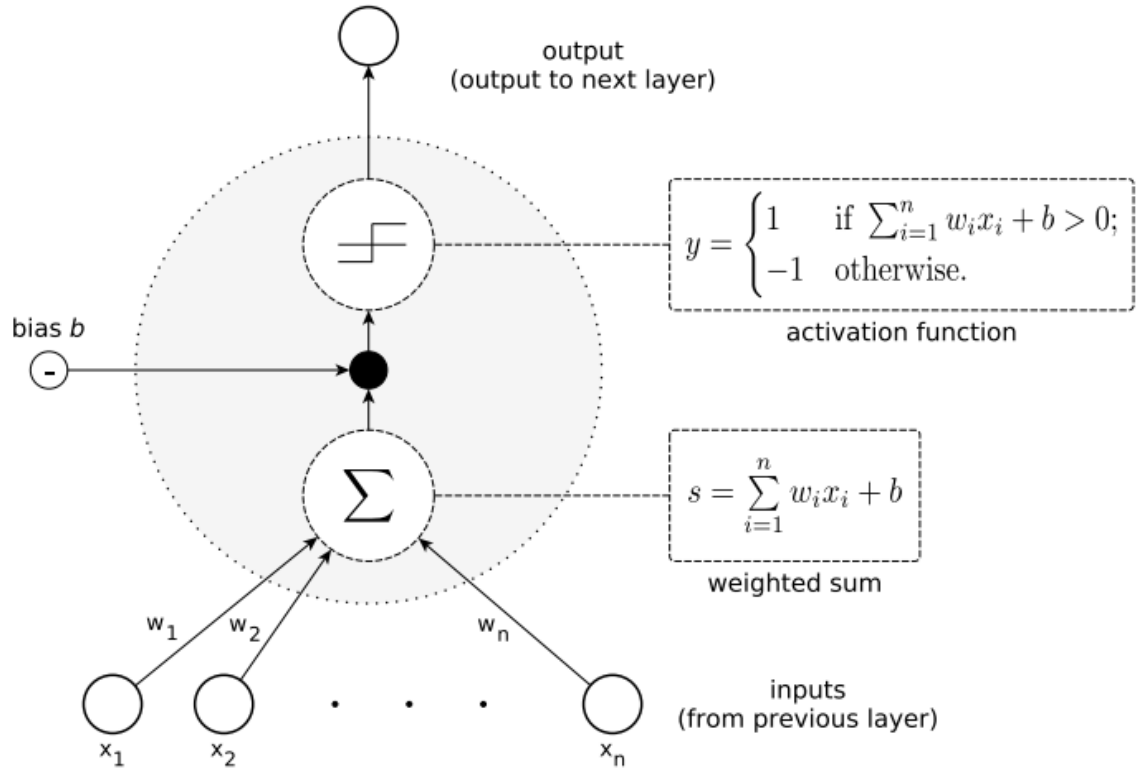
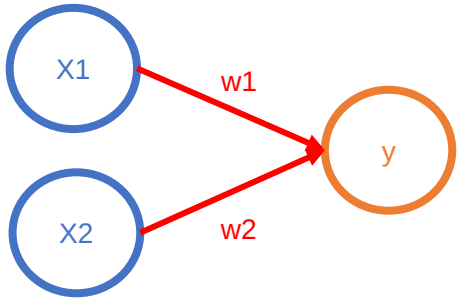


Figure 1: The general structure of the most basic type of artificial neuron, called a perceptron. Single perceptrons are limited to learning linearly separable functions.

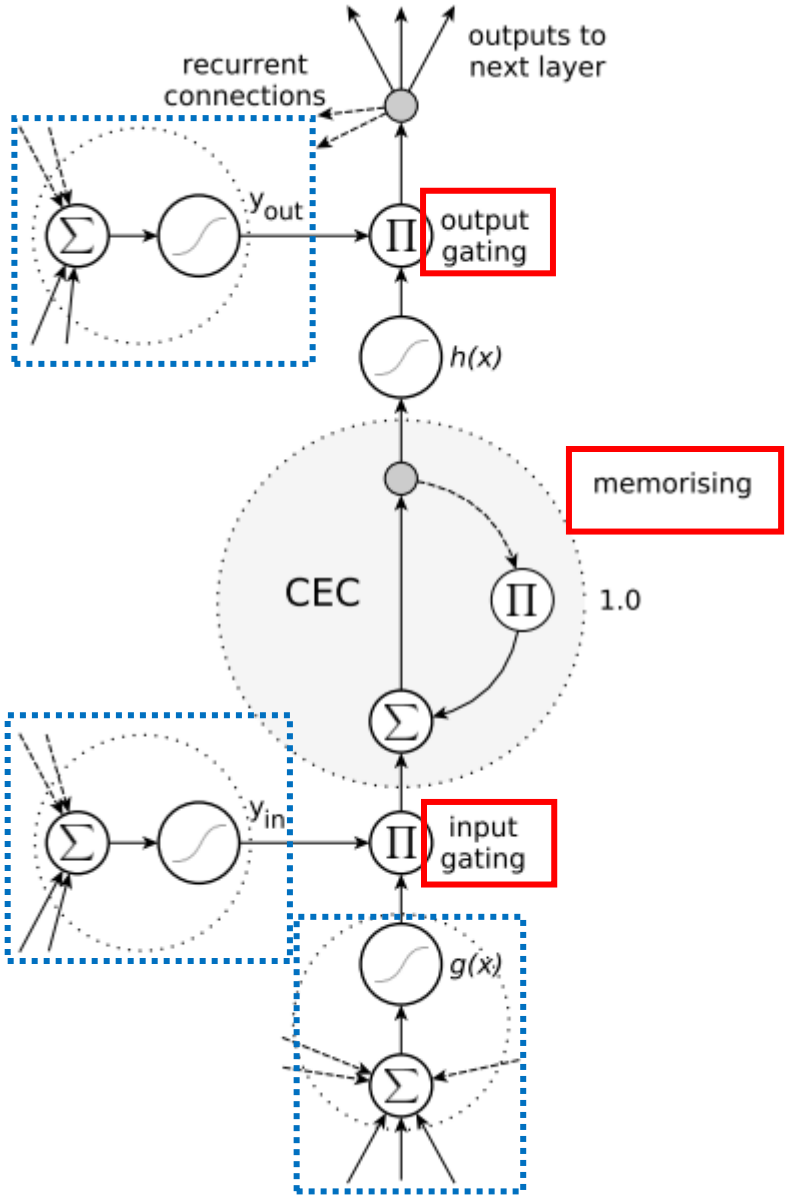


https://arxiv.org/pdf/1909.09586v1.pdf

語言模型

- LSTM: Long Short-term Memory

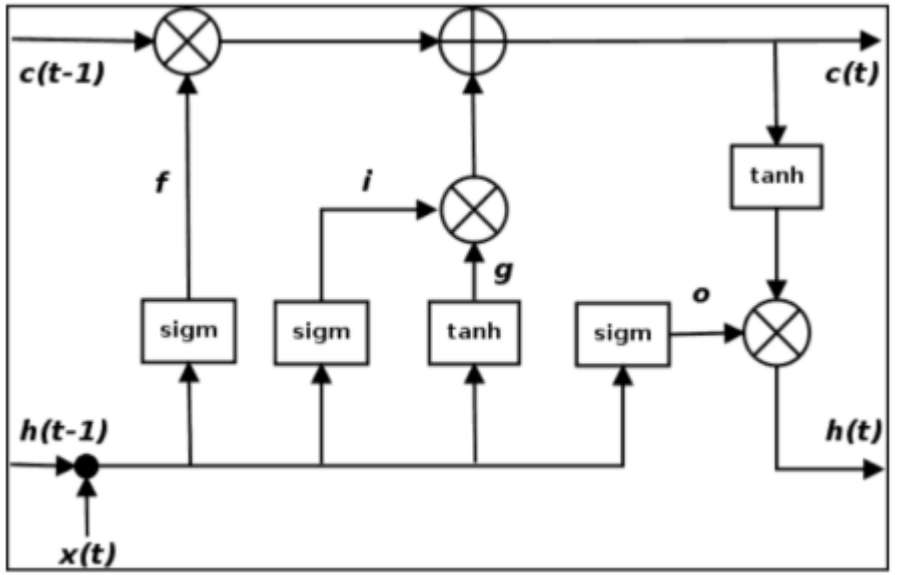
- Input gate
- Output gate
- Memory
- Forget gate



https://arxiv.org/pdf/1909.09586v1.pdf

語言模型

- LSTM: Long Short-term Memory

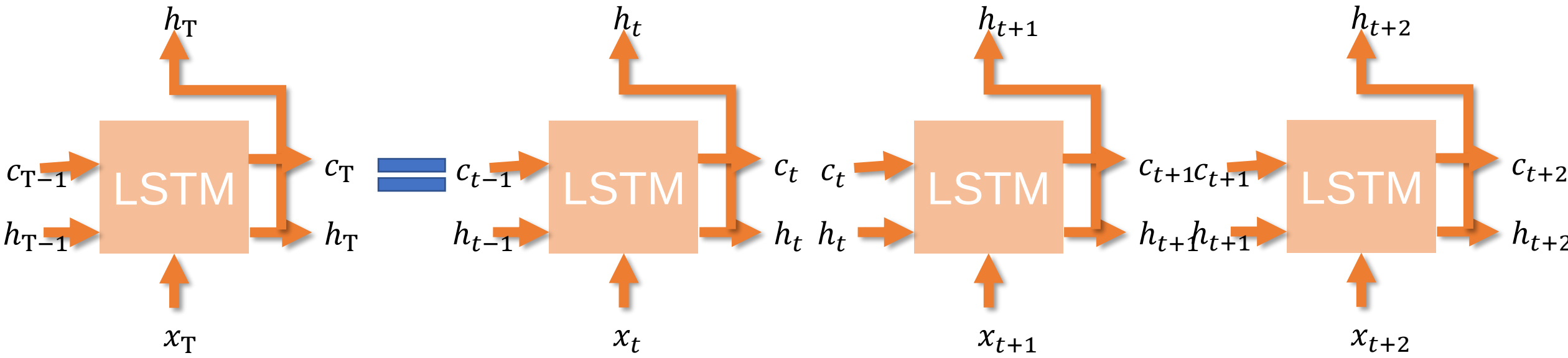


- Input gate
- Output gate
- Memory
- Forget gate

https://www.packtpub.com/big-data-and-business-intelligence/deep-learning-keras

語言模型

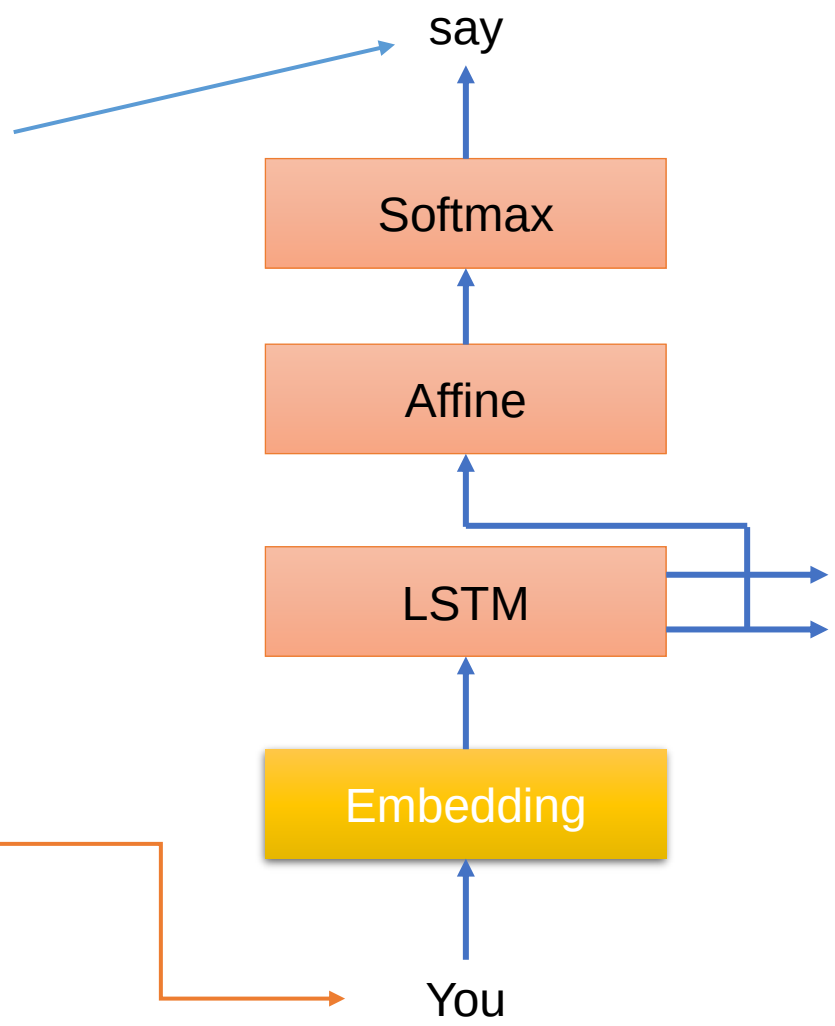
- LSTM: Long Short-term Memory



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

語言模型

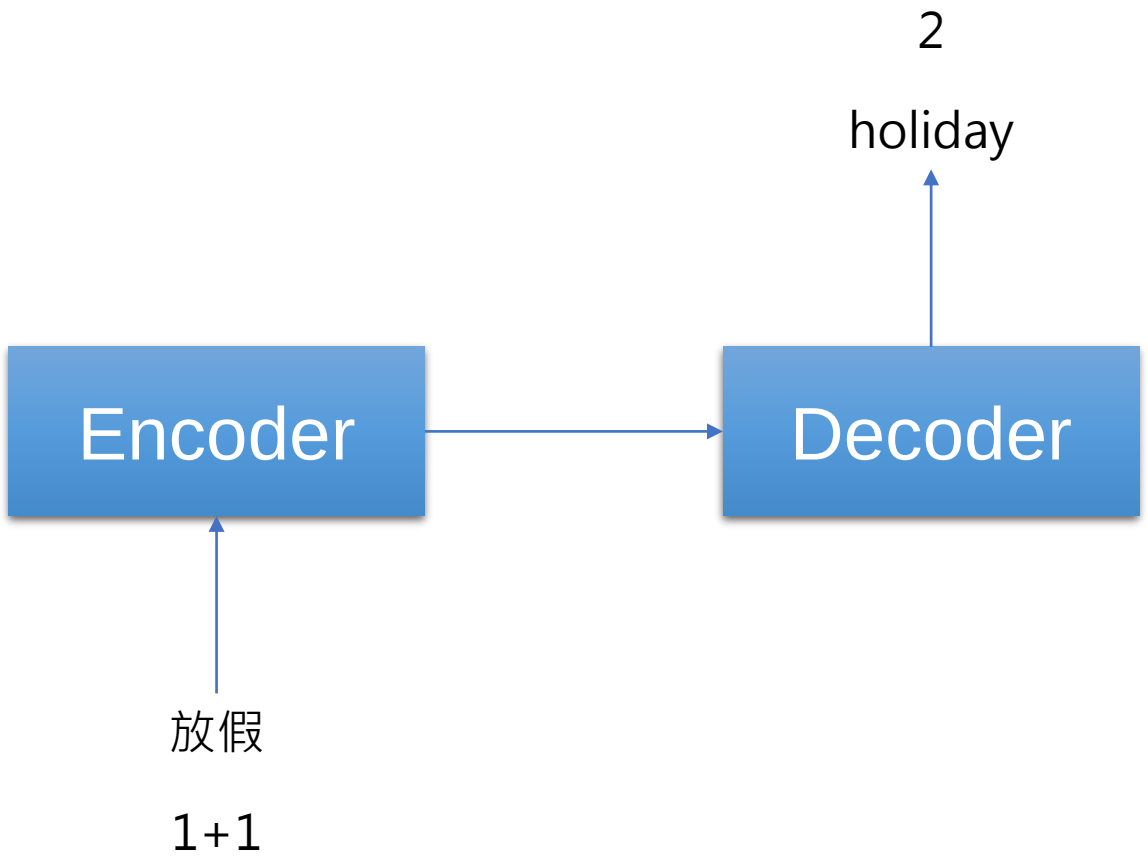
- 輸出下一個字
- 輸入上一個字



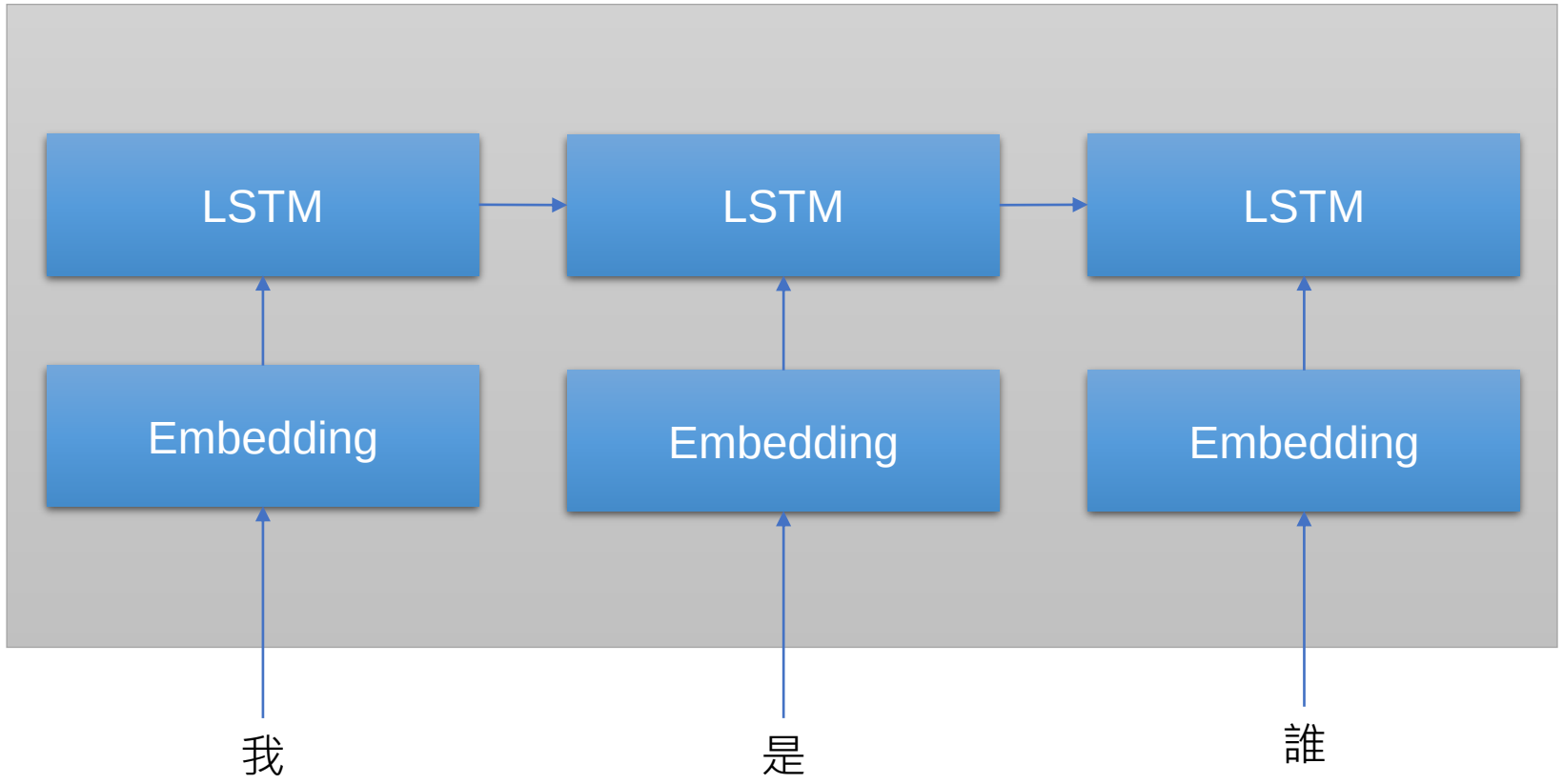
Attention

Attention - Seq2seq 簡介

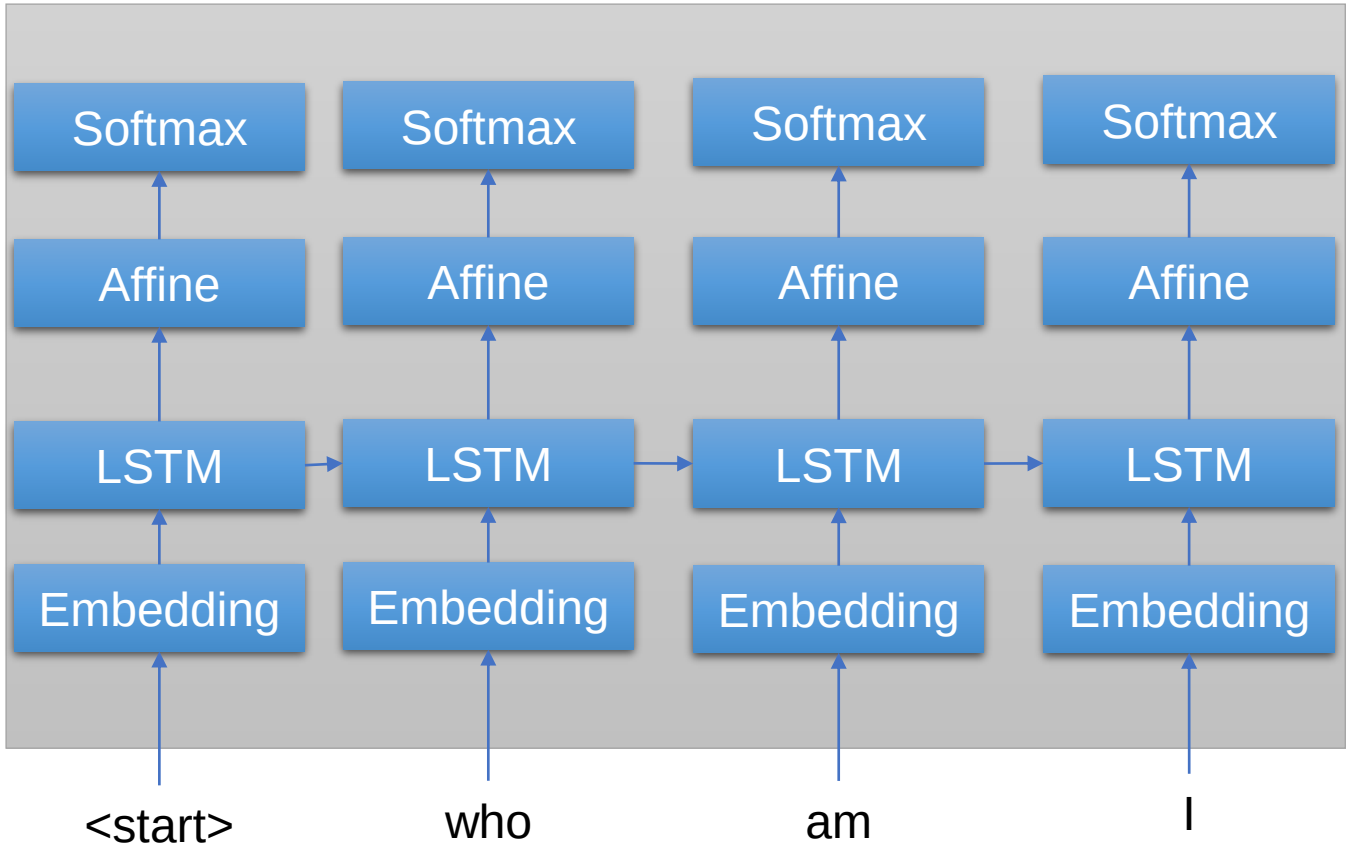
Encoder-decoder 模型



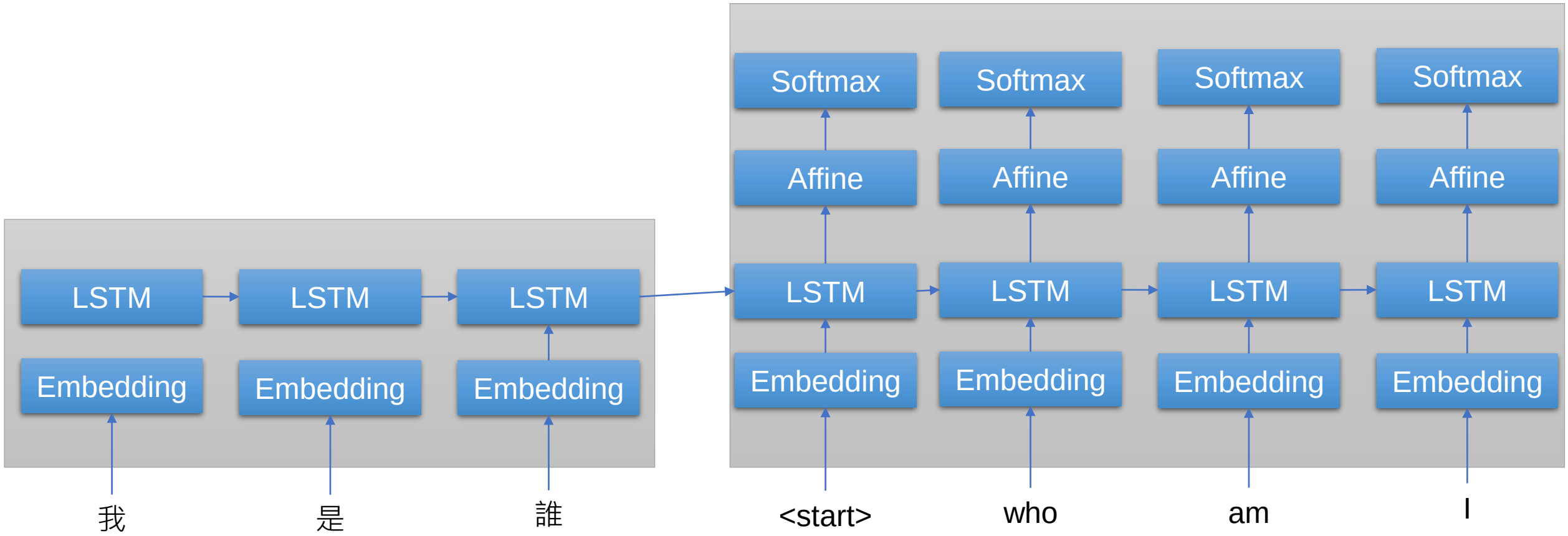
Attention – S2S Encoder 結構



Attention – S2S Decoder 結構



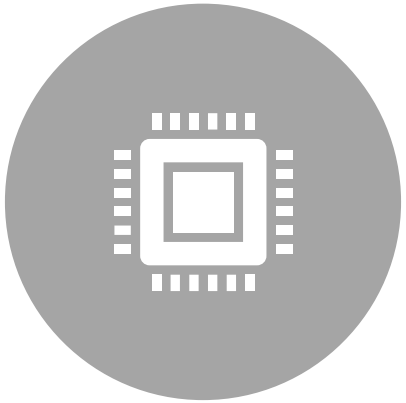
Attention – seq2seq



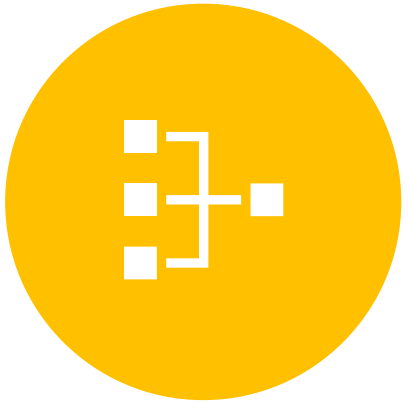
Attention

ATTENTION 是

改良型 SEQ2SEQ

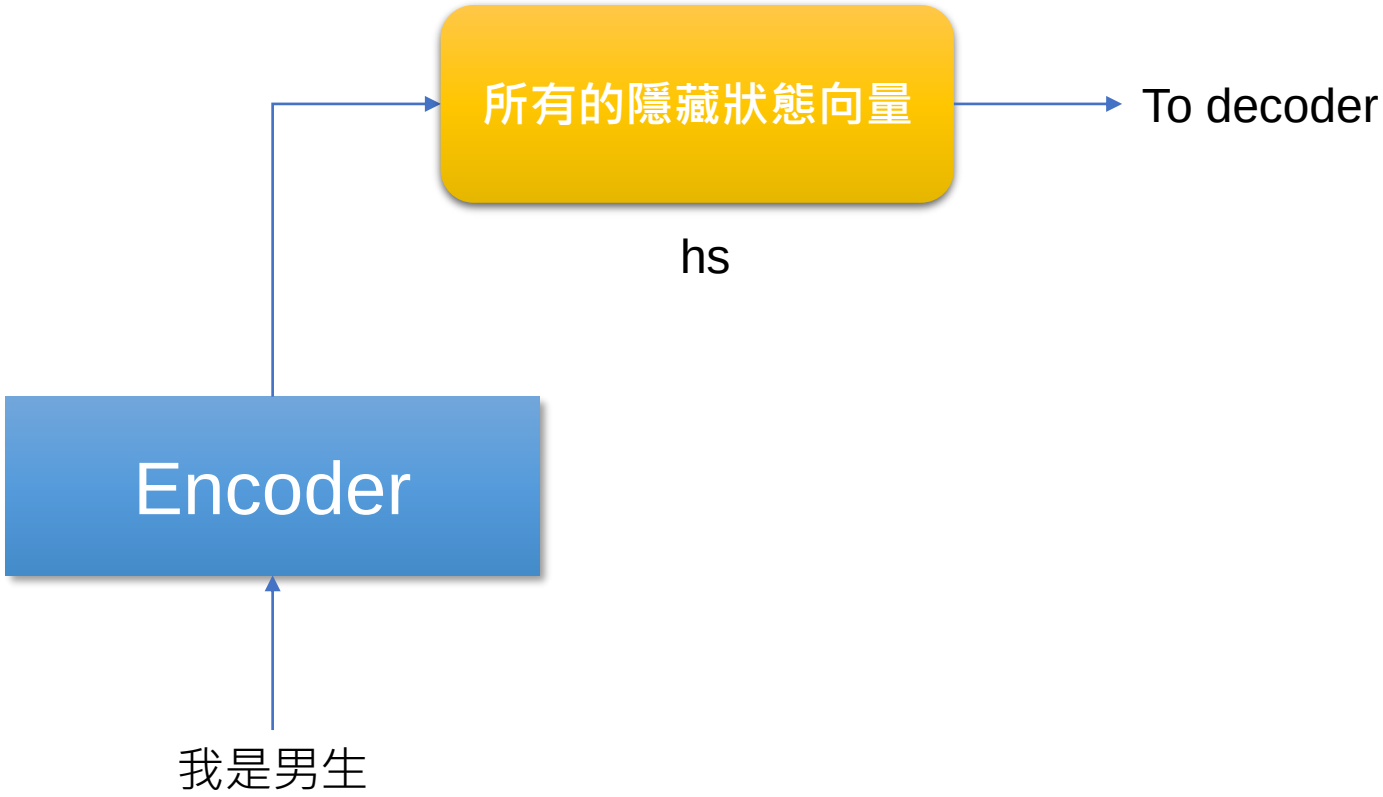


改良 ENCODER

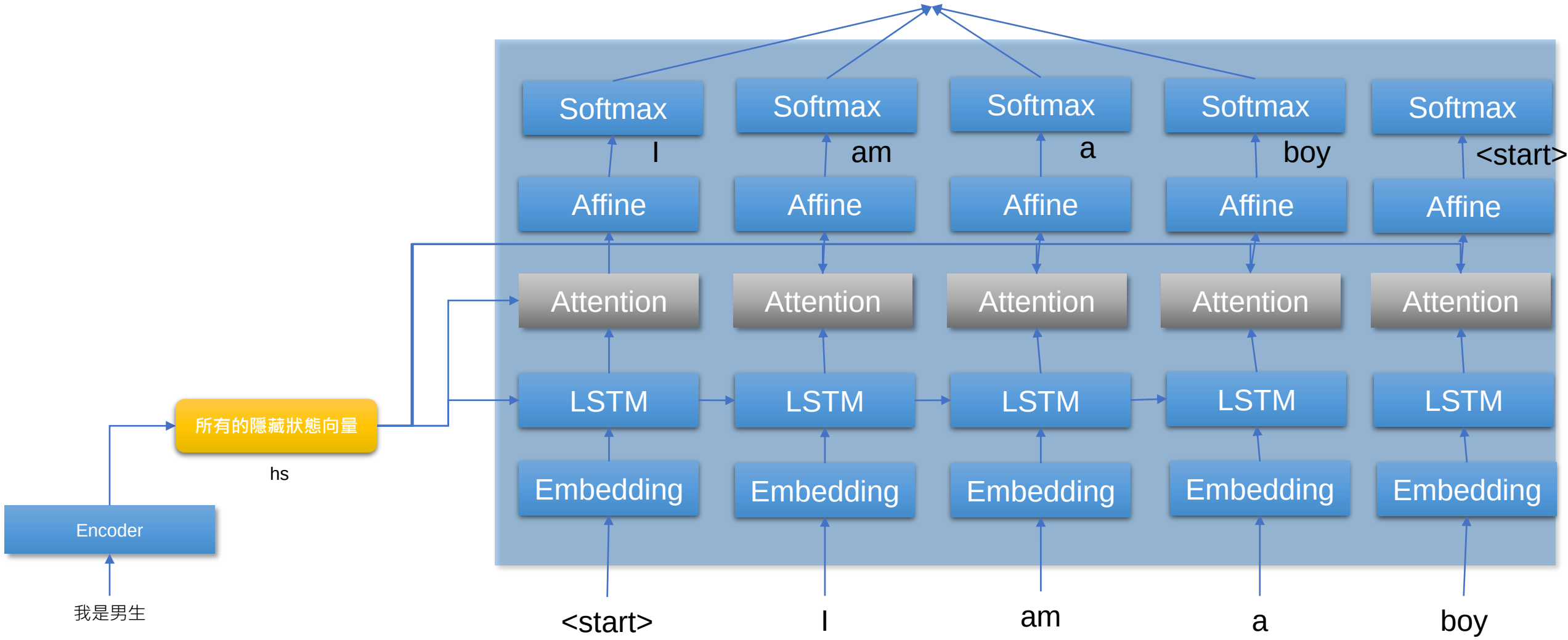


改良 DECODER

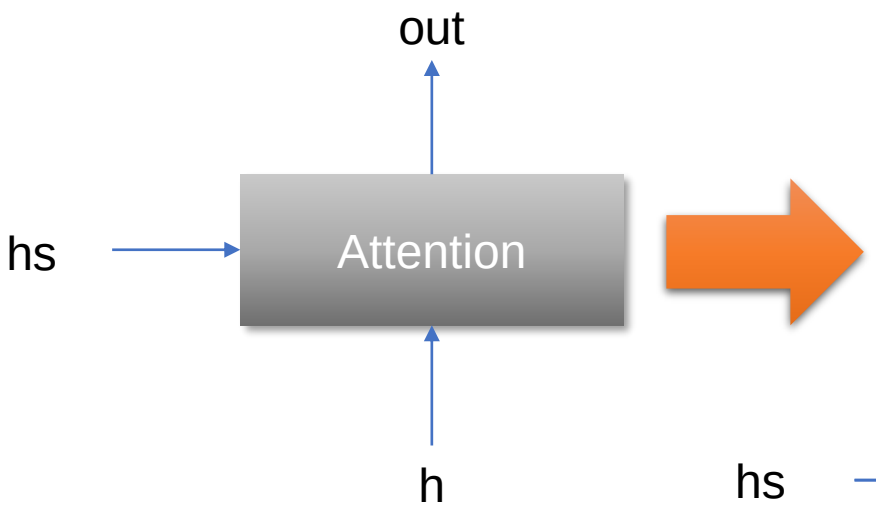
Attention – 改良 Encoder

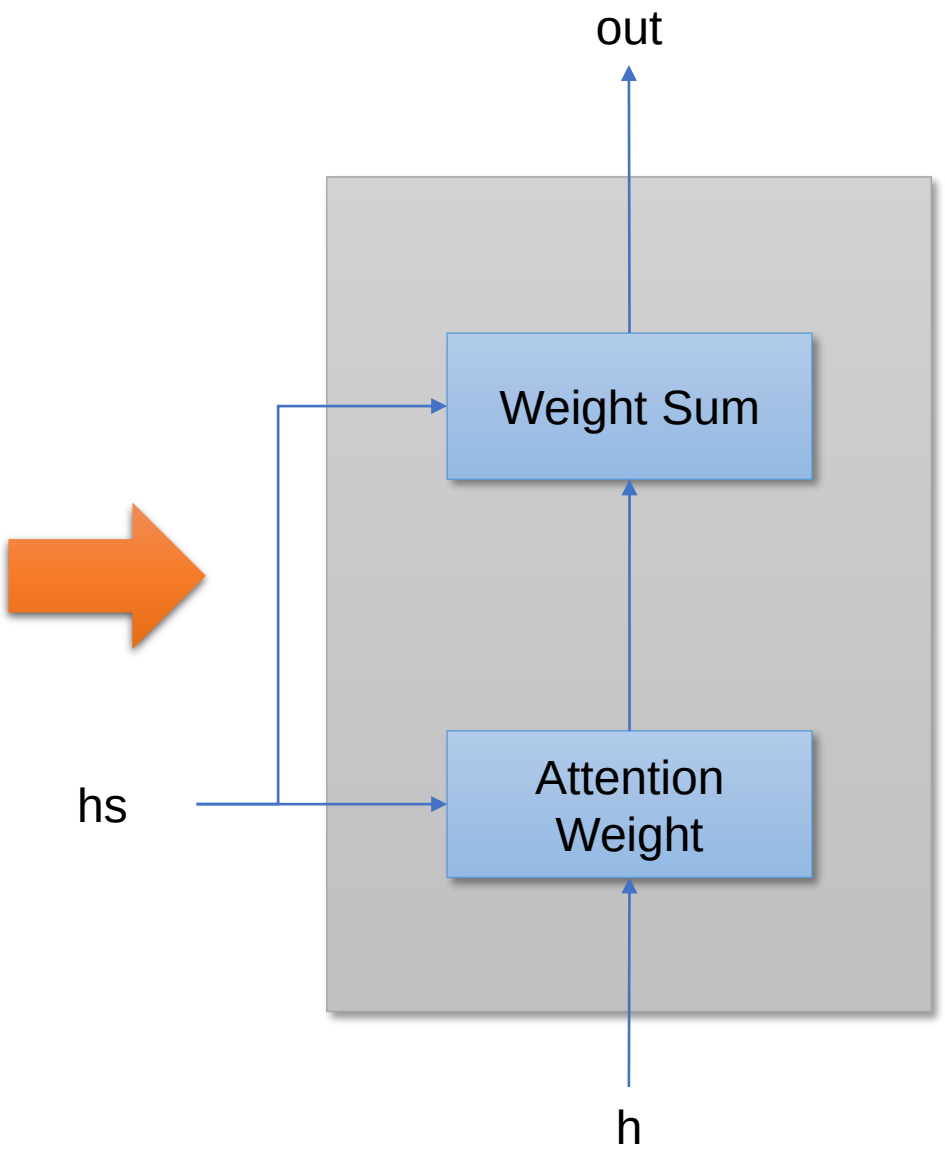


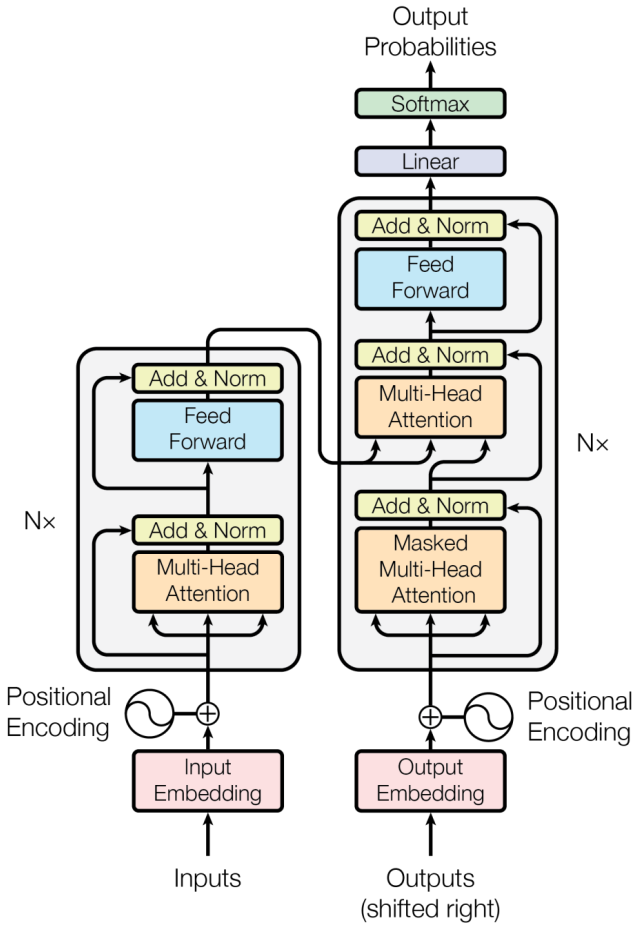
Attention – 改良 Decoder



Attention







DEMO

讓Attention 看懂日期

讓Attention 看懂日期

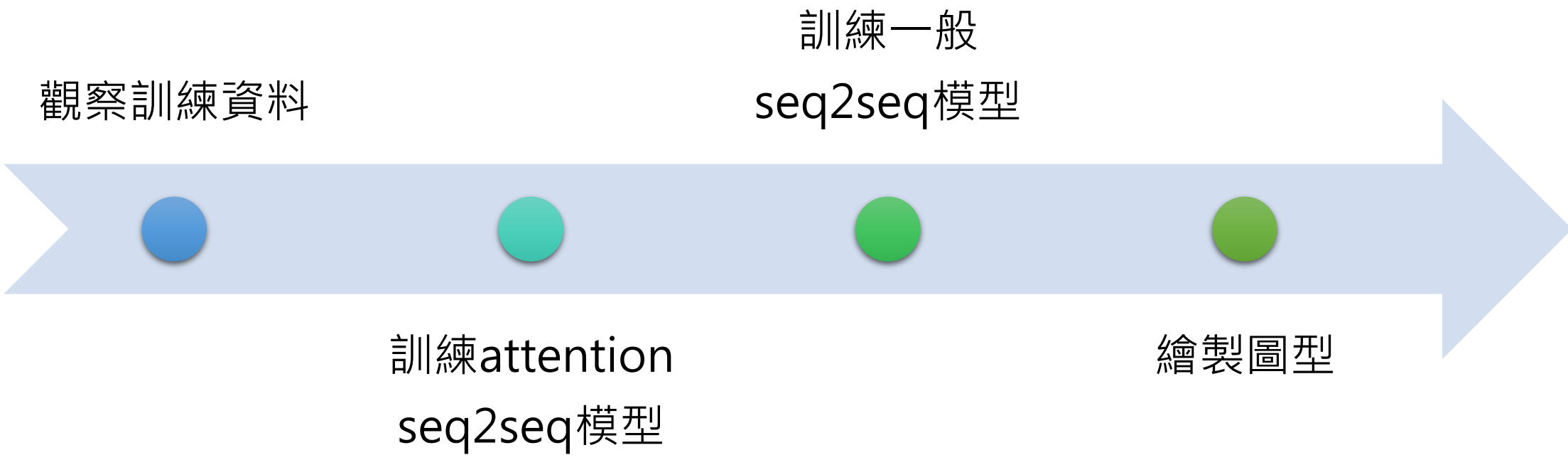
硬體規格

- 4 core CPU
- 8 G RAM
- 至少剩 20 G的磁碟空間
- GPU(不一定要有)

執行環境

- Jupyter
- Python3

讓Attention 看懂日期



DEMO

Attention 的實作參考

讓Attention 看懂日期

- 觀察資料

```
1. !head -15 dataset/date.txt
```

```
In [2]: !head -15 dataset/date.txt
```

```
september 27, 1994           _1994-09-27
August 19, 2003              _2003-08-19
2/10/93                      _1993-02-10
10/31/90                     _1990-10-31
TUESDAY, SEPTEMBER 25, 1984  _1984-09-25
JUN 17, 2013                 _2013-06-17
april 3, 1996                _1996-04-03
October 24, 1974             _1974-10-24
AUGUST 11, 1986              _1986-08-11
February 16, 2015            _2015-02-16
October 12, 1988             _1988-10-12
6/3/73                       _1973-06-03
Sep 30, 1981                 _1981-09-30
June 19, 1977                _1977-06-19
OCTOBER 22, 2005             _2005-10-22
```

讓Attention 看懂日期

- 建立 Attention model

```
# 載入資料
(x_train, t_train), (x_test, t_test) = sequence.load_data('date.txt')
char_to_id, id_to_char = sequence.get_vocab()


# 反轉輸入內容
x_train, x_test = x_train[:, ::-1], x_test[:, ::-1]
```

讓Attention 看懂日期

- 建立 Attention model

```
# 設定超參數
vocab_size = len(char_to_id)
wordvec_size = 16
hidden_size = 256
batch_size = 128
max_epoch = 10
max_grad = 5.0
```

讓Attention 看懂日期

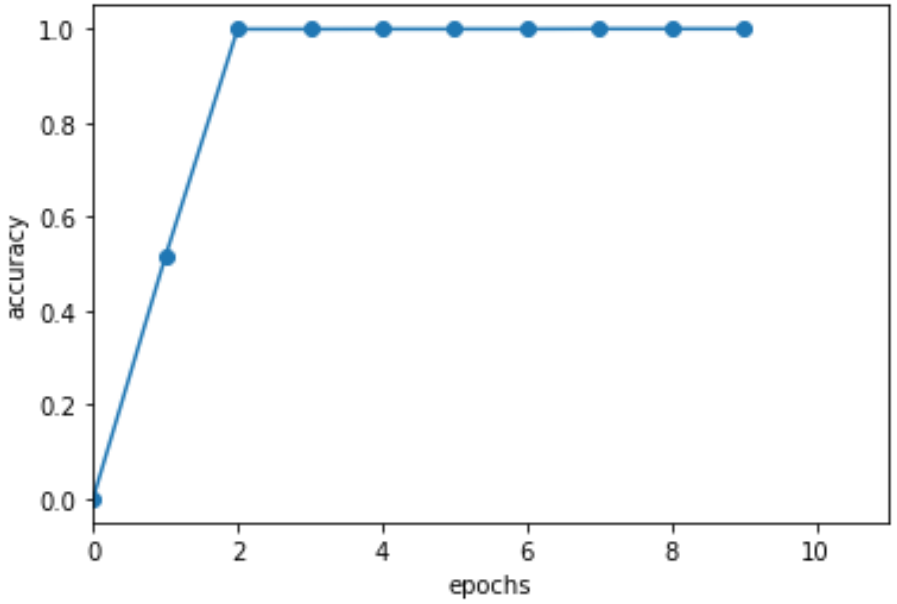
- 建立 Attention model

```
model = AttentionSeq2seq(vocab_size, wordvec_size, hidden_size)
optimizer = Adam()
trainer = Trainer(model, optimizer)
acc_list = []
for epoch in range(max_epoch):
    print(datetime.datetime.now())
    trainer.fit(x_train, t_train, max_epoch=1,
                batch_size=batch_size, max_grad=max_grad)
    correct_num = 0
    for i in range(len(x_test)):
        question, correct = x_test[[i]], t_test[[i]]
        verbose = i < 10
        correct_num += eval_seq2seq(model, question, correct,
                                    id_to_char, verbose, is_reverse=True)
    acc = float(correct_num) / len(x_test)
    acc_list.append(acc)
    print('val acc %.3f%%' % (acc * 100))
model.save_params()
```

讓Attention 看懂日期

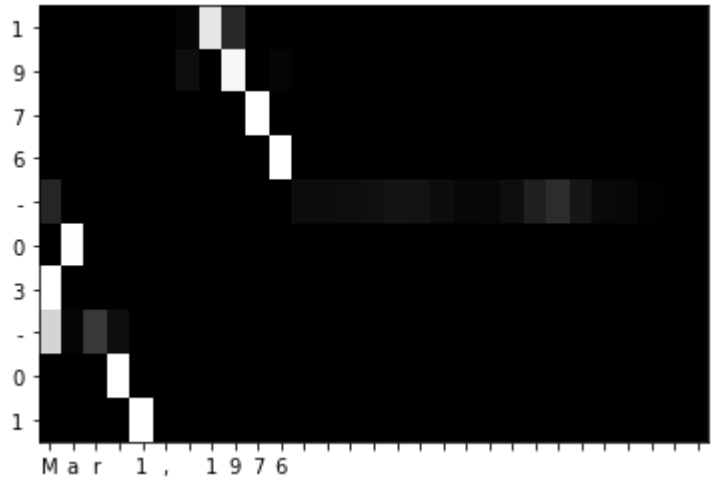
- 繪圖

```
1. x = np.arange(len(acc_list))
2. plt.plot(x, acc_list, marker='o')
3. plt.xlabel('epochs')
4. plt.ylabel('accuracy')
5. plt.ylim(-0.05, 1.05)
6. plt.xlim(0, 11)
7. plt.show()
```

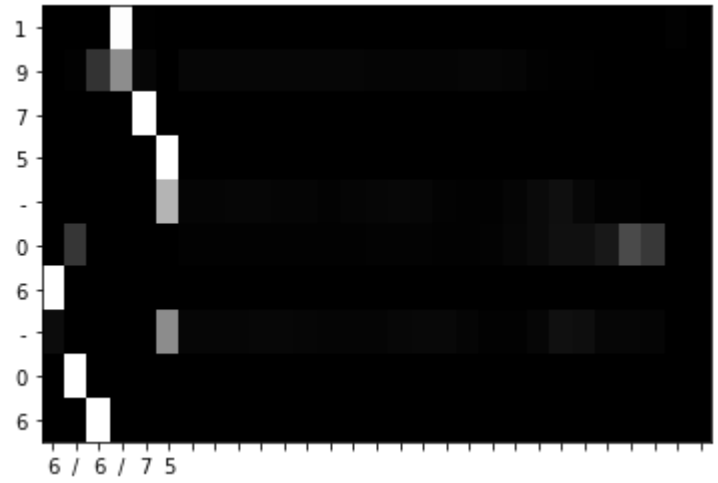


讓Attention 看懂日期

- 觀察訓練結果



學習 Mar 1, 1976 -> 1976-03-01



學習 6/6/75 -> 1975-06-06

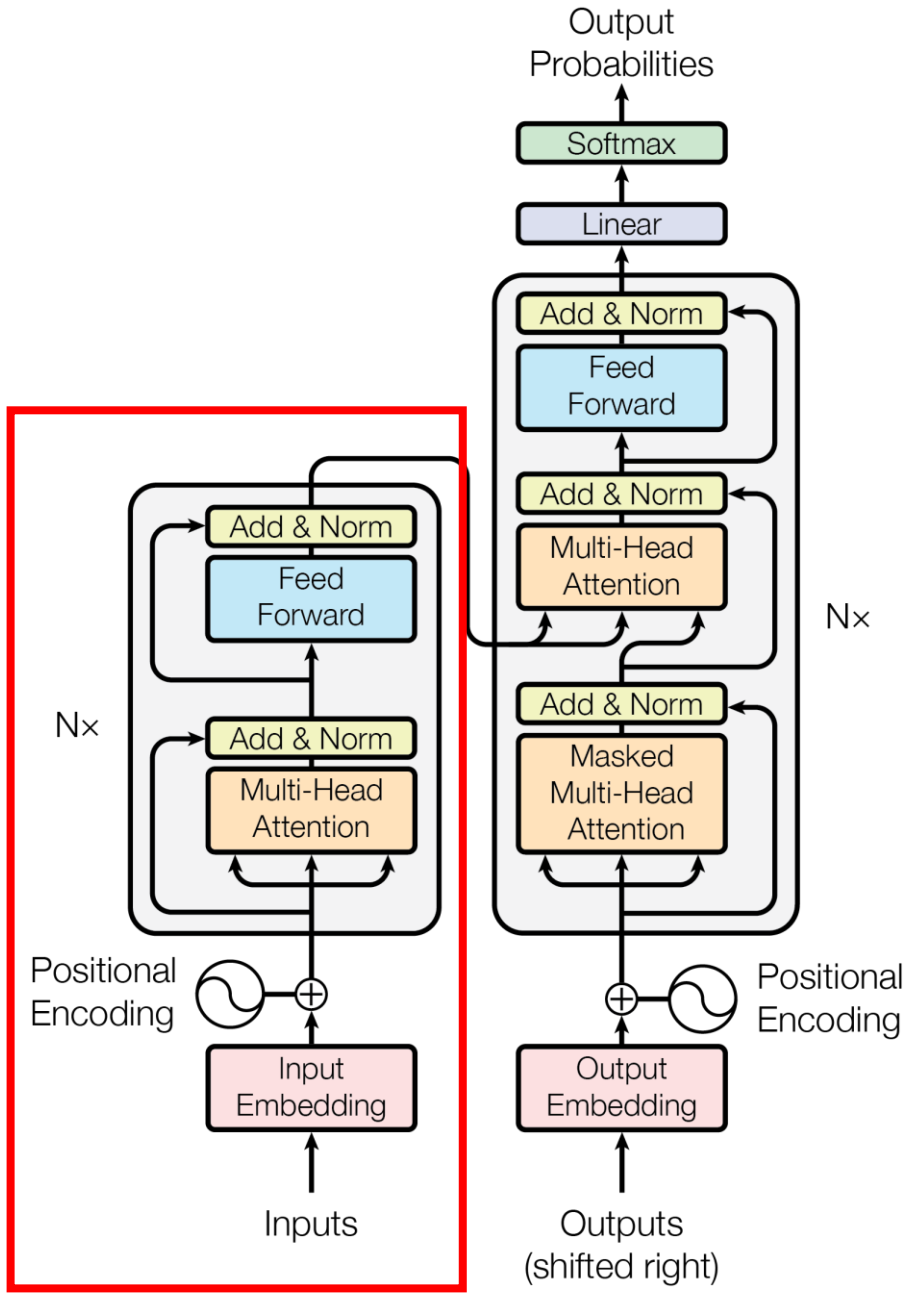
BERT

BERT 介紹

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- 論文 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding 於2018年發表，那年成為NLP的轉折年
- 2019 發表了第二版論文
- BERT模型打破了多個關於模型如何處理自然語言的記錄。在描述該模型的論文發布後不久，該團隊還開源了該模型的代碼，並提供了已經在大量數據集上進行了預訓練的模型下載版本。

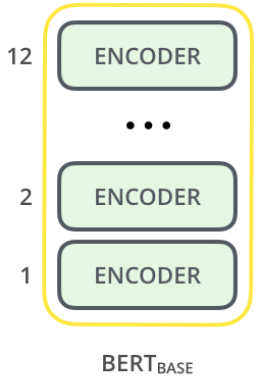
BERT 介紹

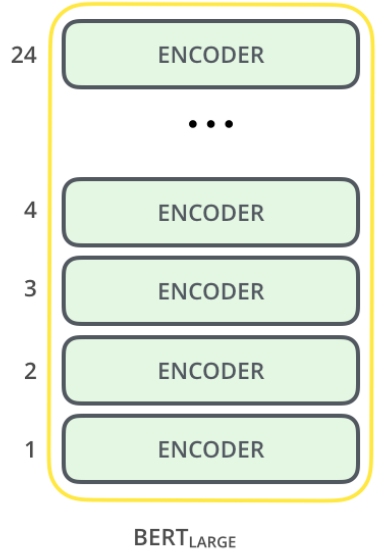
- Transformer encoder stacks
- BERT 是語言表示的一種預訓練方法，利用大量的資料集預先訓練一般的語言理解 (language understanding)



BERT 介紹

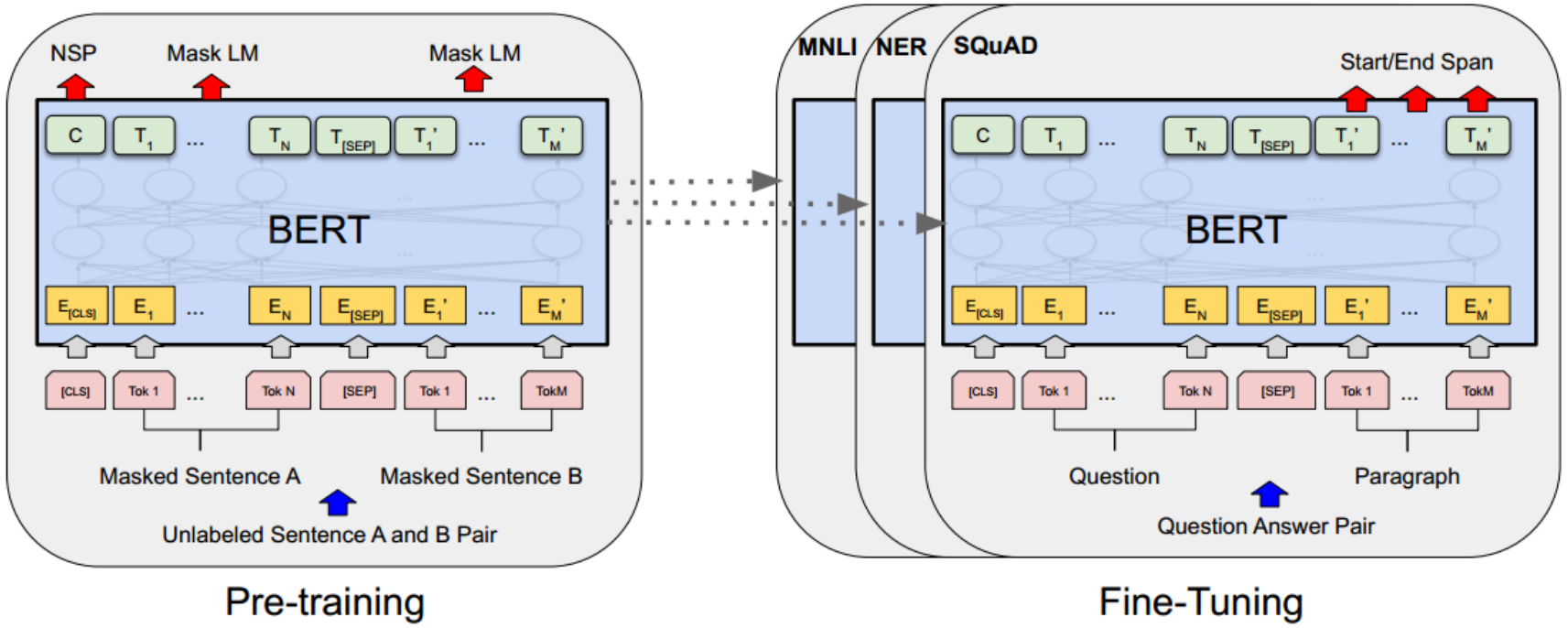
- BERT-Base
  - 12-layer, 768-hidden, 12-heads, 110M parameters
- BERT-Large
  - 24-layer, 1024-hidden, 16-heads, 340M parameters





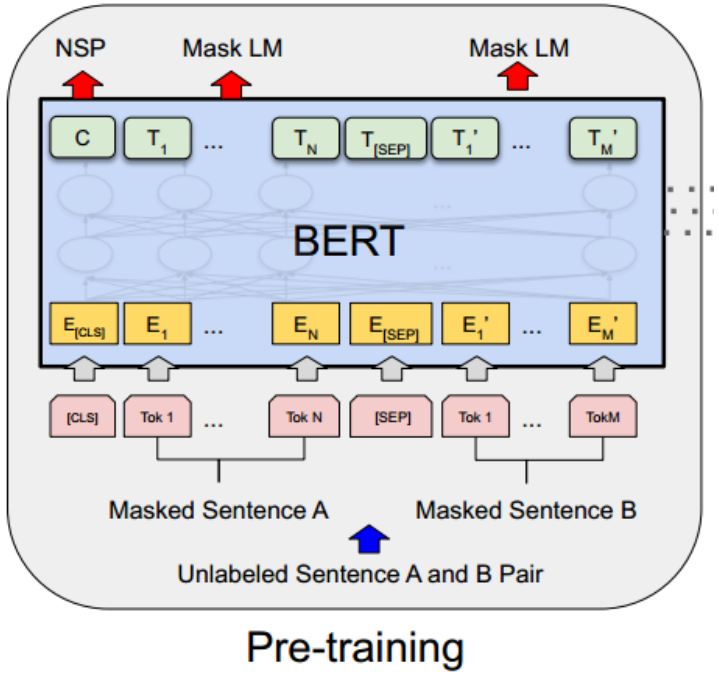
BERT 介紹

- 遷移學習(transfer learning)
  - 先訓練一個通用模型(pre-training)
  - 再依照用途做 fine-tuning



BERT 介紹

- Pre-training
  - 做克漏字填充(Masked LM)
  - 下一句預測(NSP)



Pre-training

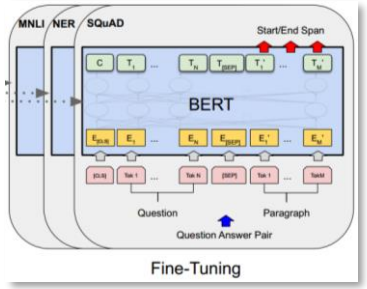
- Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:
- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

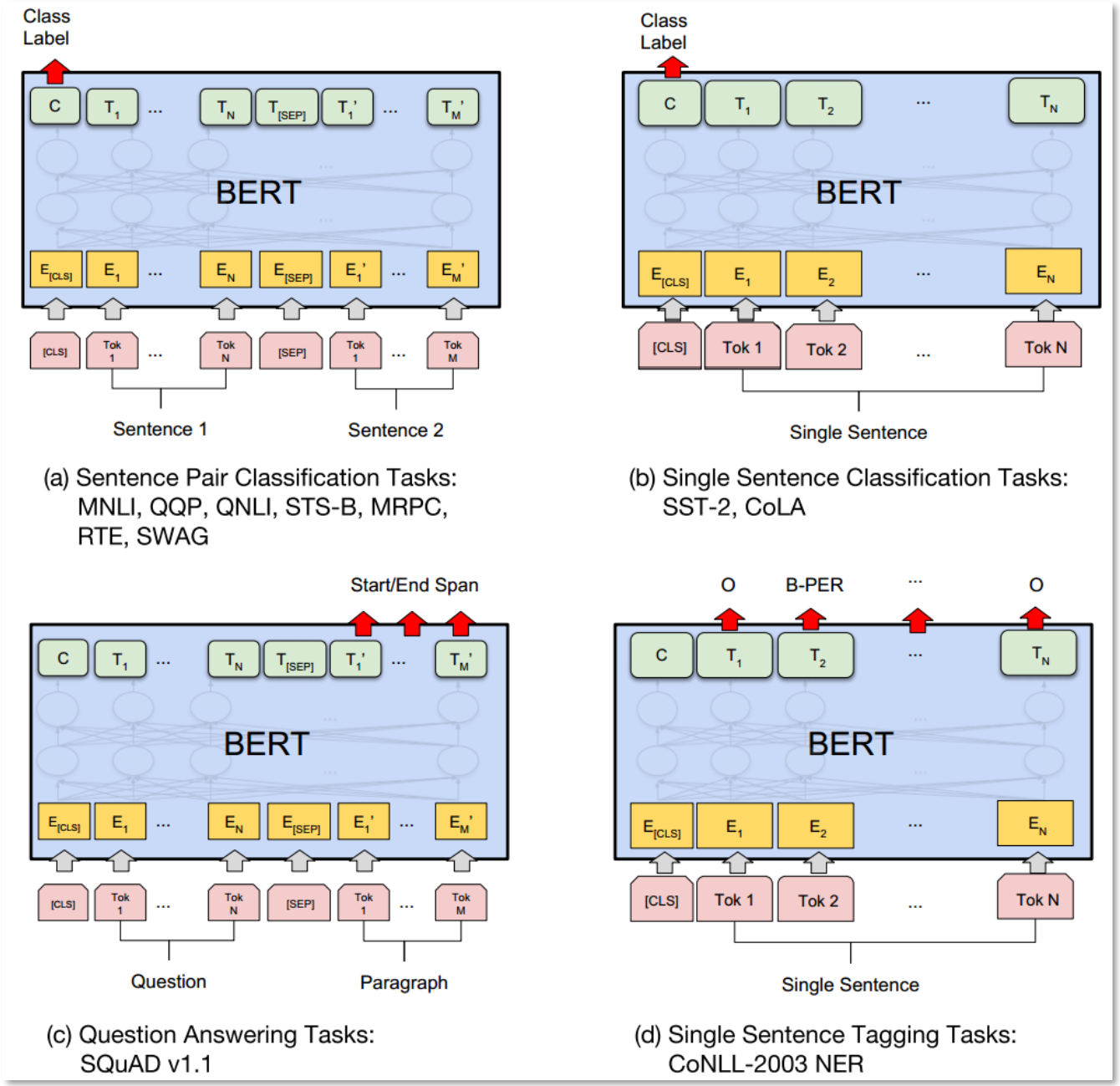
Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

BERT 介紹

- Fine-tuning

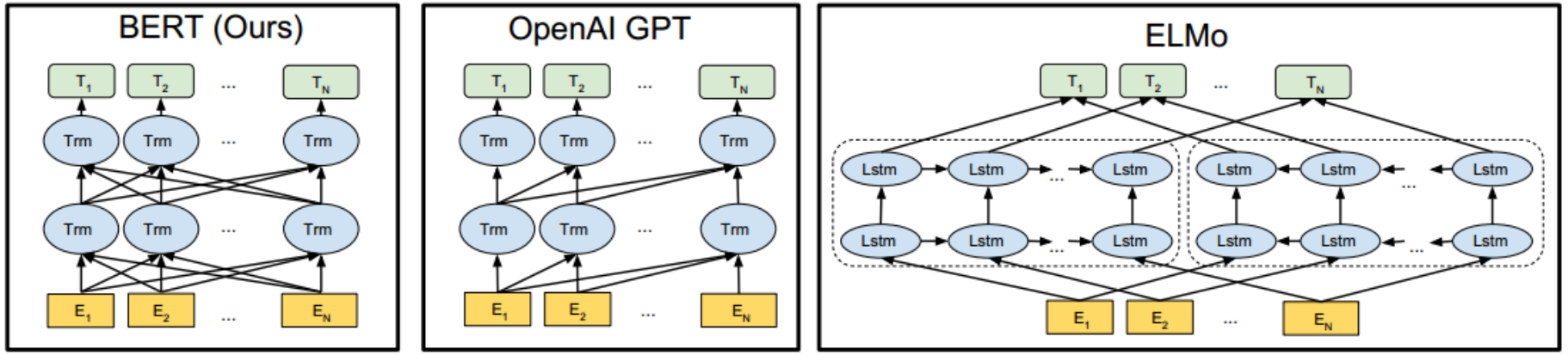


(a)成對句子分類任務	(b)單一句子分類任務
(c) 問答任務	(d) 單一句子標籤任務



BERT 介紹

- BERT 是雙向預訓練的架構
- GPT 是單向
- ELMo 是個別方向訓練聯集架構



GPT-2

GPT-2 介紹

- Generative Pre-Training
- GPT-2（是GPT的繼承者），僅經過訓練即可預測40GB的互聯網文本中的下一個單詞。由於我們擔心該技術的惡意應用，因此我們不會發布經過訓練的模型。作為負責任公開的一項實驗，我們將發布一個供研究人員進行實驗的小得多的模型以及一份技術論文。
- GPT-2是基於大型 Transformer-based 的語言模型，**具有15億個參數**，並在**800萬個網頁的數據集上進行了訓練**

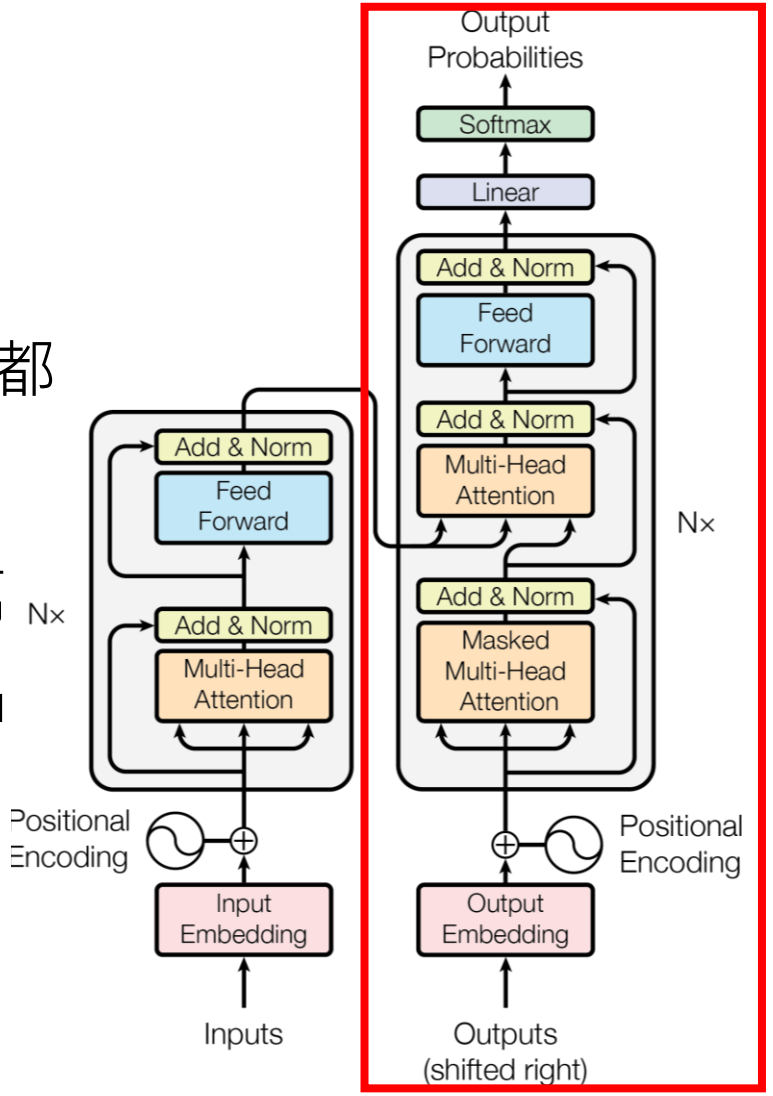
LM

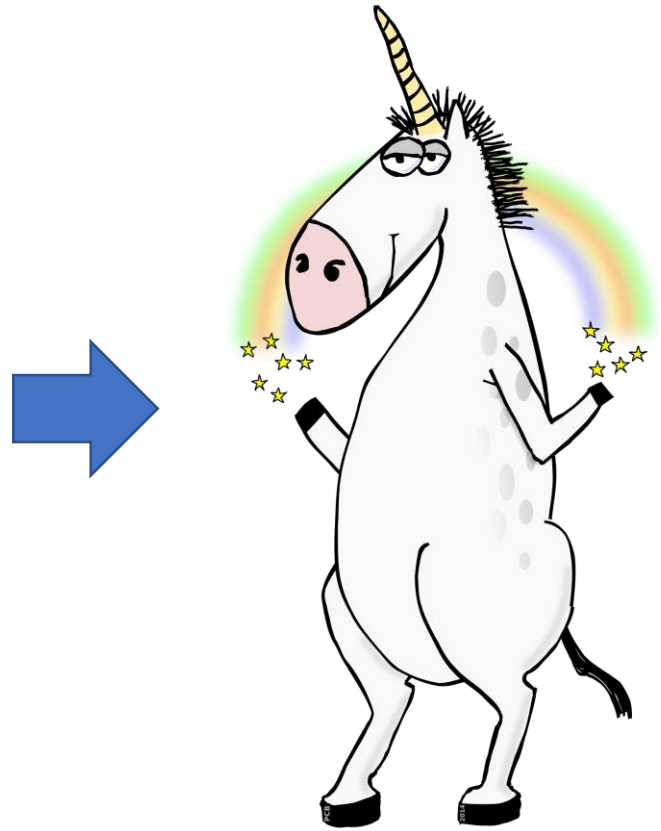
- GPT-2的訓練目標很簡單：**根據文本中的所有先前單詞，預測下一個單詞**。數據集的多樣性使這個簡單的目標包含了跨不同領域的許多任務的自然發生的演示。
- GPT-2是GPT的直接擴展，具有超過10倍的參數，並接受了超過10倍的數據量訓練。GPT-2顯示了廣泛的功能，包括生成具有空前質量的條件合成文本樣本的能力，我們在模型中使用輸入來填充模型並讓其生成冗長的延續。此外，GPT-2優於其他在特定領域（如Wikipedia，新聞或書籍）上訓練的語言模型，而無需使用這些特定於領域的訓練數據集。在諸如答疑，閱讀理解，總結和翻譯之類的語言任務上，GPT-2開始使用原始文本來學習這些任務，而沒有使用特定於任務的訓練數據。

來源: https://www.goodfreephotos.com/vector-images/unicorn-with-rainbow-vector-clipart.png.php

GPT-2 介紹

- GPT2 架構上承襲 Transformer，不過GPT2 訓練模型用的數據及參數都異常的龐大。
- 訓練數據：使用從 800 萬個網頁爬來的 40 GB 高品質內容
- 模型參數：15 億參數





來源: https://www.goodfreephotos.com/vector-images/unicorn-with-rainbow-vector-clipart.png.php

GPT-2 介紹

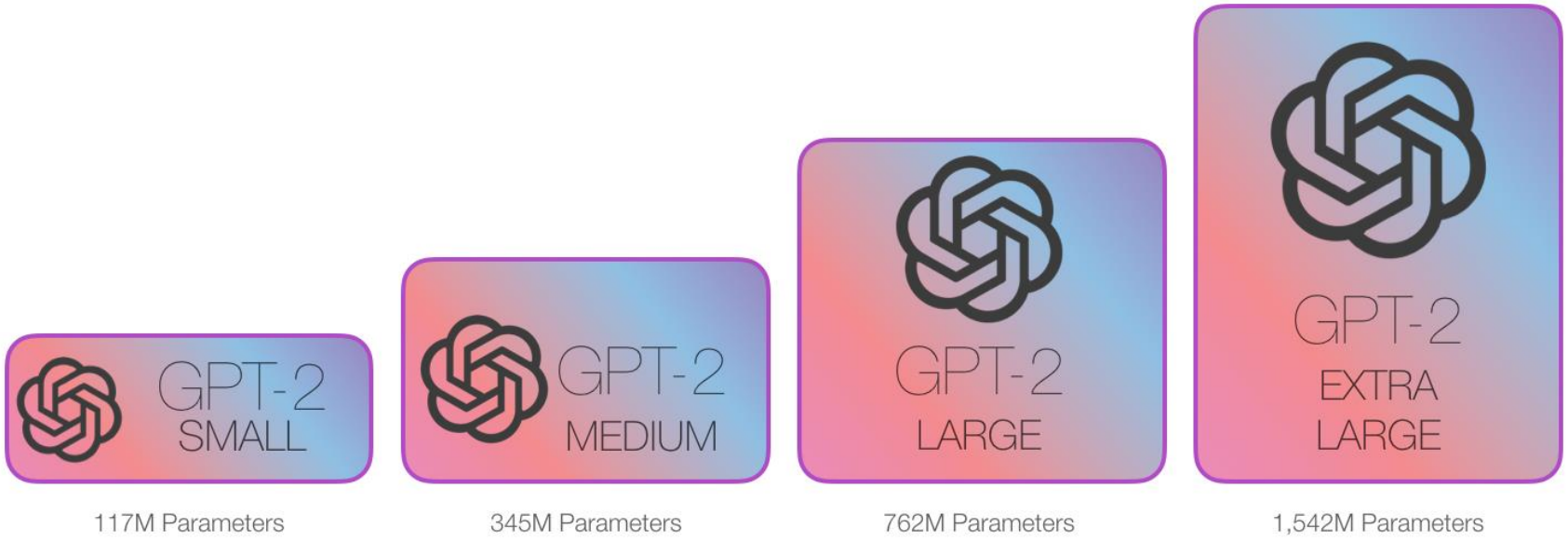
SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez. Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns. While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic." Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America. While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, "In South America, such incidents seem to be quite common."

系統提示（人工編寫）	*在一個令人震驚的發現中，科學家發現了一群獨角獸，它們生活在安第斯山脈一個偏遠的，以前未被開發的山谷中。對於研究人員而言，更令人驚訝的是，獨角獸會說完美的英語。*
模型完成（機器編寫，10次嘗試）	這位科學家以其獨特的號角Ovid的獨角獸命名了這個種群。這些四角銀白色獨角獸以前是科學界所不知道的。 現在，在將近兩個世紀之後，終於解決了引發這種奇怪現象的謎團。 拉巴斯大學的進化生物學家豪爾赫·佩雷斯（JorgePérez）博士和幾位同伴在安第斯山脈發現一個沒有其他動物或人類的小山谷時正在探索。佩雷斯注意到該山谷似乎是一個天然噴泉，周圍環繞著兩個岩石和銀色雪峰。 佩雷斯和其他人然後冒險進了山谷。佩雷斯說：“當我們到達一個山峰的頂部時，水看起來是藍色的，上面有一些晶體。” 佩雷斯和他的朋友們驚訝地看到了獨角獸群。可以從空中看到這些生物，而不必動太多就可以看到它們-它們是如此接近以至於可以觸摸它們的角。 在檢查這些怪異生物時，科學家發現這些生物也講一些相當普通的英語。佩雷斯說：“例如，我們可以看到它們具有共同的'語言'，例如方言或辯證法。” 佩雷斯博士認為，獨角獸可能起源於阿根廷，在那里人們相信這些動物是在人類到達南美洲這些地區之前居住在那裡的一群迷失人口的後代。 儘管其起源尚不清楚，但有些人認為，也許這些生物是人類和獨角獸在人類文明誕生之前相遇時創造的。佩雷斯說：“在南美，此類事件似乎很普遍。” 但是，佩雷斯還指出，要確定獨角獸是否確實是迷失的外星人種族的後裔，唯一的可能途徑就是脫氧核糖核酸。這位科學家說：“但是他們似乎能夠用英語很好地交流，我認為這是進化的跡象，或者至少是社會組織的變化。”

來源 : https://openai.com/blog/better-language-models/

GPT-2 介紹

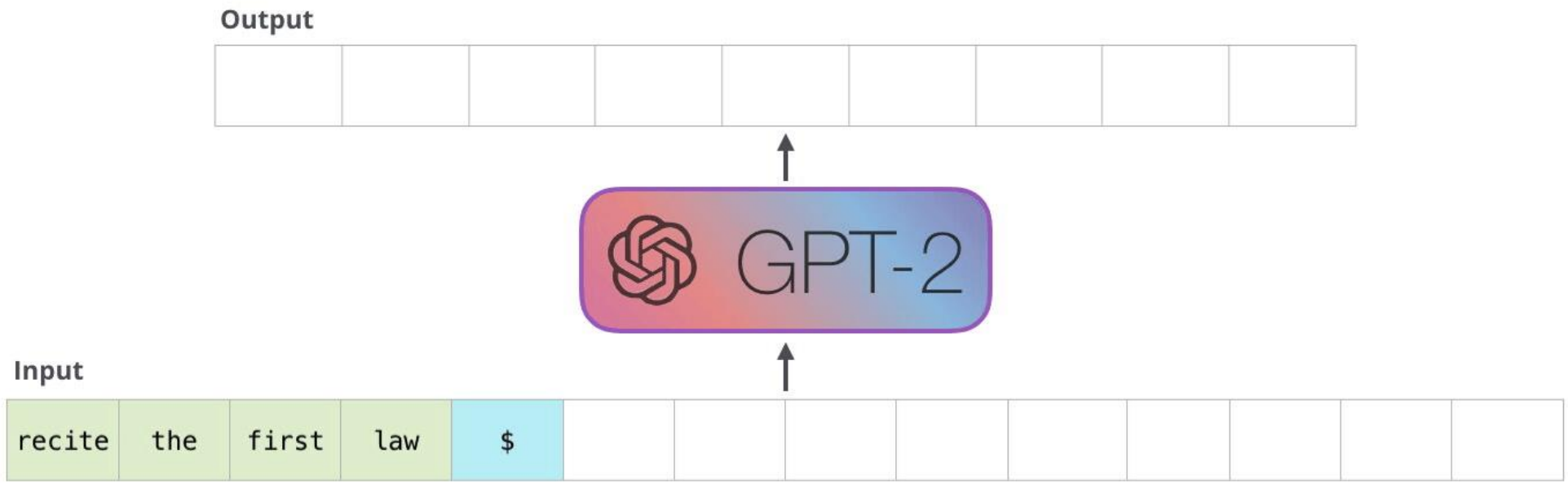
- 不同版本的GPT-2，只差異在訓練的資料量與參數
- 目前公開的有 GPT-2 small, medium 與 large



來源 : https://jalammar.github.io/illustrated-gpt2/

GPT-2 介紹

- GPT-2 根據文本中的所有先前單詞，預測下一個單詞



來源 : https://jalammar.github.io/illustrated-gpt2/

利用GPT-2 產生內容

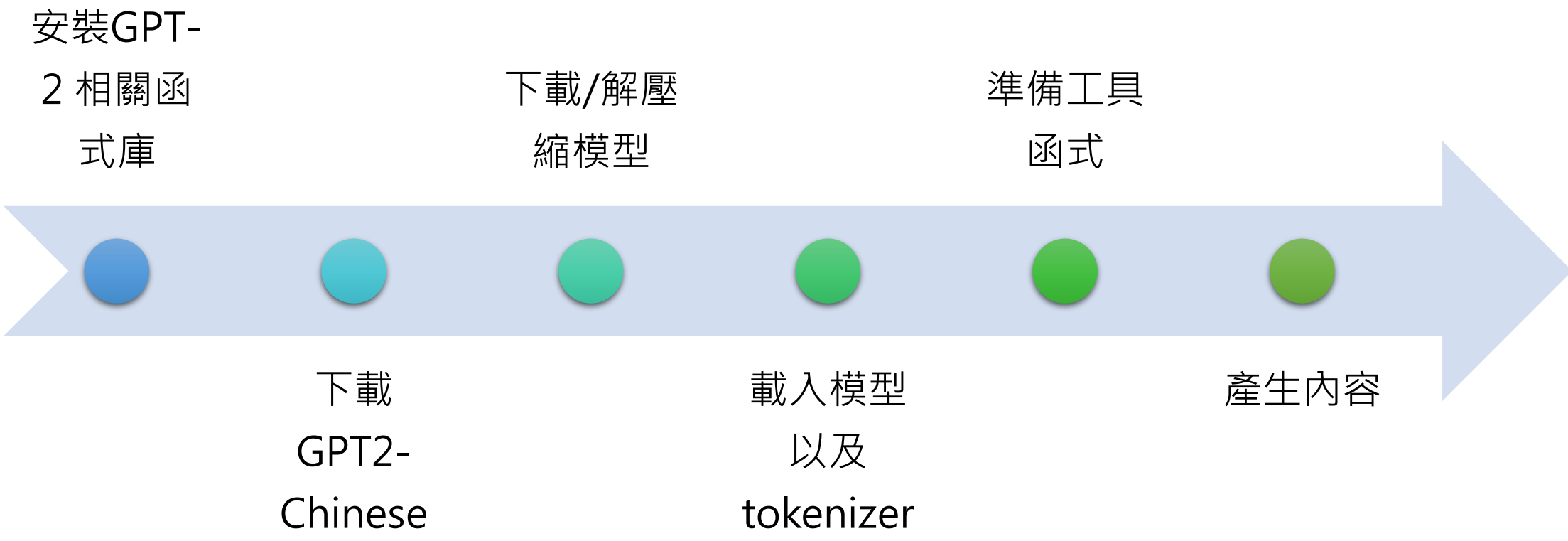
硬體規格

- 4 core CPU
- 8 G RAM
- 至少剩 20 G的磁碟空間
- GPU(不一定要有)

執行環境

- Jupyter
- Python3

利用GPT-2 產生內容


安裝GPT-2 相關函式庫
下載/解壓縮模型
準備工具函式
下載GPT2-Chinese
載入模型以及tokenizer
產生內容

利用GPT-2 產生內容

- 安裝GPT-2 相關函式庫
- 下載GPT2-Chinese (https://github.com/Morizeyao/GPT2-Chinese)
- 下載/解壓縮模型
- 載入模型以及 tokenizer
- 準備工具函式
- 產生內容

DEMO

參考資料

- Deep learning from scratch 2 (O’Reilly)
- Word2Vec Model
- Genism Word2Vec 簡易教學
- **GPT2-Chinese**
- **Activation Functions Explained - GELU, SELU, ELU, ReLU and more**

thank you
gracias
merci
danke
謝謝
ngiyabonga
شكراً جزيلاً
teşekkür ederim
рахмат
Баярлалаа
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
хвала
asante
manana
obrigada
tenki
murakoze
mersi
kia ora
barka
welalin
tack
spas
kiitos
dankie
dhanyavad
nanni
nandri
bedankt
bayarlalaa
gràcie
hvala
mauruuru
köszönöm
enkosi
dziękuję
akun
dankon
ačiū
mochchakkeram
djiere dieuf
tau
дякую
mamnun
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
sobodi
děkuji
obrigado
mèsi
didi madloba
sagolun
sukriya
kop khun krap
ありがとう
najis tuke
kam sah hamnida
rahmat
terima kasih
tanemirt
rahmet
grazie
tak
ευχαριστώ
taiku
arigatô
takk
dakujem
trugarez
diolch
dhanyavadagalu
shukriya
merce
мерси
xièxie
감사합니다

補充教材

Vagrant 介紹

Vagrant 簡介

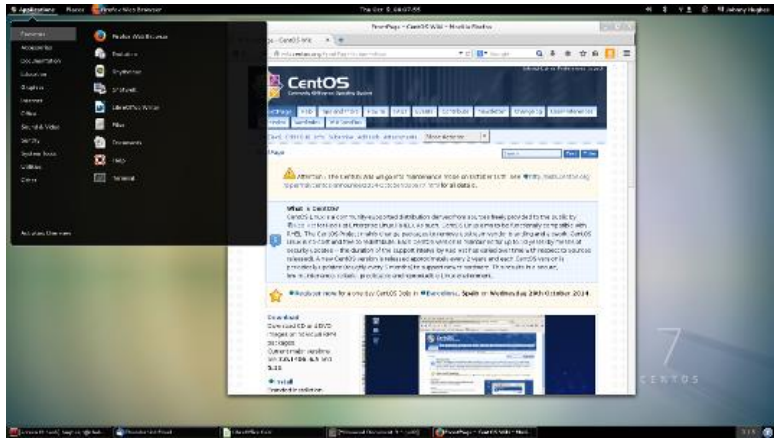
Vagrantfile + 

= Virtual machine

vagrant init centos/7

```
# -*- mode: ruby -*-
Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  config.vm.network "private_network", ip: "192.168.33.10"
  config.vm.synced_folder "./data", "/vagrant_data"
  config.vm.provider "virtualbox" do |vb|
    # Display the VirtualBox GUI when booting the machine
    vb.memory = "8096"
    vb.cpus = 4
  end
end
```

→ vagrant up →



Vagrant 安裝

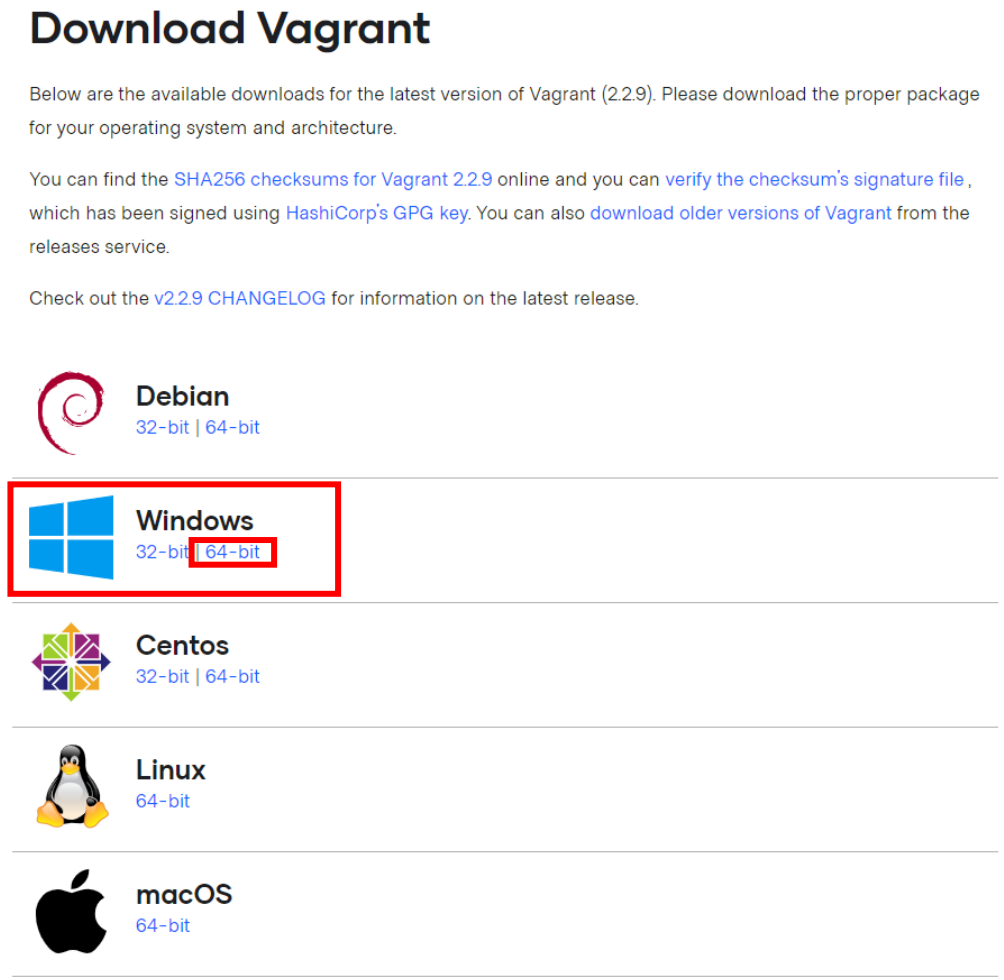
- 先安裝 VirtualBox
- 下載網址：

  https://www.virtualbox.org/wiki/Downloads

Vagrant 安裝

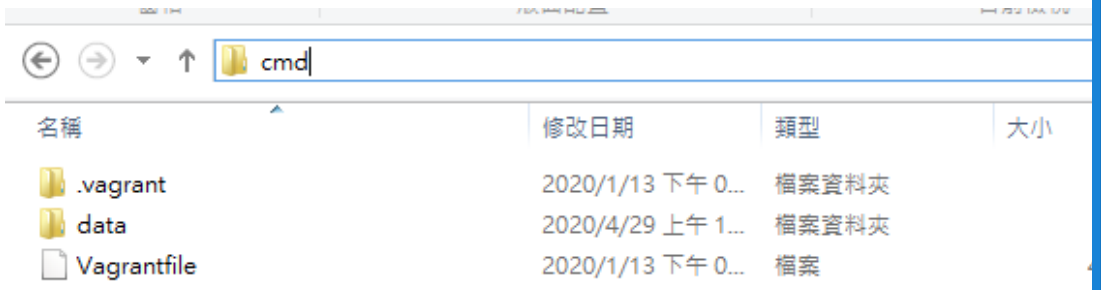
- 依照不同的環境，要安裝不同版本的Vagrant
- 下載網址：

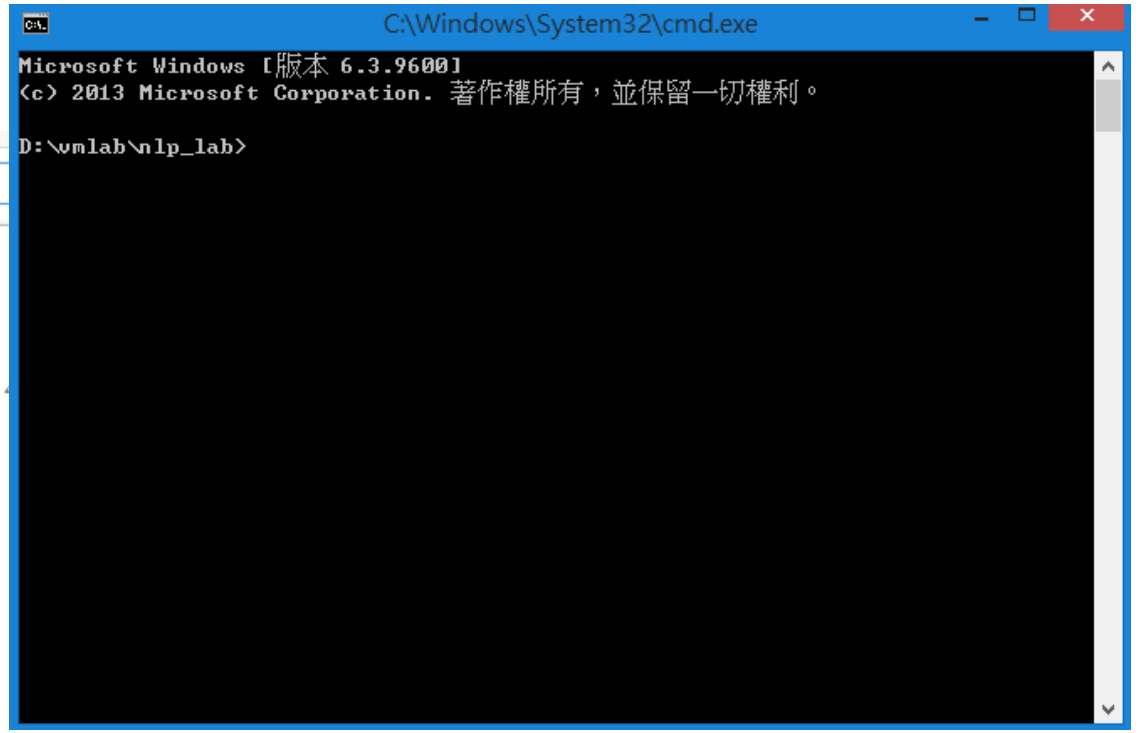
  https://www.vagrantup.com/downloads.html



利用 Vagrant 建立VM

- 開啟檔案總管
- 建立資料夾，名稱為 vmlab，然後在裡面建立一個nlp_lab資料夾
- 進入nlp_lab後，在檔案總館的導覽路徑上輸入cmd 後按下Enter，會跳出 windows 終端機



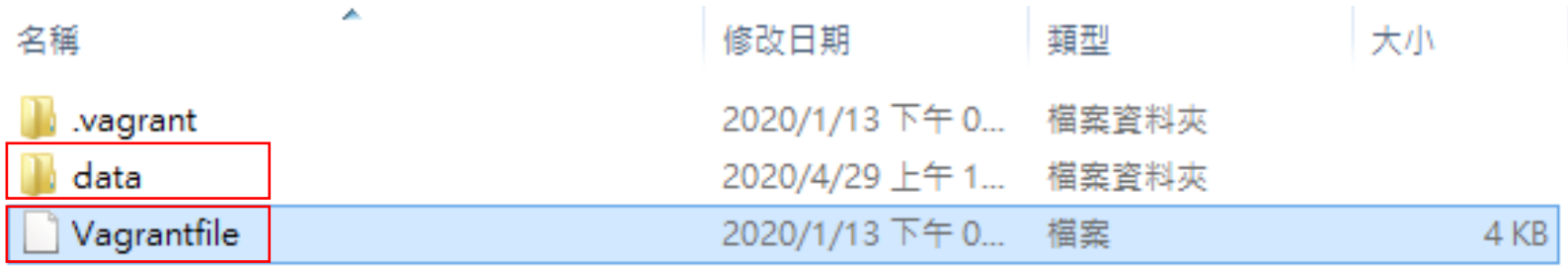


利用 Vagrant 建立VM

- 在終端機中輸入指令

```
vagrant init
```

- 按下enter 執行後，會出現 Vagrantfile 檔案
- 建立一個資料夾 data 作為與VM 資料交換用的資料夾



安裝 Vagrant plugin

- 安裝 vagrant-vbguest
- 在終端機中輸入指令

```
vagrant plugin install vagrant-vbguest
```

- automatically installs the host's VirtualBox Guest Additions on the guest system.

利用 Vagrant 建立VM

- 編輯 Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  config.vm.network "private_network", ip: "192.168.33.10"
  config.vm.synced_folder "./data", "/vagrant_data"

  config.vm.provider "virtualbox" do |vb|
    # vb.gui = true
    vb.memory = "8096"
    vb.cpus = 4
  end
end
```

利用 Vagrant 建立VM

- Vagrantfile編輯好之後，輸入指令

```
vagrant up
```

- 等候約10~20分鐘後，Centos 7 就建立完成

登入建立的 VM

- vagrant up 跑完之後，輸入指令

```
vagrant ssh
```

- 就登入建立好的VM，可以開始使用了

```
D:\vmlab\nlp_lab
λ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'centos/7' is up to date...
==> default: A newer version of the box 'centos/7' for provider 'virtualbox' is
==> default: available! You currently have version '1905.1'. The latest is version
==> default: '2004.01'. Run `vagrant box update` to update.
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
[default] GuestAdditions 5.2.18 running --- OK.
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Rsyncing folder: /cygdrive/d/vmlab/nlp_lab/ => /vagrant
==> default: Mounting shared folders...
    default: /vagrant_data => D:/vmlab/nlp_lab/data
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.

D:\vmlab\nlp_lab
λ vagrant status
Current machine states:

default                   running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.

D:\vmlab\nlp_lab
λ vagrant ssh
Last login: Wed Apr 29 03:31:42 2020 from 10.0.2.2
[vagrant@localhost ~]$
```

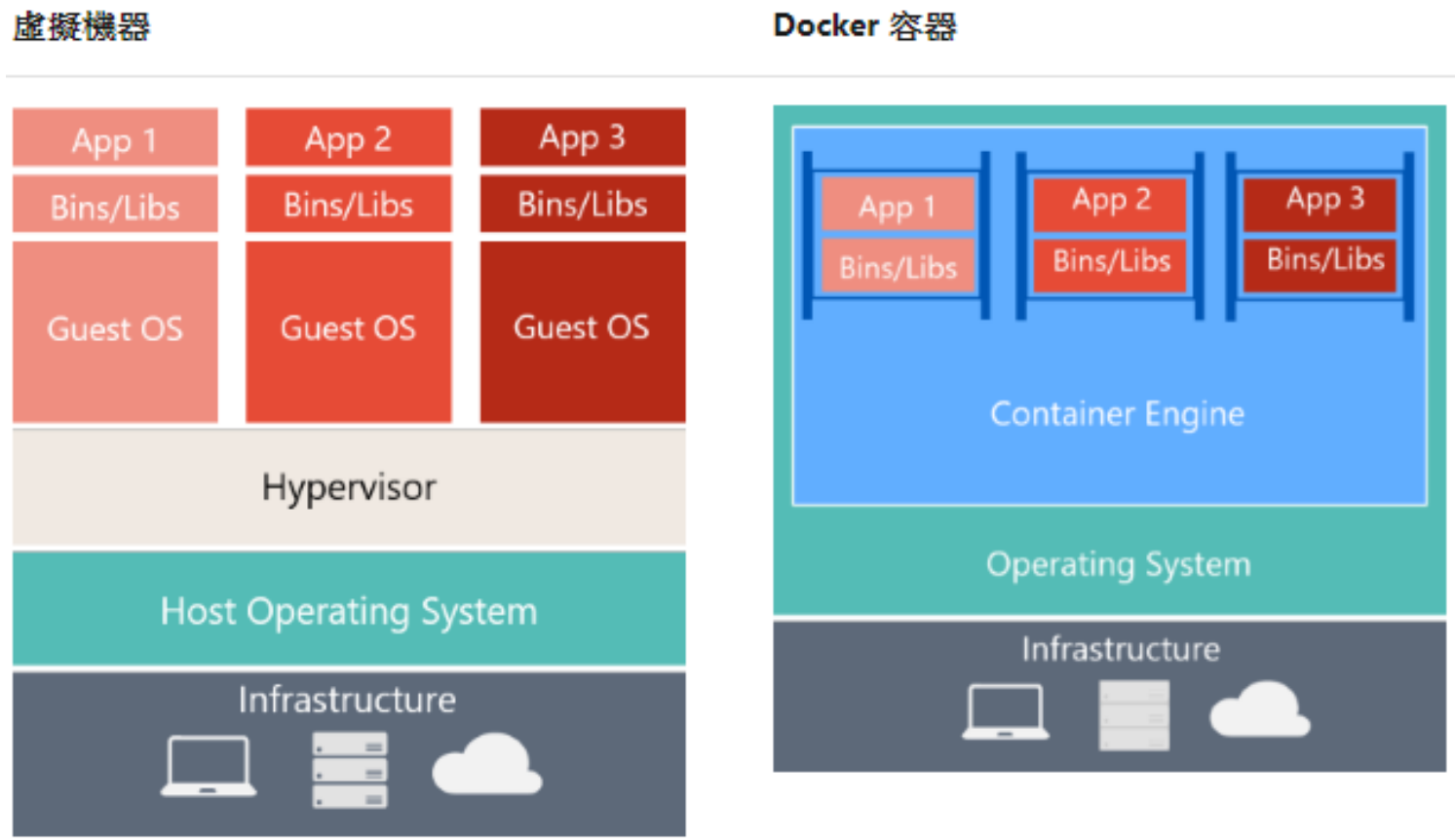
常用Vagrant 指令

- vagrant init
- vagrant up
- vagrant ssh
- vagrant status
- vagrant halt
- vagrant suspend
- vagrant destory

```
λ vagrant --help
Usage: vagrant [options] <command> [<args>]

    -v, --version                    Print the version and exit.
    -h, --help                       Print this help.

Common commands:
     box             manages boxes: installation, removal, etc.
     destroy         stops and deletes all traces of the vagrant machine
     global-status   outputs status Vagrant environments for this user
     halt            stops the vagrant machine
     help            shows the help for a subcommand
     init            initializes a new Vagrant environment by creating a Vagrantfile
     login           log in to HashiCorp's Vagrant Cloud
     package         packages a running vagrant environment into a box
     plugin          manages plugins: install, uninstall, update, etc.
     port            displays information about guest port mappings
     powershell      connects to machine via powershell remoting
     provision       provisions the vagrant machine
     push            deploys code in this environment to a configured destination
     rdp             connects to machine via RDP
     reload          restarts vagrant machine, loads new Vagrantfile configuration
     resume          resume a suspended vagrant machine
     snapshot        manages snapshots: saving, restoring, etc.
     ssh             connects to machine via SSH
     ssh-config      outputs OpenSSH valid configuration to connect to the machine
     status          outputs status of the vagrant machine
     suspend         suspends the machine
     up              starts and provisions the vagrant environment
     validate        validates the Vagrantfile
     vbguest         plugin: vagrant-vbguest: install VirtualBox Guest Additions to the machine
     version         prints current and latest Vagrant version
```

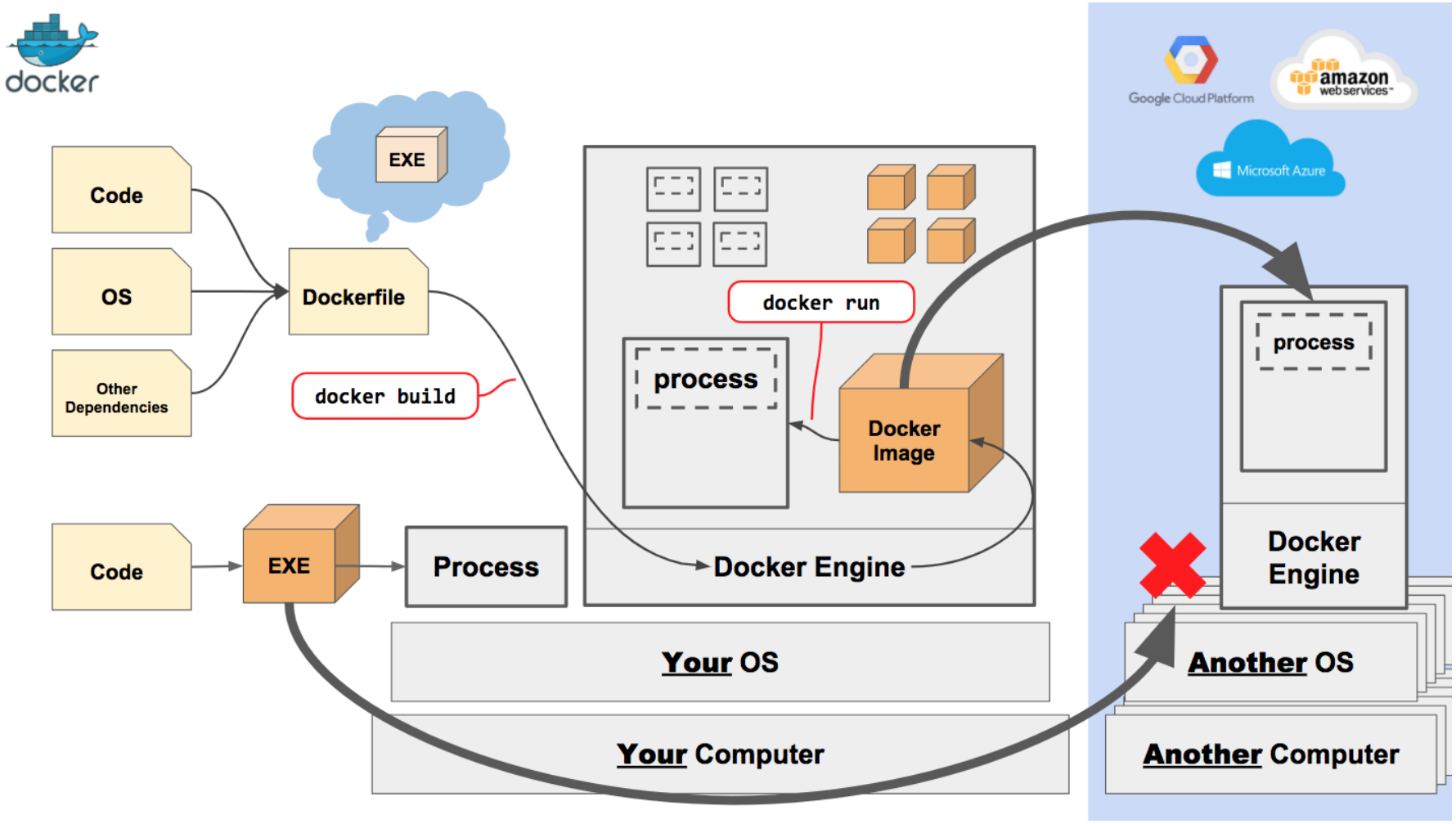
Docker 介紹

Docker 簡介



https://docs.microsoft.com/zh-tw/dotnet/architecture/microservices/container-docker-introduction/docker-defined

Docker 簡介



https://www.freecodecamp.org/news/docker-quick-start-video-tutorials-1dfc575522a0/

安裝 Docker (CentOS)

反安裝舊版 docker

```
sudo yum remove docker \
                docker-client \
                docker-client-latest \
                docker-common \
                docker-latest \
                docker-latest-logrotate \
                docker-logrotate \
                docker-engine
```

安裝 Docker (CentOS)

SET UP THE REPOSITORY

```
sudo yum install -y yum-utils
```

```
sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

安裝 Docker (CentOS)

安裝 docker

```
sudo yum install docker-ce docker-ce-cli containerd.io
```

啟動 docker

```
sudo systemctl start docker
sudo systemctl enable docker
```

測試 docker

```
sudo docker run hello-world
```

Docker compose 介紹

yum install -y yum-utils

安裝 Docker-compose

下載最新的穩定版本

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

調整檔案權限

```
sudo chmod +x /usr/local/bin/docker-compose
```

建立檔案連結到 /usr/bin

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

以建立WordPress 為例

利用下列docker 指令建立WordPress

```
$ docker run --name some-wordpress -p 8080:80 -d wordpress
```

WordPress 需要一個資料庫來存放資料

```
$ docker run --name some-wordpress -e WORDPRESS_DB_HOST=10.1.2.3:3306 \
    -e WORDPRESS_DB_USER=使用者名稱 -e WORDPRESS_DB_PASSWORD=密碼 -d wordpress
```

上述兩行docker 指令就可以建立一個WordPress 網站，可以開瀏覽器輸入網址 http://localhost:8080開啟

https://hub.docker.com/_/wordpress/

以建立WordPress 為例

建立檔案 docker-compose.yml

```
$ vi docker-compose.yml
```

利用下列docker-compose 指令建立WordPress

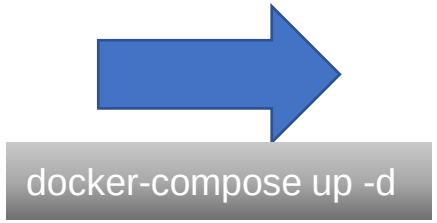
```
$ docker-compose up -d
```

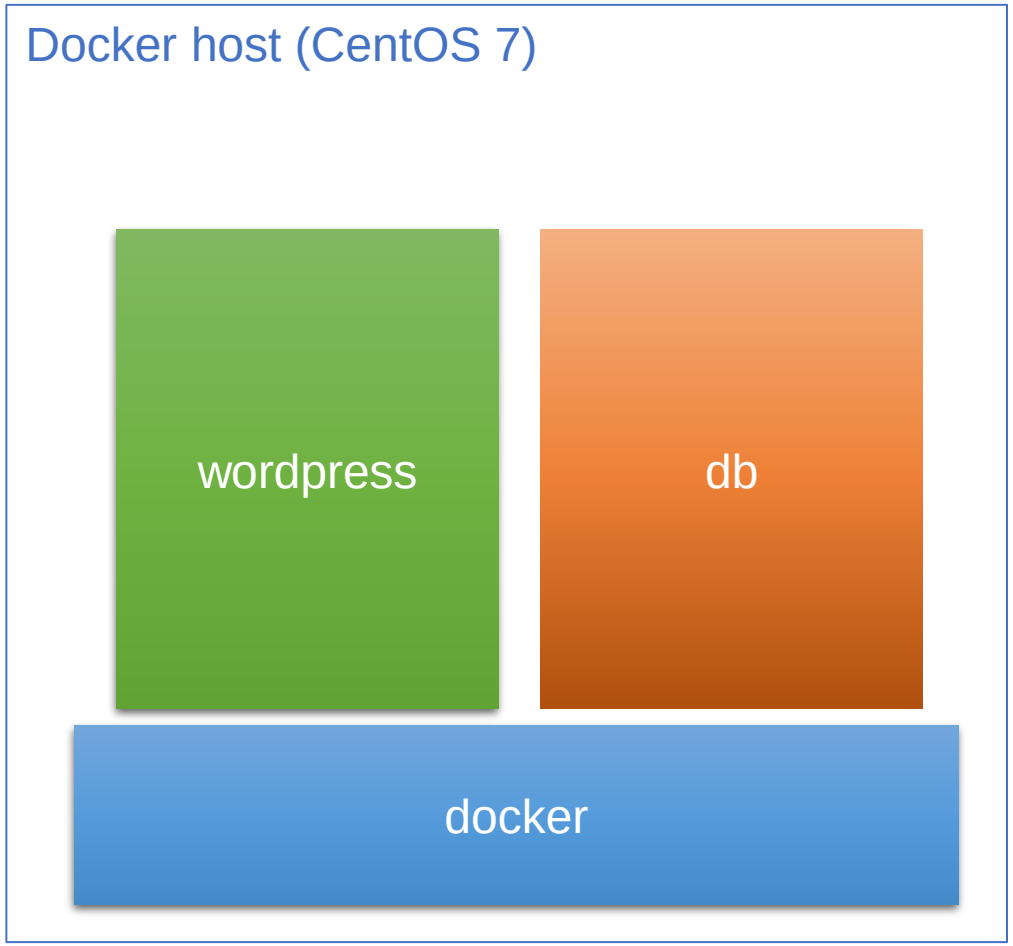
```
version: '3.1'

services:

  wordpress:
    image: wordpress
    restart: always
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: exampleuser
      WORDPRESS_DB_PASSWORD: examplepass
      WORDPRESS_DB_NAME: exampledb
    volumes:
      - wordpress:/var/www/html

  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: exampledb
      MYSQL_USER: exampleuser
      MYSQL_PASSWORD: examplepass
      MYSQL_RANDOM_ROOT_PASSWORD: '1'
    volumes:
      - db:/var/lib/mysql

volumes:
  wordpress:
  db:
```

https://hub.docker.com/_/wordpress/

以建立WordPress 為例

```yaml
version: '3.1'

services:

  wordpress:
    image: wordpress
    restart: always
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: exampleuser
      WORDPRESS_DB_PASSWORD: examplepass
      WORDPRESS_DB_NAME: exampledb
    volumes:
      - wordpress:/var/www/html

  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: exampledb
      MYSQL_USER: exampleuser
      MYSQL_PASSWORD: examplepass
      MYSQL_RANDOM_ROOT_PASSWORD: '1'
    volumes:
      - db:/var/lib/mysql

volumes:
  wordpress:
  db:
```





Jupyter 安裝與使用

何謂 Jupyter

- Jupyter 是從Ipython 衍生出的項目，是一種基於web 介面操作的計算環境
- 基本上Jupyter 與執行的語言沒有一定的關係，目前在上面常用的是Python、R、markdown 等
- Jupyter 可以很容易的在網頁上開發新專案，或作為學習研究之用，在 Jupyter 上執行的結果會保留在網頁上，方便檢視與討論。

取得上課程式碼

- 操作下列指令取得程式

```
1. mkdir -p /opt/nlp
2. cd /opt/nlp
3. git clone https://github.com/justin2061/2019_nlp.git
4. cd 2019_nlp
5. Chmod -R 777 work
```

安裝 Jupyter (docker)

docker-compose.yml

```yaml
version: "3.3"

services:
  notebook:
    image: jupyter/tensorflow-notebook
    container_name: jupyter
    hostname: jupyter
    restart: unless-stopped
    volumes:
      - ./work:/home/jovyan/work
      - ./jupyter_notebook_config.py:/root/.jupyter/jupyter_notebook_config.py
    ports:
      - 9999:8888
```

version: "3.3"

services:

notebook:

image: jupyter/tensorflow-notebook

container_name: jupyter

hostname: jupyter

restart: unless-stopped

volumes:

- ./work:/home/jovyan/work

- ./jupyter_notebook_config.py:/root/.jupyter/jupyter_notebook_config.py

ports:

- 9999:8888

安裝 Jupyter (docker)

- 利用docker-compose 指令啟動 jupyter

```
1. ls -la
2. cd /opt/nlp/2019_nlp
3. docker-compose up -d
```

```
root@localhost:/opt/nlp/work# docker-compose up -d
Creating network "nlp_default" with the default driver
Pulling notebook (jupyter/tensorflow-notebook:)...
latest: Pulling from jupyter/tensorflow-notebook
5b7339215d1d: Pull complete
14ca88e9f672: Pull complete
a31c3b1caad4: Pull complete
b054a26005b7: Pull complete
51cd026b34bd: Pull complete
c583df1459c5: Pull complete
4faa8cc1412c: Pull complete
fb492281364f: Pull complete
dd8998f7ce9b: Pull complete
c7e0aa6a3d57: Pull complete
74394ad8292e: Pull complete
3178e9445266: Pull complete
ef048d47cab0: Pull complete
5319e76389f6: Pull complete
ffe75d5f8603: Pull complete
16d0e5bb0db6: Pull complete
8fb913e38cfe: Pull complete
1f37c1a7d550: Pull complete
9f850a978466: Pull complete
9d7f36ed6376: Pull complete
1c89cff84cd0: Pull complete
647a0e768c0a: Pull complete
10a14b8bd75f: Pull complete
35887245c10c: Pull complete
Creating jupyter ... done
root@localhost:/opt/nlp/work# docker-compose ps
 Name               Command              State           Ports
------------------------------------------------------------------------
jupyter   tini -g -- start-notebook.sh   Up      0.0.0.0:9999->8888/tcp
root@localhost:/opt/nlp/work#
```

啟動 Jupyter (docker)

- 利用以下指令啟動觀察啟動 token：

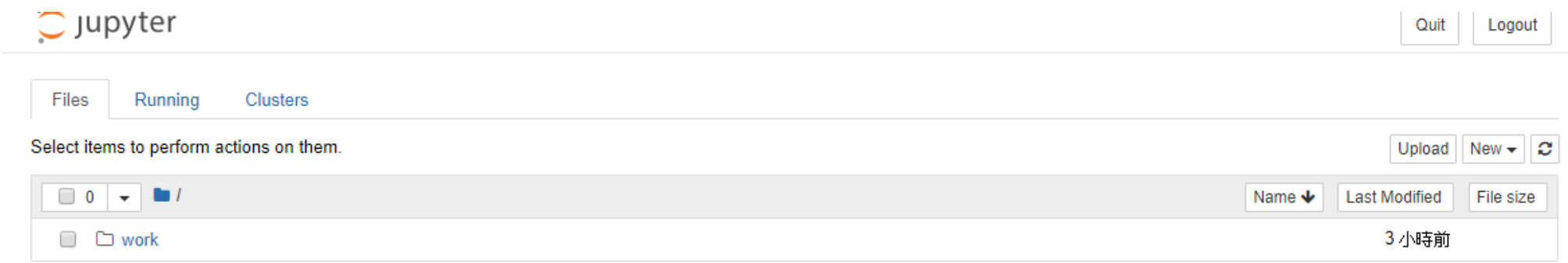
1. docker-compose logs -f

```
jupyter      | Executing the command: jupyter notebook
jupyter      | [I 09:04:48.133 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
jupyter      | [I 09:04:50.307 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.7/site-packages/jupyterlab
jupyter      | [I 09:04:50.308 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
jupyter      | [I 09:04:51.991 NotebookApp] Serving notebooks from local directory: /home/jovyan
jupyter      | [I 09:04:51.991 NotebookApp] The Jupyter Notebook is running at:
jupyter      | [I 09:04:51.991 NotebookApp] http://jupyter:8888/?token=a7fbe0fee8192cd860e796a219f496b0ff68e375b7da7d3e
jupyter      | [I 09:04:51.991 NotebookApp]  or http://127.0.0.1:8888/?token=a7fbe0fee8192cd860e796a219f496b0ff68e375b7da7d3e
jupyter      | [I 09:04:51.991 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
jupyter      | [C 09:04:51.996 NotebookApp]
jupyter      |
jupyter      |     To access the notebook, open this file in a browser:
jupyter      |         file:///home/jovyan/.local/share/jupyter/runtime/nbserver-8-open.html
jupyter      |     Or copy and paste one of these URLs:
jupyter      |         http://jupyter:8888/?token=a7fbe0fee8192cd860e796a219f496b0ff68e375b7da7d3e
jupyter      |      or http://127.0.0.1:8888/?token=a7fbe0fee8192cd860e796a219f496b0ff68e375b7da7d3e
jupyter      | [W 09:12:59.251 NotebookApp] Clearing invalid/expired login cookie username-139-162-75-164-9999
jupyter      | [W 09:12:59.252 NotebookApp] Forbidden
```

- 依上圖，觀察到jupyter 的網址為：http://jupyter:8888/?token=a7fbe0fee8192cd860e796a219f496b0ff68e375b7da7d3e
- 網址中的 “jupyter” 請調整為自己 VM 的 IP，8888改為9999 ，例如 http://139.162.75.164:9999/?token=a7fbe0fee8192cd860e796a219f496b0ff68e375b7da7d3e
- 用瀏覽器開啟 jupyter 畫面

啟動 Jupyter

- Jupyter 為 http://自己VM的IP:9999/?token=觀察到的token
- 用瀏覽器開啟 jupyter 畫面



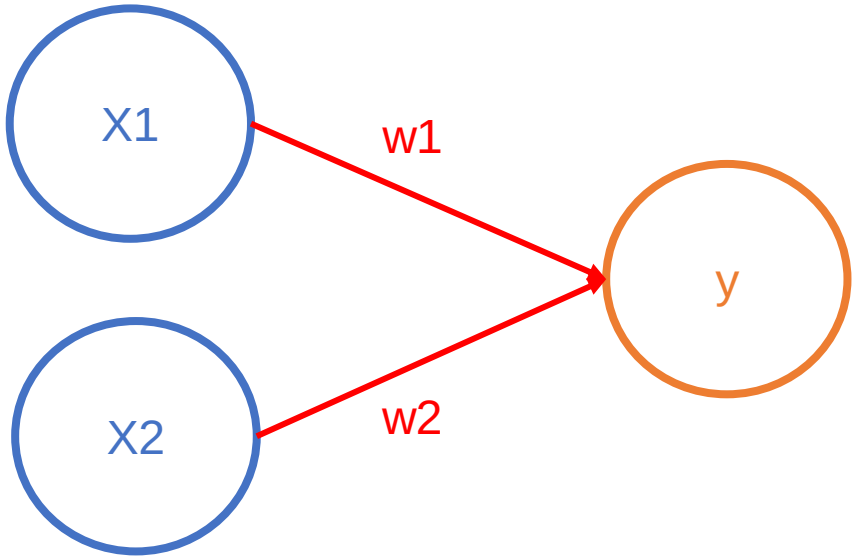
類神經網路簡介

概要

- 感知器
- 邏輯閘
- 神經網路

感知器(Perceptron)

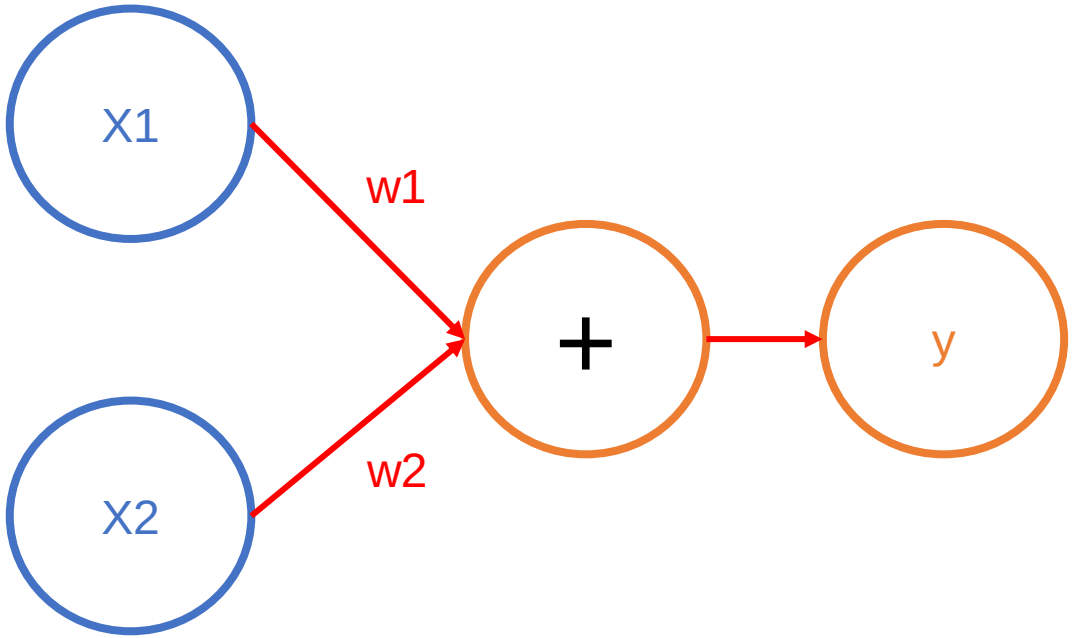
$$y = w_1 x_1 + w_2 x_2$$



參考連結

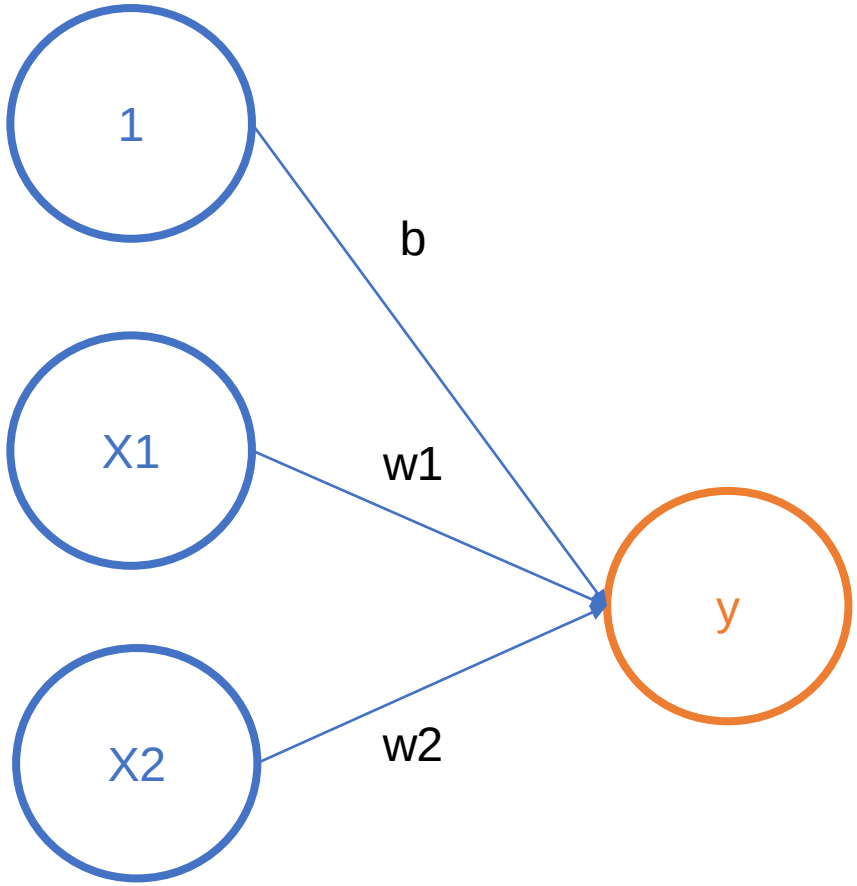
計算圖(Computational graph)

$$y = w_1 x_1 + w_2 x_2$$



感知器(Perceptron)

$$y = b + w_1 x_1 + w_2 x_2$$



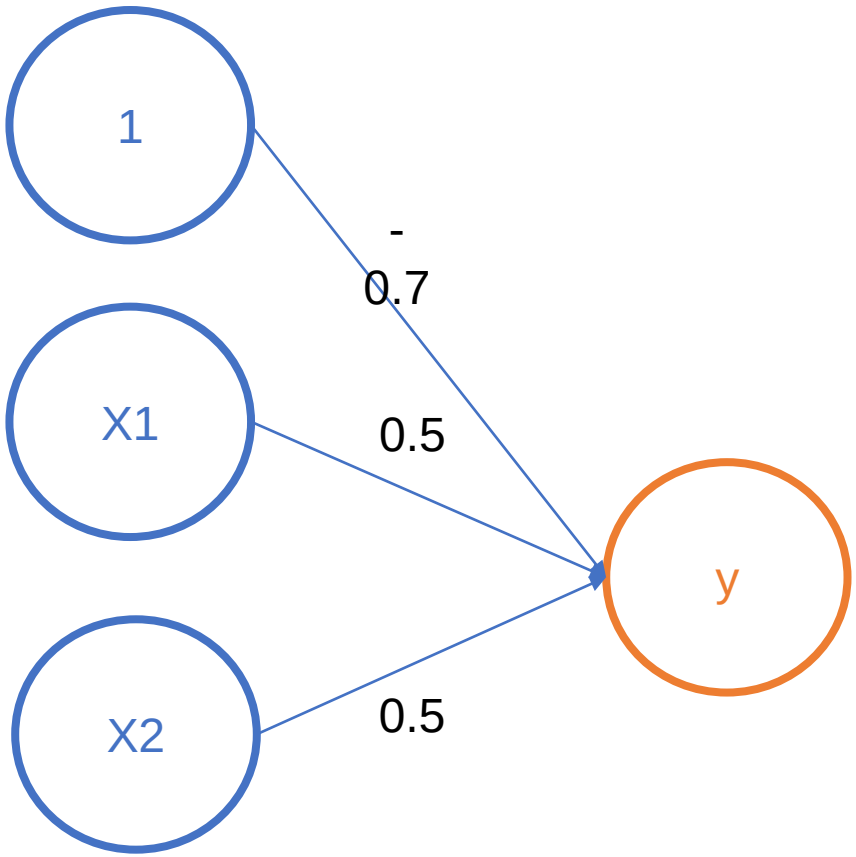
參考連結

邏輯閘

圖片來源：https://www.reddit.com/r/ProgrammerHumor/comments/6qlpwd/the_whole_droid_family/

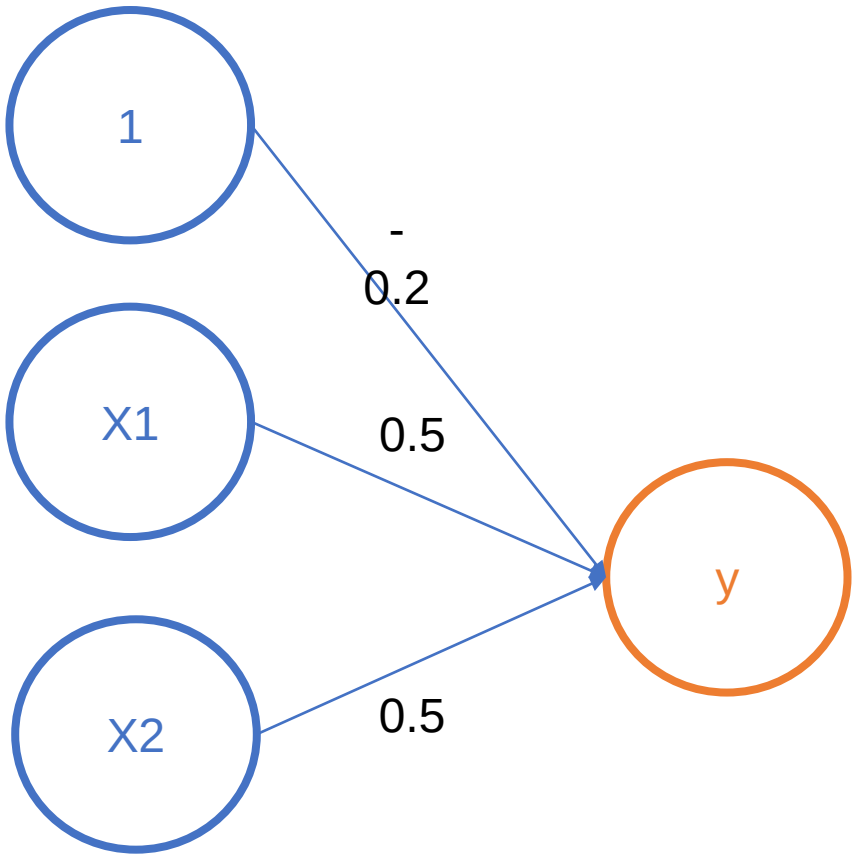
AND gate

X1	X2	Y
0	0	0
1	0	0
0	1	0
1	1	1



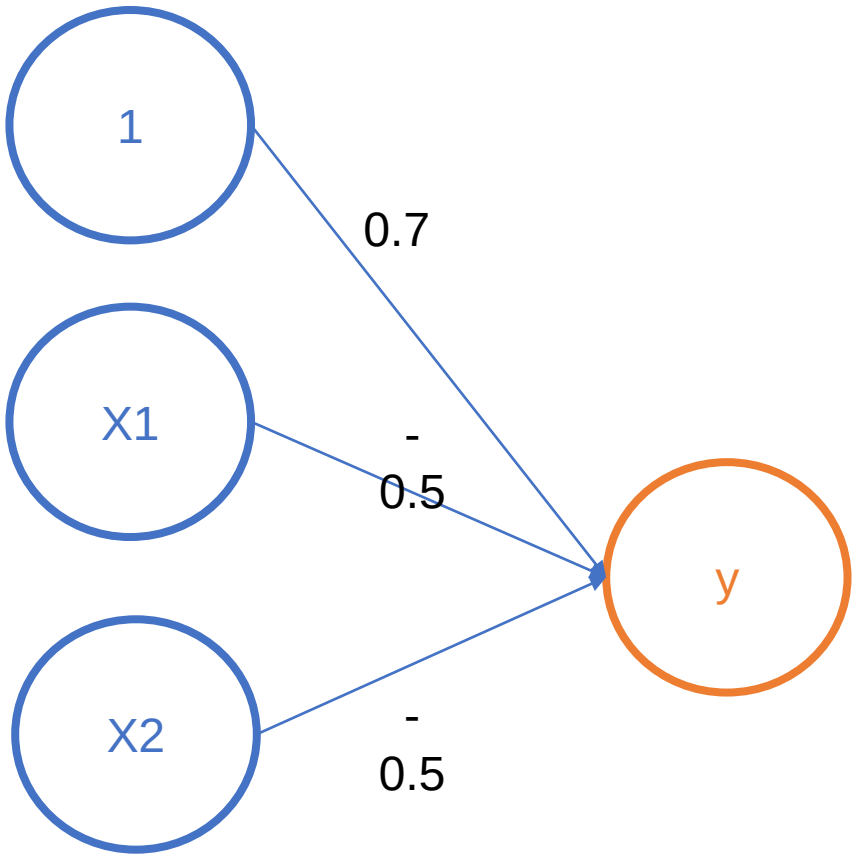
OR gate

X1	X2	Y
0	0	0
1	0	1
0	1	1
1	1	1



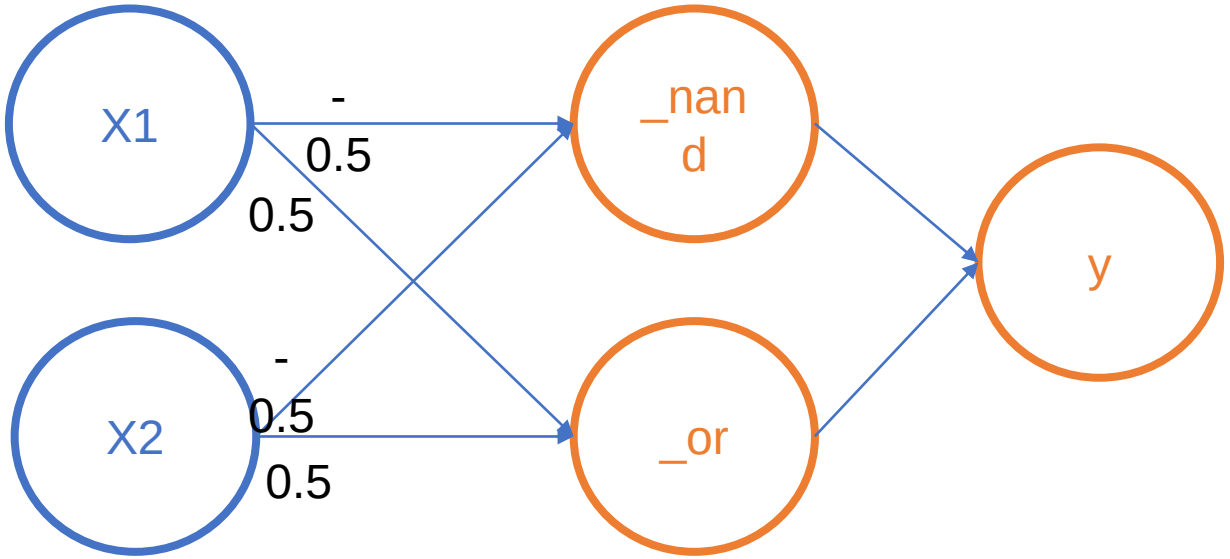
NAND gate

X1	X2	Y
0	0	1
1	0	1
0	1	1
1	1	0



XOR gate

X1	X2	Y
0	0	0
1	0	1
0	1	1
1	1	0

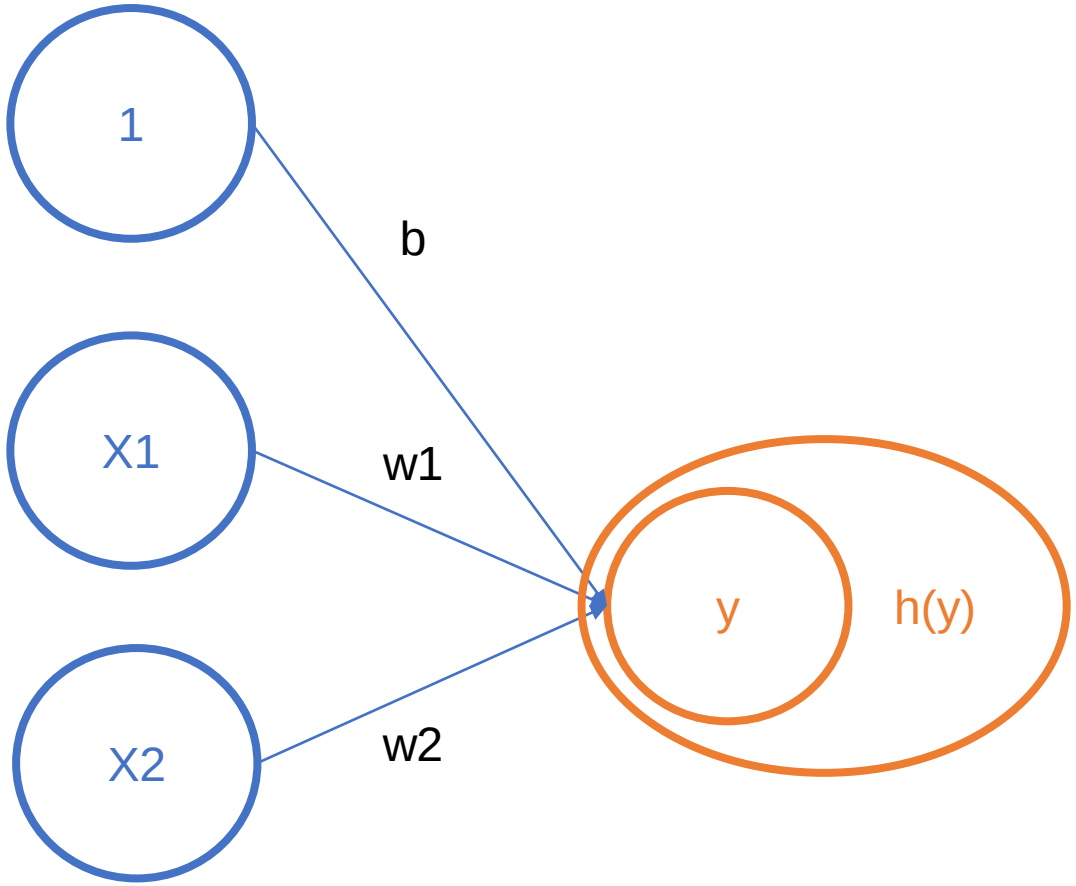


神經網路

$$y = b + w_1 x_1 + w_2 x_2$$

活化函數

$$z = h(y)$$

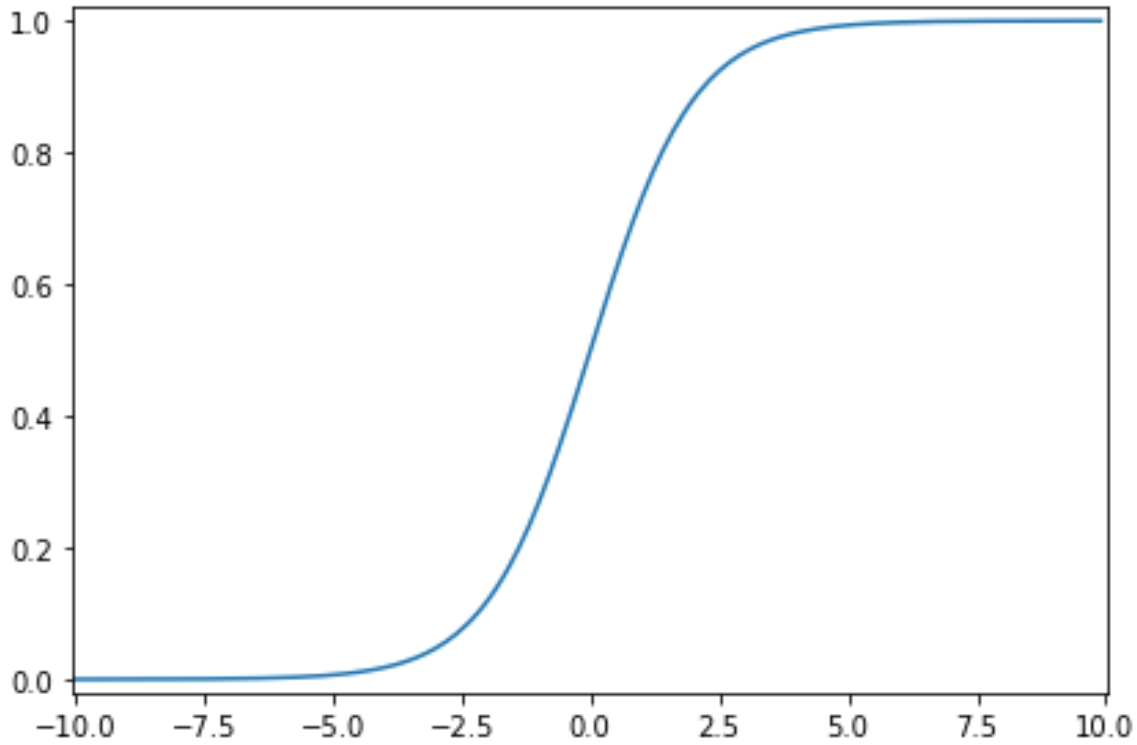


活化函數

Sigmoid函數

$$h(x) = \frac{1}{1 + \exp(-x)}$$

```
def sigmoid(x):
    s = 1 / (1 + np.exp(-x))
    return s
```

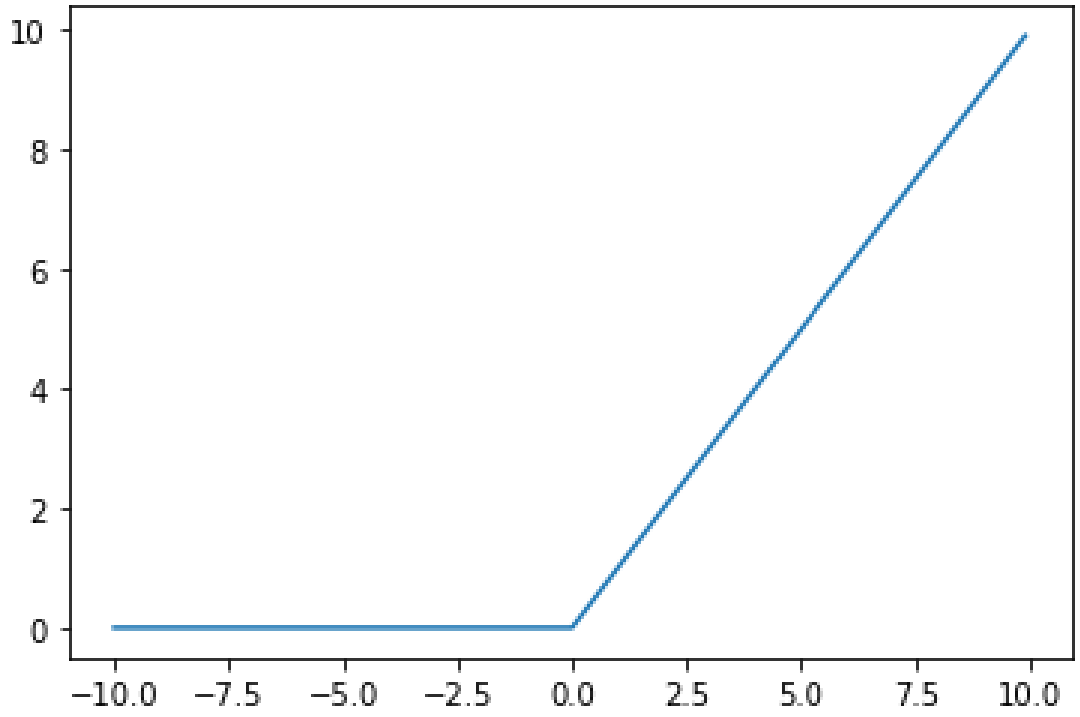


DEMO

活化函數

ReLU函數

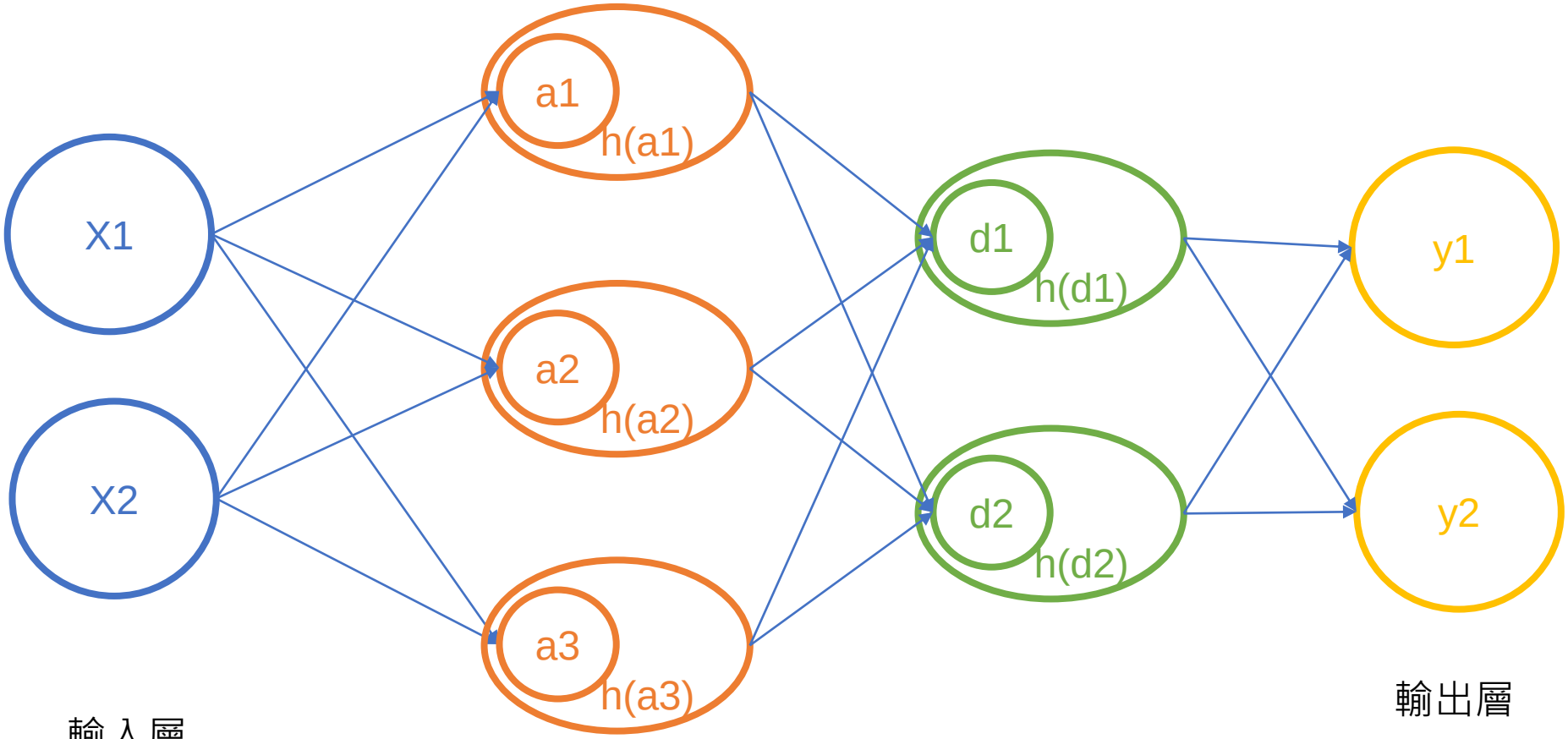
$$h(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$




```
def relu(x):
    return np.maximum(0,x)
```

DEMO

神經網路



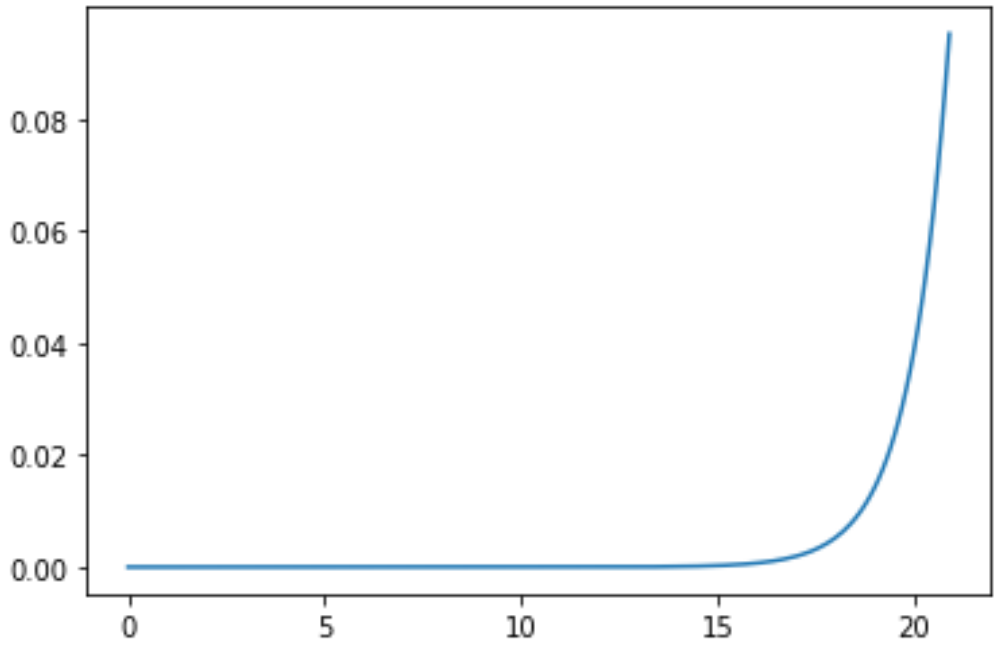
輸出層

Softmax函數

$$y_k = \frac{\exp(a_k)}{\sum_{1^1=1}^{n} \exp(a_i)}$$

```
def softmax(x):
    c = np.max(x)
    exp_ac = np.exp(x-c)
    y = exp_ac / np.sum(exp_ac)
    return y
```

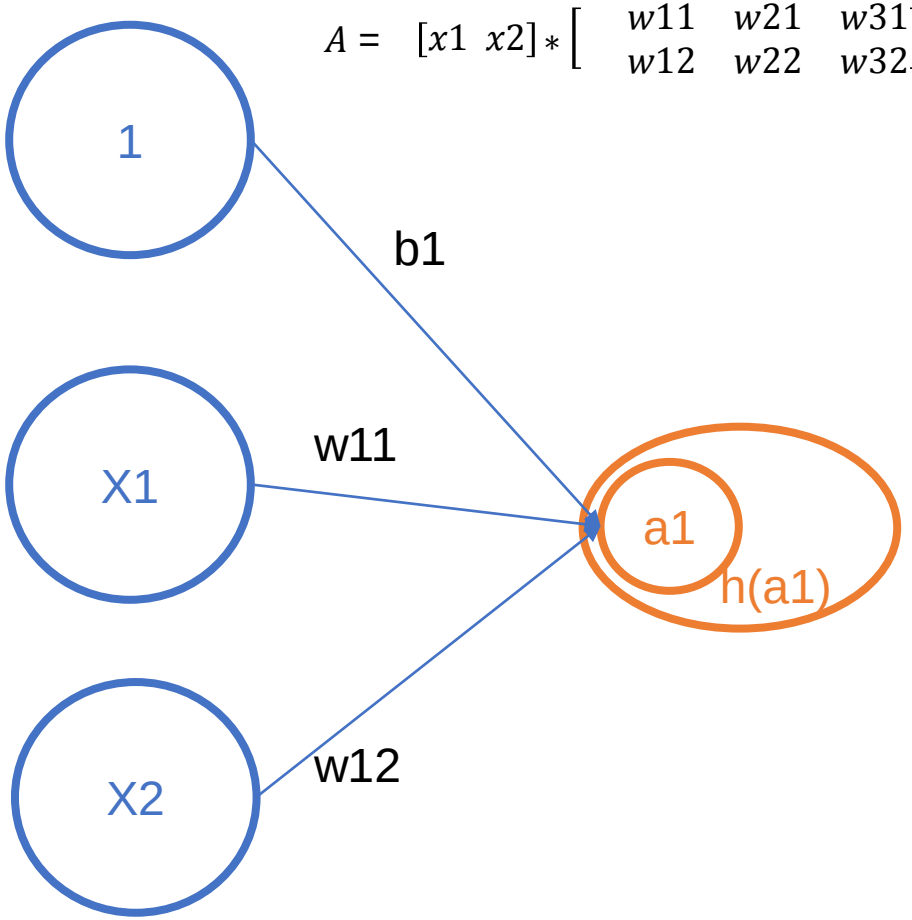
DEMO



神經網路的訊息傳遞

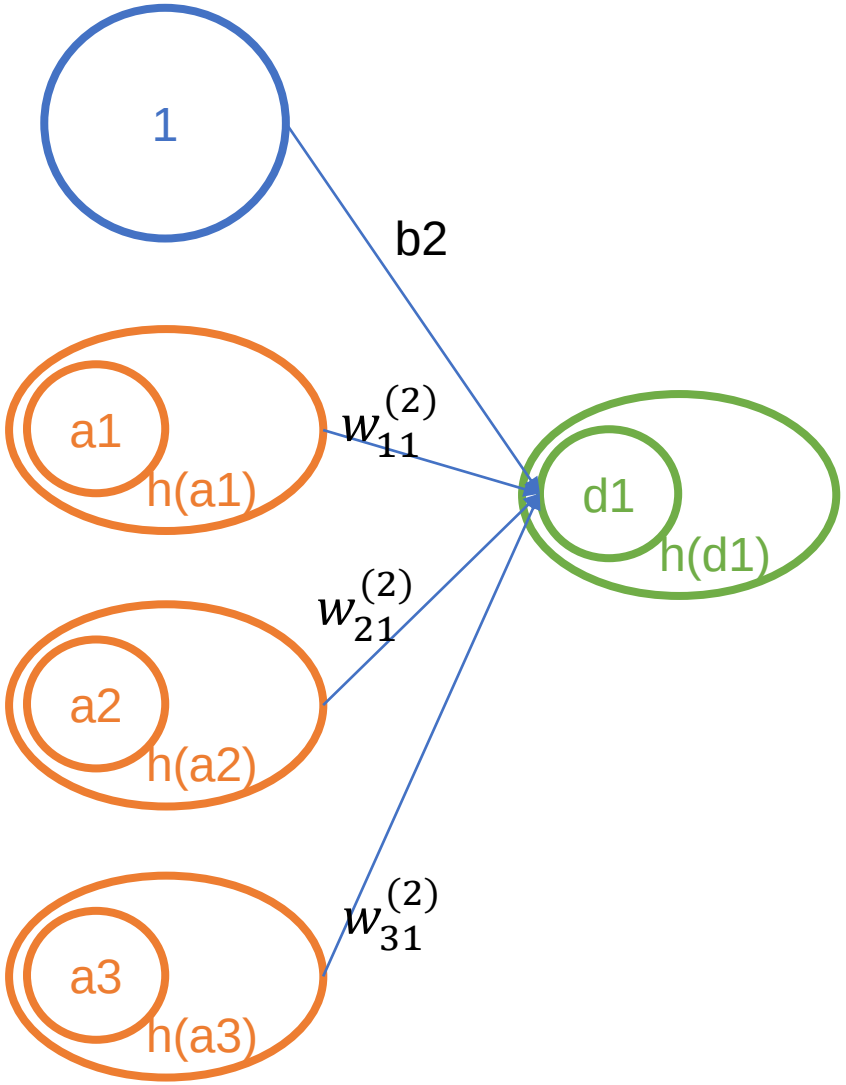
$$A = XW + B$$

$$A = [x1\ \ x2] * \begin{bmatrix} w11 & w21 & w31 \\ w12 & w22 & w32 \end{bmatrix} + [b1\ \ b2\ b3]$$




```
In [1]: import numpy as np

In [16]: def sigmoid(x):
             s = 1 / (1 + np.exp(-x))
             return s

In [15]: X = np.array([0.9, 0.5])
         X
Out[15]: array([0.9, 0.5])

In [13]: W1 = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
         W1
Out[13]: array([[0.1, 0.3, 0.5],
                [0.2, 0.4, 0.6]])

In [14]: B1 = np.array([0.3, 0.4, 0.7])
         B1
Out[14]: array([0.3, 0.4, 0.7])

In [7]: X.shape
Out[7]: (2,)

In [10]: W1.shape
Out[10]: (2, 3)

In [9]: B1.shape
Out[9]: (3,)

In [11]: A1 = np.dot(X, W1) + B1

In [12]: A1
Out[12]: array([0.49, 0.87, 1.45])

In [17]: Z1 = sigmoid(A1)

In [18]: Z1
Out[18]: array([0.62010643, 0.7047457 , 0.80999843])
```

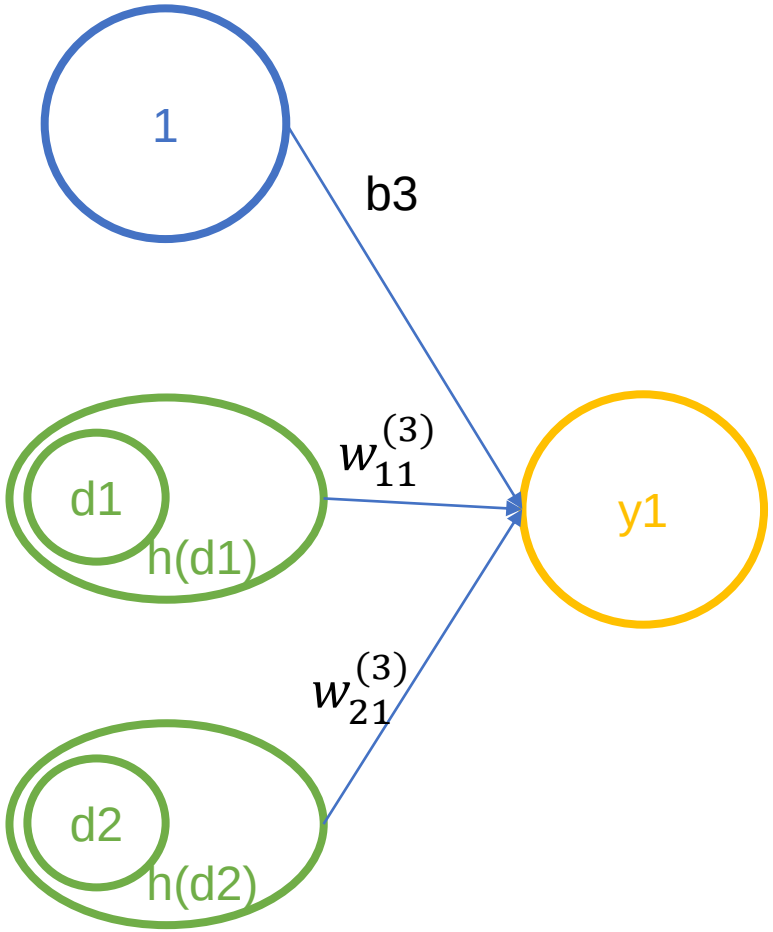
神經網路的訊息傳遞



第一層到第二層

```
In [19]: W2 = np.array([[0.1, 0.3],[0.5,0.7],[0.9,0.4]])
         W2

Out[19]: array([[0.1, 0.3],
                [0.5, 0.7],
                [0.9, 0.4]])

In [20]: B2 = np.array([0.2, 0.3])
         B2

Out[20]: array([0.2, 0.3])

In [21]: Z1.shape

Out[21]: (3,)

In [22]: W2.shape

Out[22]: (3, 2)

In [23]: B2.shape

Out[23]: (2,)

In [25]: A2 = np.dot(Z1, W2)+B2
         A2

Out[25]: array([1.34338208, 1.30335329])

In [26]: Z2 = sigmoid(A2)
         Z2

Out[26]: array([0.79304557, 0.7863988 ])
```

神經網路的訊息傳遞



第二層到輸出層

In [27]:
```
def softmax(x):
    c = np.max(x)
    exp_ac = np.exp(x-c)
    y = exp_ac / np.sum(exp_ac)
    return y
```

In [34]:
```
Z2.shape
```

Out[34]: (2,)

In [31]:
```
W3 = np.array([[0.5, 0.7],[0.4,0.9]])
W3.shape
```

Out[31]: (2, 2)

In [30]:
```
B3 = np.array([0.5, 0.8])
B3.shape
```

Out[30]: (2,)

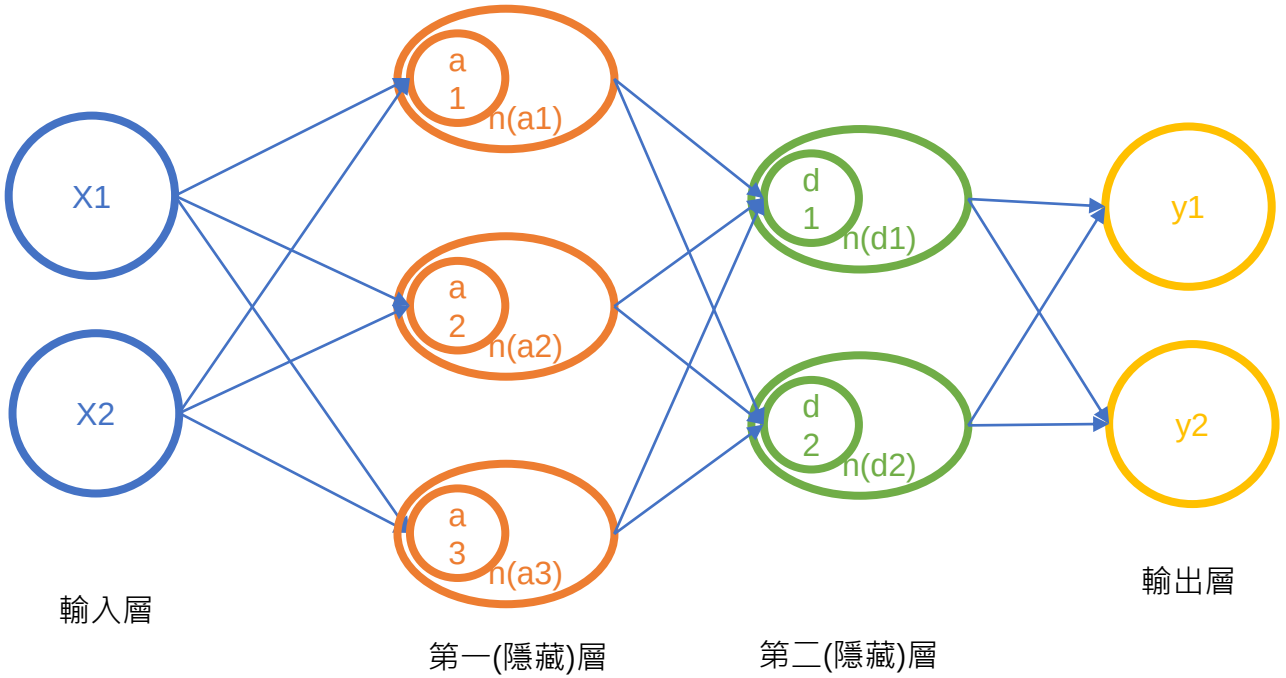
In [36]:
```
A3 = np.dot(Z2, W3) + B3
A3
```

Out[36]: array([1.21108231, 2.06289082])

In [37]:
```
Z3 = softmax(A3)
Z3
```

Out[37]: array([0.29905362, 0.70094638])

神經網路的訊息傳遞




```
In [41]: def init_network():
             network = {}
             network['W1'] = W1
             network['b1'] = B1
             network['W2'] = W2
             network['b2'] = B2
             network['W3'] = W3
             network['b3'] = B3
             return network
```

```
In [45]: def forward(network, x):
             W1,W2,W3 = network['W1'], network['W2'], network['W3']
             b1,b2,b3 = network['b1'], network['b2'], network['b3']

             a1 = np.dot(x, W1) + b1
             z1 = sigmoid(a1)

             a2 = np.dot(z1, W2) + b2
             z2 = sigmoid(a2)

             a3 = np.dot(z2, W3) + b3
             z3 = softmax(a3)
             return z3
```

```
In [46]: network = init_network()
         network
```

```
Out[46]: {'W1': array([[0.1, 0.3, 0.5],
                       [0.2, 0.4, 0.6]]),
          'b1': array([0.3, 0.4, 0.7]),
          'W2': array([[0.1, 0.3],
                       [0.5, 0.7],
                       [0.9, 0.4]]),
          'b2': array([0.2, 0.3]),
          'W3': array([[0.5, 0.7],
                       [0.4, 0.9]]),
          'b3': array([0.5, 0.8])}
```

```
In [47]: x = np.array([0.9, 0.5])
         y = forward(network, x)
         y
```

```
Out[47]: array([0.29905362, 0.70094638])
```

淺談活化函數

概要

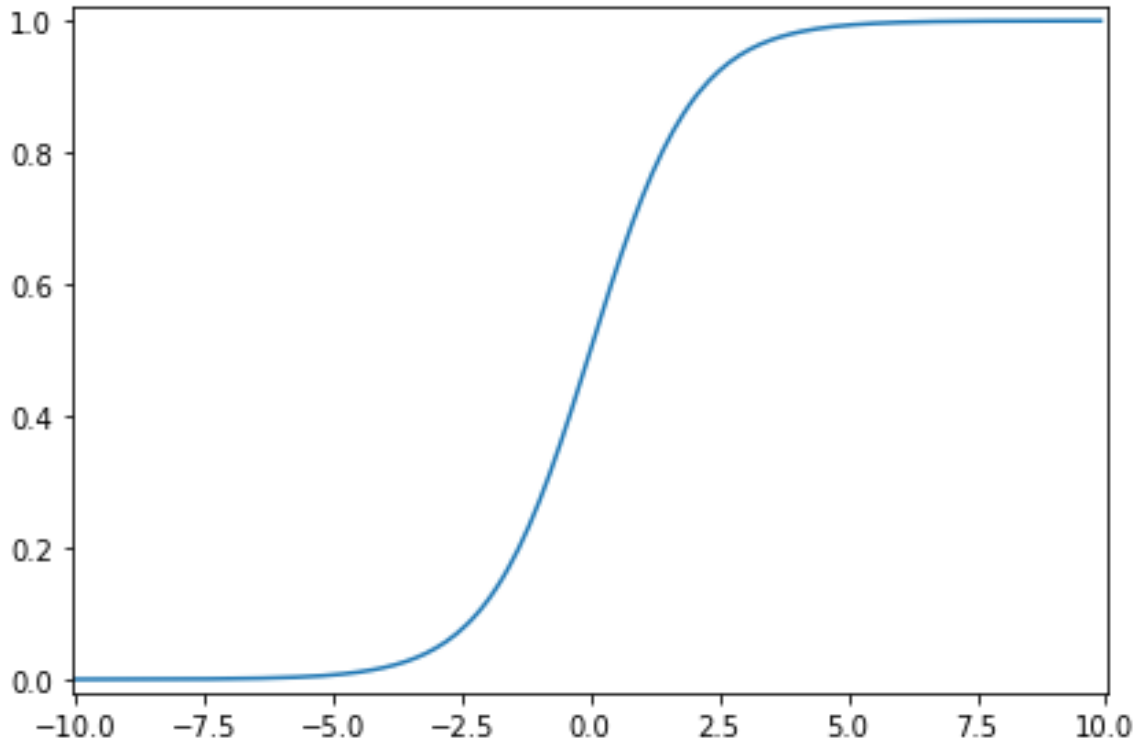
- Sigmoid
- ReLU
- Softmax
- tanh

活化函數

Sigmoid函數

$$h(x) = \frac{1}{1 + \exp(-x)}$$

```
def sigmoid(x):
    s = 1 / (1 + np.exp(-x))
    return s
```

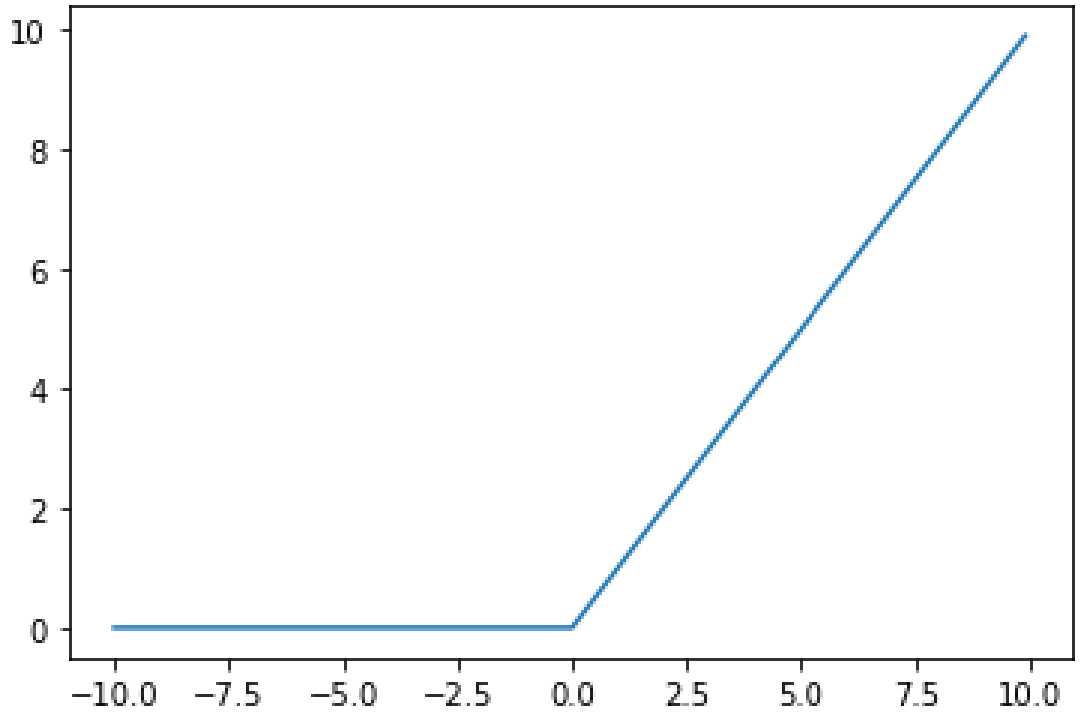


DEMO

活化函數

ReLU函數

$$h(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$



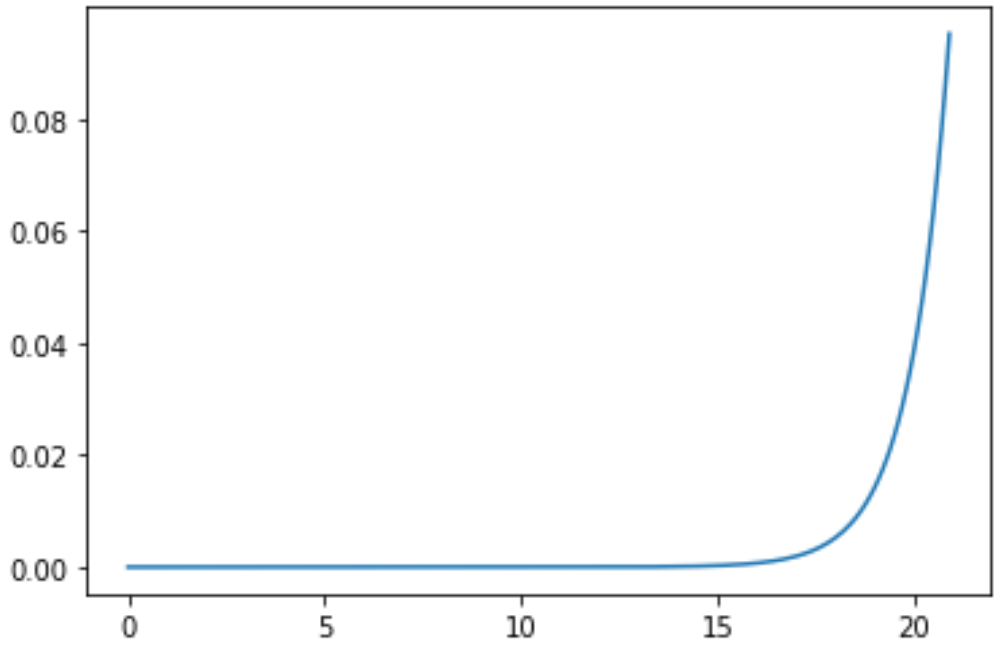

```
def relu(x):
    return np.maximum(0,x)
```

DEMO

活化函數

Softmax函數

$$y_k = \frac{\exp(a_k)}{\sum_{1^1=1}^{n} \exp(a_i)}$$




```
def softmax(x):
    c = np.max(x)
    exp_ac = np.exp(x-c)
    y = exp_ac / np.sum(exp_ac)
    return y
```

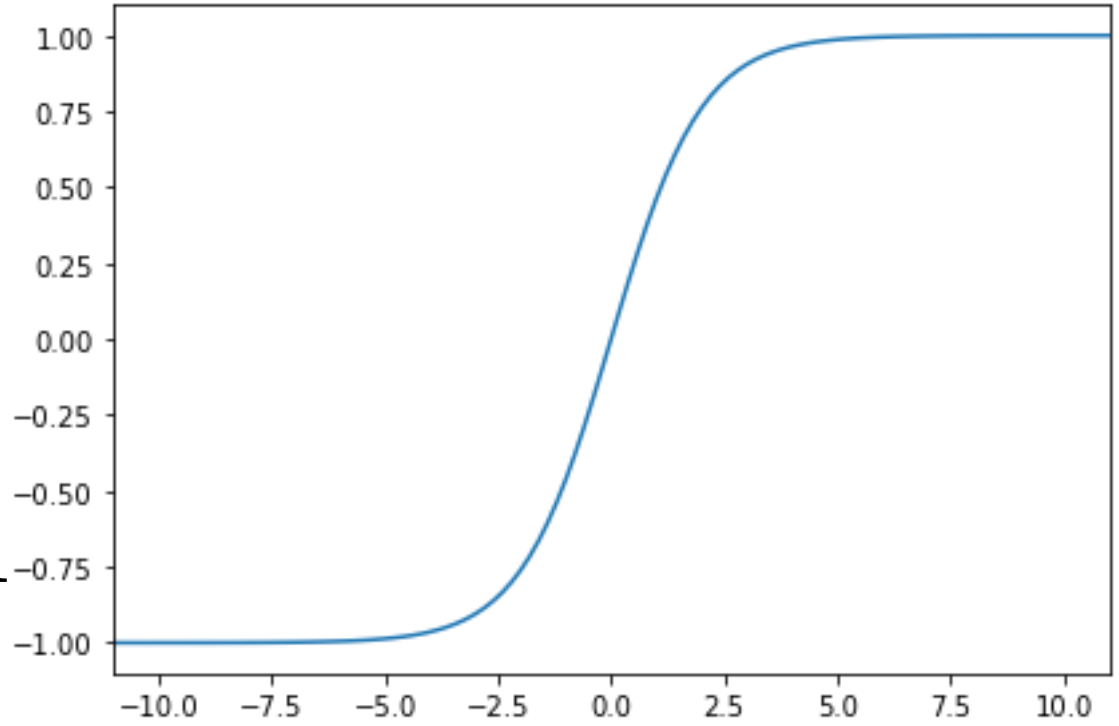
DEMO

活化函數

tanh函數

$$\tanh x = \frac{\sinh x}{\cosh x}$$

$$\tanh x = \frac{2}{1 - e^{-2x}} - 1$$




```
def tanh(x):
    return 2*sigmoid(2*x) - 1
```

$$\tanh x = 2\ sigmoid(2x) - 1$$

DEMO

Jieba on Python

https://github.com/fxsjy/jieba

概要

- 特點
- 安裝
- 基本分詞

Jieba 特點

- 三種分詞模式：
  - 精确模式：試圖將句子最精確切開，適合文本分析；
  - 全模式：把句子中所有的可以成詞的詞語都掃描出來, 速度非常快，但是不能解決歧異
  - 搜索引擎模式：在精確模式的基礎上，對長詞再次切分，提高召回率，適合用於搜索引擎分詞。
- 繁體分詞
- 自定義辭典

Jieba 安裝

- 全自動安裝：easy_install jieba 或者 pip install jieba / pip3 install jieba
- 半自動安裝：先下載 http://pypi.python.org/pypi/jieba/ ，解壓後執行 python setup.py install
- 手動安裝：將 jieba 目錄放置於目前的目錄或者 site-packages 目錄
- 通過 import jieba 來引用

Jieba 基本分詞

- jieba.**cut** 方法接受三個輸入參數: 需要分詞的字串；cut_all 參數用來控制是否採用全模式；HMM 參數用來控制是否使用 HMM 模型
- jieba.cut_**for_search** 方法接受兩個參數：需要分詞的字串；是否使用 HMM 模型。該方法適合用於搜尋引擎構建倒排索引的分詞，細微性比較細
- 待分詞的字串可以是 unicode 或 UTF-8 字串、GBK 字串。注意：不建議直接輸入 GBK 字串，可能無法預料地錯誤解碼成 UTF-8
- jieba.cut 以及 jieba.cut_for_search 返回的結構都是一個可反覆運算的 **generator**，可以使用 for 迴圈來獲得分詞後得到的每一個詞語(unicode)，或者用
- jieba.**lcut** 以及 jieba.**lcut_for_search** 直接返回 **list**
- jieba.Tokenizer(dictionary=DEFAULT_DICT) 新建自訂分詞器，可用於同時使用不同詞典。jieba.dt 為默認分詞器，所有全域分詞相關函數都是該分詞器的映射。

Jieba 基本分詞

- 全模式

```
1. seg_list = jieba.cut("西元前三世紀的古希臘數學家 現在被認為是幾何之父", cut_all=True)
2. print("Full Mode: " + "/ ".join(seg_list))  # 全模式
```

```
In [67]: seg_list = jieba.cut("西元前三世紀的古希臘數學家 現在被認為是幾何之父", cut_all=True)
         print("Full Mode: " + "/ ".join(seg_list))  # 全模式

         Full Mode: 西元/ 西元前/ 三世/ 紀/ 的/ 古/ 希/ 臘/ 數/ 學/ 家/ / / 現/ 在/ 被/ 認/ 為/ 是/ 幾/ 何/ 之父
```

Jieba 基本分詞

- 精確模式(預設模式)

1. seg_list = jieba.cut("西元前三世紀的古希臘數學家 現在被認為是幾何之父", cut_all=False)
2. print(“Default Mode: ” + “/ ”.join(seg_list)) # 精確模式

```
In [70]: seg_list = jieba.cut("西元前三世紀的古希臘數學家 現在被認為是幾何之父", cut_all=False)
         print("Default Mode: " + "/ ".join(seg_list))  # 精確模式

         Default Mode: 西元前/ 三世/ 紀的/ 古希臘/ 數學家/  / 現在/ 被/ 認為/ 是/ 幾何/ 之父
```

Jieba 基本分詞

- 搜尋引擎模式

1. seg_list = jieba. cut_for_search("西元前三世紀的古希臘數學家 現在被認為是幾何之父", cut_all=True)
2. print(“Search Mode: ” + “/ ”.join(seg_list)) # 搜尋引擎模式

```
In [71]: seg_list = jieba.cut_for_search("西元前三世紀的古希臘數學家 現在被認為是幾何之父")
         print("Search Mode: " + "/ ".join(seg_list))  # 搜尋引擎模式

         Search Mode: 西元/ 西元前/ 三世/ 紀的/ 古希臘/ 數學家/  / 現在/ 被/ 認為/ 是/ 幾何/ 之父
```

Jieba 基本分詞

- 各模式比較

In [69]:
```
seg_list = jieba.cut("西元前三世紀的古希臘數學家 現在被認為是幾何之父", cut_all=True)
print("Full Mode: " + "/ ".join(seg_list))  # 全模式
```

```
Full Mode: 西元/ 西元前/ 三世/ 紀/ 的/ 古/ 希/ 臘/ 數/ 學/ 家/ / / 現/ 在/ 被/ 認/ 為/ 是/ 幾/ 何/ 之父
```

In [70]:
```
seg_list = jieba.cut("西元前三世紀的古希臘數學家 現在被認為是幾何之父", cut_all=False)
print("Default Mode: " + "/ ".join(seg_list))  # 精確模式
```

```
Default Mode: 西元前/ 三世/ 紀的/ 古希臘/ 數學家/  / 現在/ 被/ 認為/ 是/ 幾何/ 之父
```

In [71]:
```
seg_list = jieba.cut_for_search("西元前三世紀的古希臘數學家 現在被認為是幾何之父")
print("Search Mode: " + "/ ".join(seg_list))  # 搜尋引擎模式
```

```
Search Mode: 西元/ 西元前/ 三世/ 紀的/ 古希臘/ 數學家/  / 現在/ 被/ 認為/ 是/ 幾何/ 之父
```