Word2Vec

2017.3.24

陳冠穎

上次只有說明 **skip-gram** 搭配

negative sampling 的架構

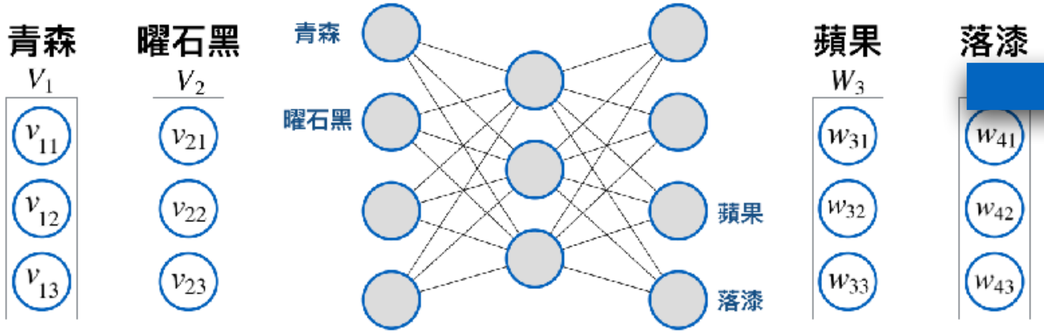
說好的下回待續

前情提要

Word2Vec

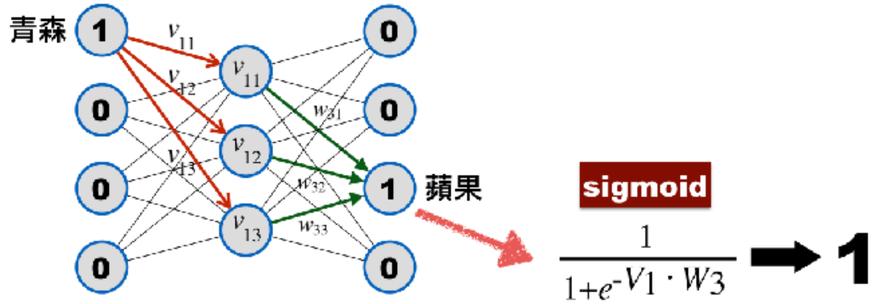
隨機初始化每個詞彙的詞向量

（隨機給每個 **weight** 不同的數值）

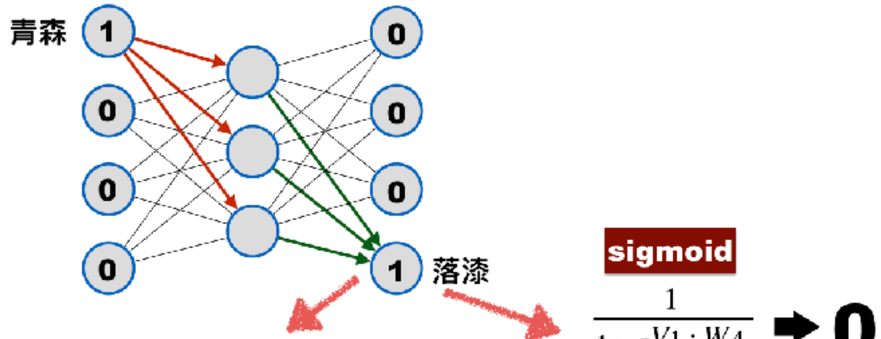


35

由於 蘋果 出現在 青森 的上下文中，所以要訓練類神經網路，在 蘋果 位置可以輸出 **1**



若 青森 的上下文不會出現 落漆



內積 $V_1 \cdot W_4 = v_{11}w_{41} + v_{12}w_{42} + v_{13}w_{43}$

49

Min. cost function: cross-entropy

$$J = -\log\left(\frac{1}{1+e^{-V_I^T \cdot W_{pos}}}\right) - \sum_{neg} \log\left(1-\frac{1}{1+e^{-V_I^T \cdot W_{neg}}}\right)$$

希望輸出 **1**　　希望輸出 **0**

0

其中 V_I 為輸入端的詞向量，而 W_{pos} 和 W_{neg} 為輸出端的詞向量。

通常，對於每筆 V_I 而言，會找一個 **positive example** 和多個 **negative example** ，因此用 $\sum$ 將這些 **negative example** 算出來的結果加起來。

57

一個詞彙 = 一個向量

利用 cosine similarity 計算兩個詞彙的關聯性

詞彙的向量化

和前後文息息相關

Word2Vec 原理

將詞彙做 one-hot encoding，再透過 word2vec 類神經網路計算，求出降維後的詞向量（語意向量）

Word2Vec 假設

擁有相同上下文的詞彙，具有相同的語意

實際上

Word2Vec 有兩種模型架構，有兩種求解策略

輸入層

{ CBOW
skip-gram

輸出層

{ hierarchical softmax
negative sampling

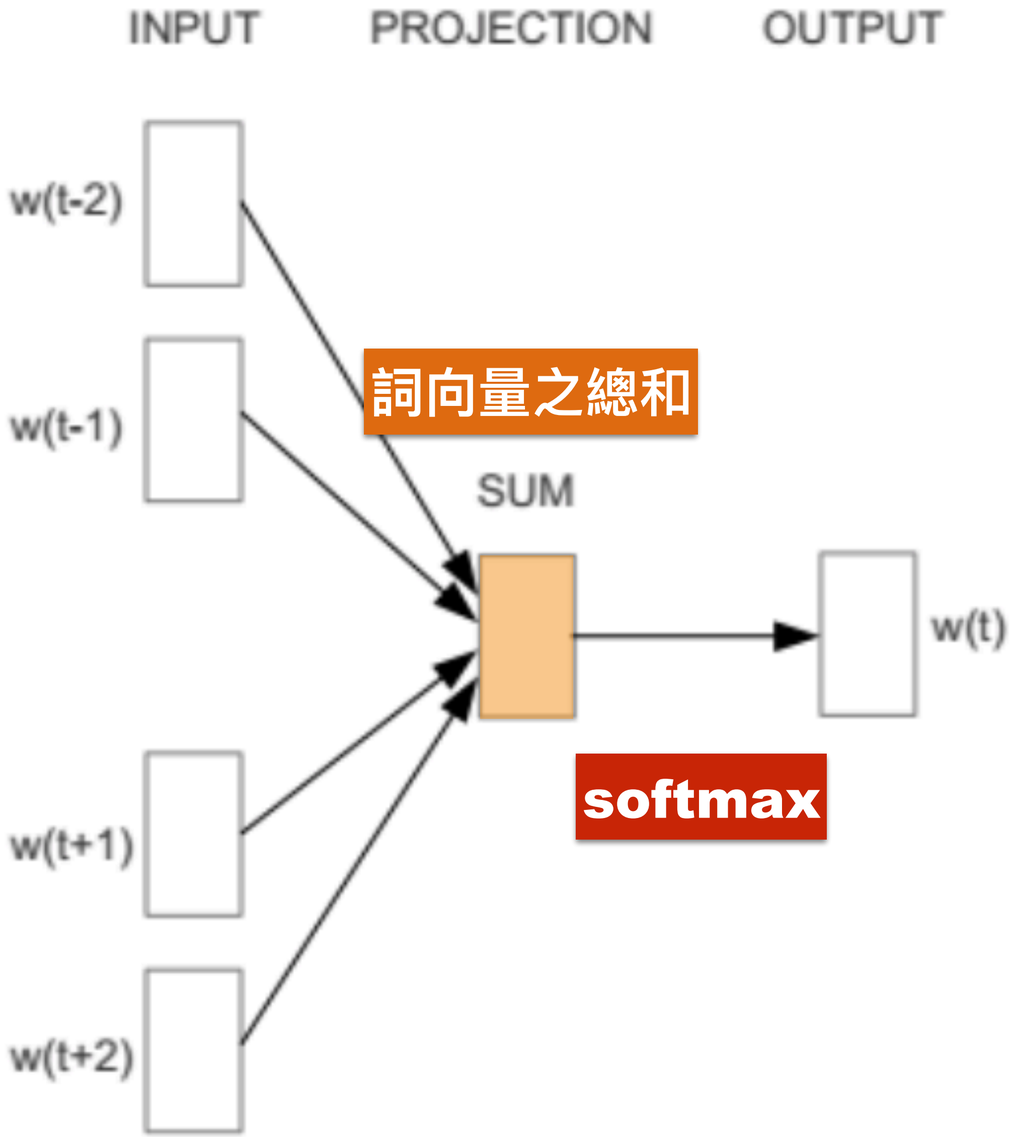
目的：減少於輸出層的訓練時間

CBOW: P(word | context)

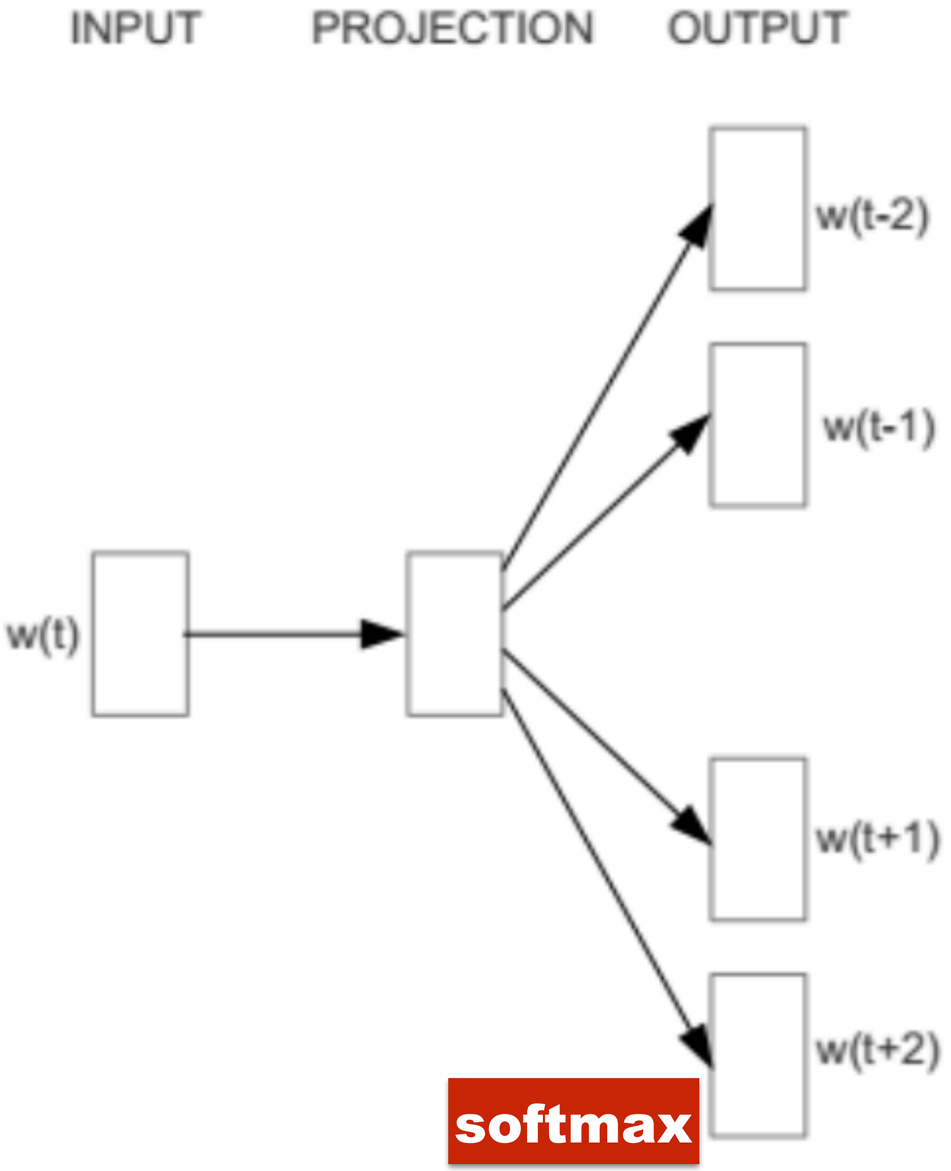
輸入 context word ➡ 訓練 target word

skip-gram: P(context | word)

輸入 target word ➡ 訓練 context word



CBOW



Skip-gram

CBOW 對於 context 資訊是以「詞向量之和」來表示，skip-gram 對於 context 中的每一個詞都會單獨計算。

skip-gram 的計算比 CBOW 複雜、慢

skip-gram 對於「低頻詞」效果較好

In CBOW the vectors from the context words are averaged before predicting the center word. In skip-gram there is no averaging of embedding vectors. It seems like the model can learn better representations for the rare words when their vectors are not averaged with the other context words in the process of making the predictions.

negative sampling

負樣本抽樣：

「高頻詞」被選為「負樣本（模型要輸出 0）」的機率較高

hierarchical softmax

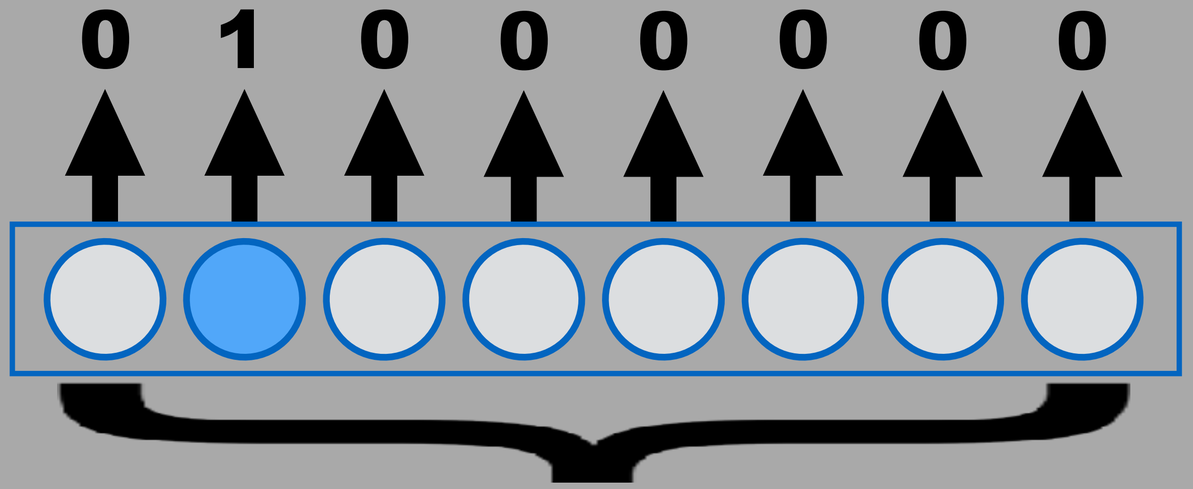
階層式 softmax：

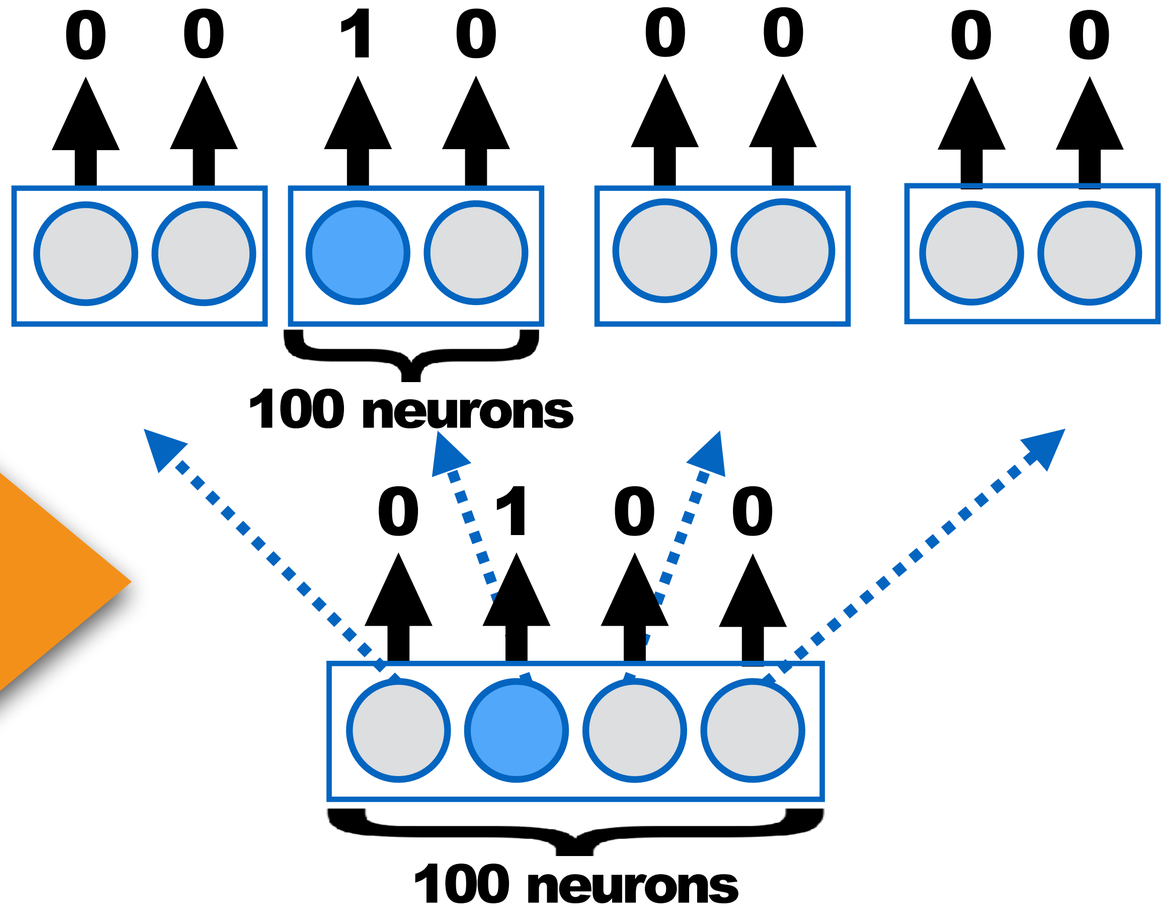
將輸出層的字做階層式分類（二元分類樹），減少模型訓練時間

Before

Hierarchical Softmax

After





如果 Y 有 10000 種字，訓練時就要直接對 P(Y=y|X=x) 做計算，即是對這 10000 種字做計算，使 y 所對應的神經元輸出為 1 ，其它 9999 個神經元輸出 0 ，這樣要計算 10000 次

只要計算 200 次

Word2Vec 參數建議

- 詞向量維度：通常越高越好，但耗記憶體
- **Window size： Skip-gram 一般設 10 ， CBOW 一般設 5**
- 當語料的句子太短時，設置過大的 **window size** 會導致結果不理想

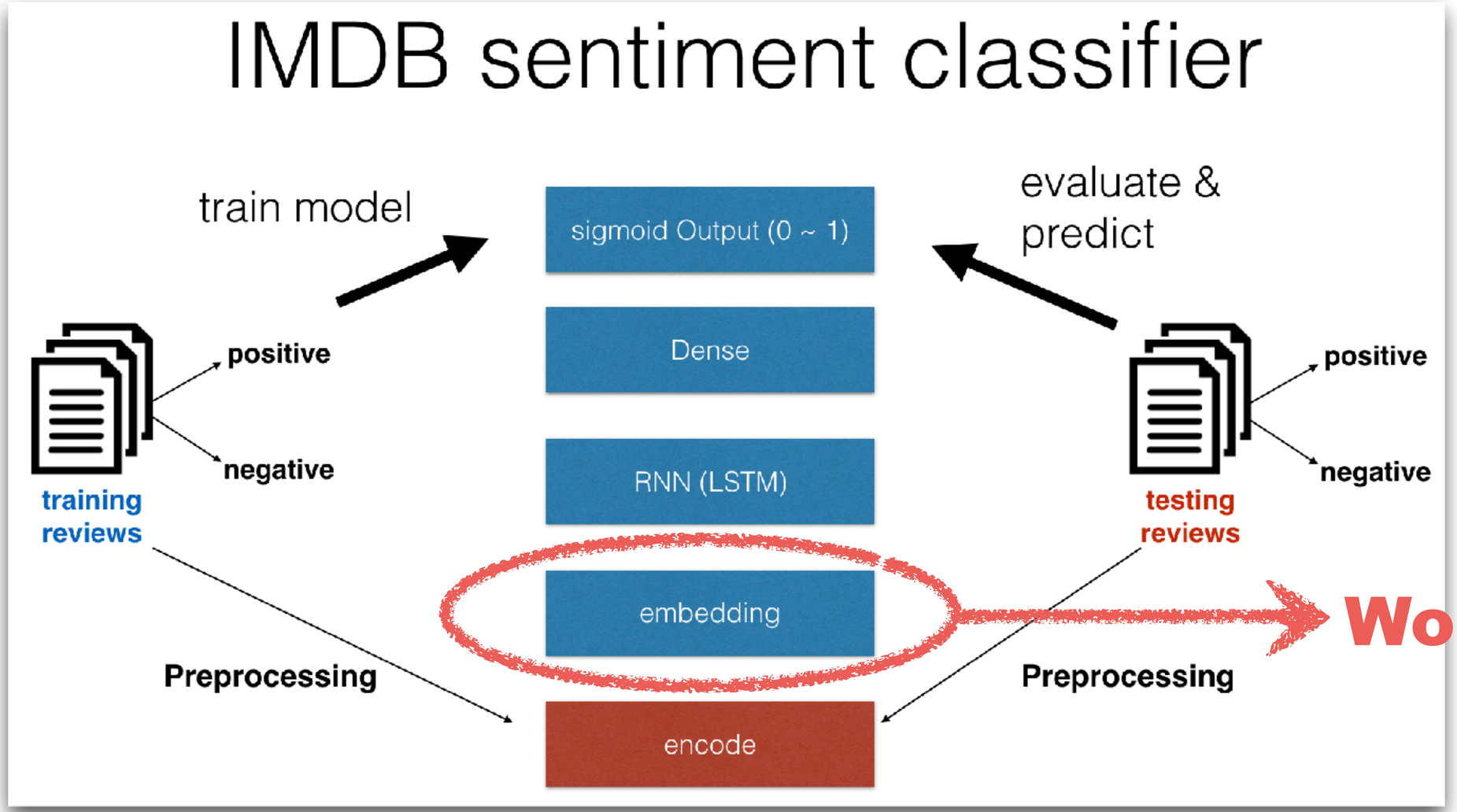
我的心得

Word2Vec 假設「處於相似上下文中的詞，具有相似的含義」。一般認為這種相似包含語意層面的相似，但實際上卻很難用自然語言、語意去做解釋。

但 **Word2Vec** 卻很適合將詞向量作為 **feature** 運用在其他機器學習模型中！

或是套用在其他「具有上下文和中心詞」的類比關係之資料分析中！

Word2Vec + RNN



OOD

2017.3.24

陳冠穎

OO 的概念


KEY CUTTING MACHINE
VOLTS 220V


BALDWIN

類別 (class): 勾勒藍圖

具體化: 執行建構子

(一對多關係)

實體 (instance): 實際使用的物件

封裝 (encapsulation)

- 物件開放讓他人存取的東西應該是「最少且必要的」

- 適當保持私密性：
  適當地將類別的 member 或 method 設為私有的 (private)

- 間接讀取私密資料：
  針對 private member 設置 setter & getter

封裝 (encapsulation)

封裝讓程式碼易於理解，藉由：

1. 類別的名稱
2. 類別的行為

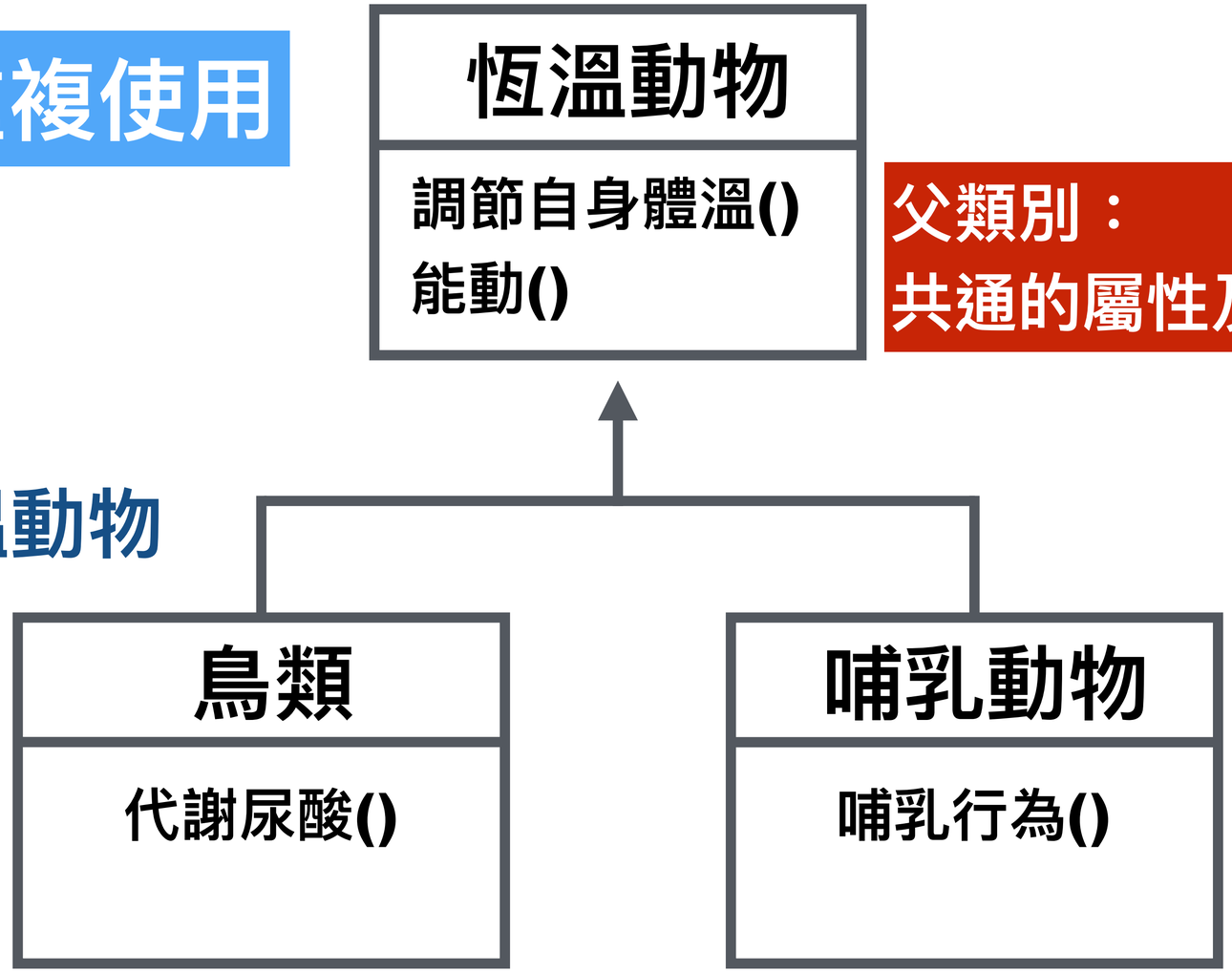
易於理解它和系統之間的關係與職責，此特性可以縮短日後重新檢視程式碼時所花費的時間。

繼承 (inheritance)

繼承讓程式碼可以重複使用

鳥類 is a 恆溫動物

鳥類 is a kind of 恆溫動物



OO 的優點

在建構大系統時，更加能夠體會到

提高程式碼的重複使用性

(code reusability)

避免重複的程式碼

容易擴充、維護程式碼

讓你的程式碼越寫越少

花費時間分析物件的責任歸屬、
分析物件之間共有的責任與資料

OO 的缺點

初期撰寫的時間成本高，日後才能享受到好處

OOD 的精神

單一職責原則

(single responsibility principle)

一個類別只有一種職責，
也避免單一職責分散在各個類別中

不斷地思考...

日後要如何**維護**程式碼

水能載舟亦能覆舟

是不是所有程式都需要 OO？

刺激的來了...

Code Review

我寫了一手廢 code...

拜託打臉

Before

1. 讀取設定：指定文章來源、時間範圍
2. 讀取Hive資料庫
   - 存取crawler DB下面的資料表
   - 存取segment資料表
   - 資料合併：做join，並且去重複
3. 資料前處理
   - 去除數字、英文標點符號
   - 去除英文字 (當作stop words)
   - 計算TF值：主文ID --> word
   - 計算TF-IDF值
4. 主題模型
   - 資料準備 (整理資料格式)
   - LDA建模
   - 主題的關鍵字分布
   - 文章的主題分布
   - 取出主題的代表文章
5. 網路聲量分析
   - 繪製LDA模型的主題-關鍵字機率圖
   - 關鍵字聲量： weighted method
   - 關鍵字聲量： merged method
   - 主題聲量
   - 繪製時間趨勢變化圖
   - 整理不分時間軸資料：SNA圖的輸入資料

After

The End.