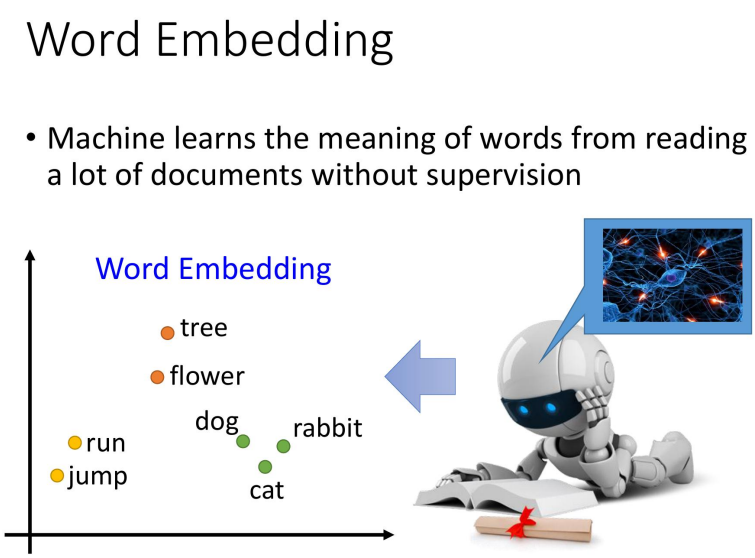
臺灣大學人工智慧中心
科技部人工智慧技術暨全幅健康照護聯合研究中心
http://aintu.tw

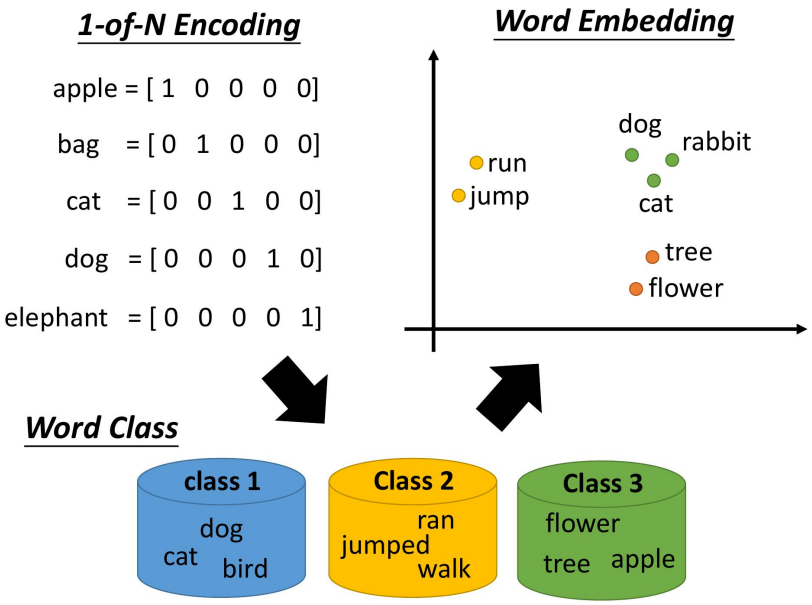
Word Embedding



Word Embedding 是 Dimension Reduction的一種。主要的過程便是用Vector來表示Word。主要有兩種方法：

1-of-N encoding

第一種叫做 1-of-N encoding，每一個 word，用一個vector來表示
這個 vector 的 dimension，是所有的 word 數目。



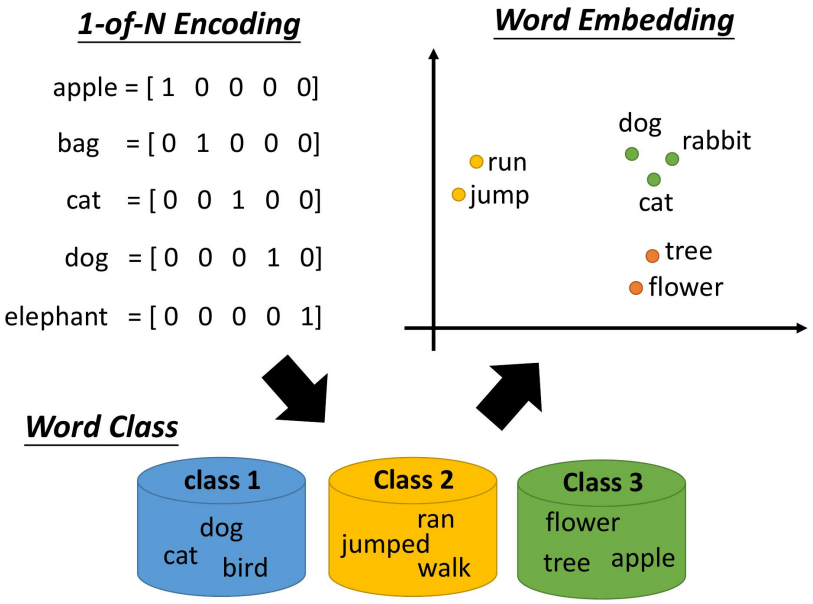
假設這個世界上有十萬個 word，那 1-of-N encoding 的 dimension就是十萬維。每一個 word，對應到其中一維。

所以，對apple這個word的vector來說 ，它的第一維是 1，其他都是 0。bag 就是第二維是 1，cat 就是第三維是 1，以此類推...

缺點：如果用這種方式來描述一個 word，vector 會比較不 informative。
由於每一個 word，它的 vector 都沒什麼關聯，所以從這個 vector 裡面，沒有辦法得到其他的資訊。（比如說，cat 跟 dog都是動物這件事，你便沒辦法知道）
因此之後就有另一個方法就做Word Class。

A clustering method: Word Class

Word Class，就是把有同樣性質的 word，cluster 成一群一群的，然後用那一個 word 所屬的 class 來表示這個 word。這個就是 Dimension Reduction 裡面 clustering 的概念。



比如說，dog, cat 跟 bird，它們都是class 1，ran, jumped, walk 是 class 2，flower, tree, apple 是 class 3，等等...

但是，光用class 其實是不夠的。光做 clustering 的話，一些 information（比如說，動物的 class與植物的 class，都屬於生物）便沒有辦法被呈現出來。class與class之間的關聯無法被顯現。

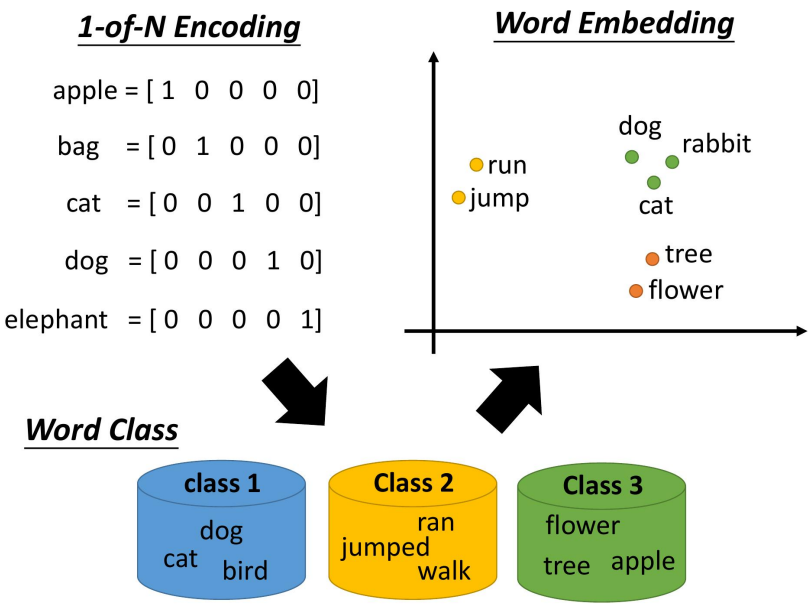
因此，我們需要的是 Word Embedding。

Word Embedding

Word Embedding 就是把每一個 word都 project 到一個 high dimensional 的 space 上面。

雖然說，這個 space 是 high dimensional的，但是它的維度其實遠比 1-of-N encoding 的 dimension 還要低。

實際上，對1-of-N encoding來說，英文有 10 萬詞彙，它就是 10 萬維但如果是用 Word Embedding，通常是 50 維、100維的 space就可以了。或者可以說從 1-of-N encoding 到 Word Embedding，就是一個 Dimension Reduction 的 process。



以上面這張圖word embedding 的部分為例，我們可以看到的結果是類似 semantic，類似語意的詞彙在這個圖上是比較接近的。而且在這個 Word Embedding 的 space 裡面，每一個 dimension，可能都有它特別的含意。比如說上圖的橫軸的dimension可能就代表了生物和其他的非生物之間的差別，而縱軸這個 dimension 可能就代表了跟動作與靜止的關聯。

那怎麼做 Word Embedding呢？

Word Embedding 是一個 unsupervised 的 approach。

要讓 machine知道每一個詞彙的含義是什麼，需要讓 machine 閱讀大量的文章，來確定embedding後的feature vector 應該長什麼樣子。

這是一個 unsupervised 的 problem，實際上做的事情就是learn 一個 Neural Network， input 是詞彙，output 則是那一個詞彙所對應的 Word Embedding裡的 vector。

然而我們手上有的 train data只是一大堆的文字，所以我們只有Input，但是我們沒有 output，我們不知道每一個 Word Embedding 應該長什麼樣子。所以對於我們要找的 NN function，我們只有單項的輸入，不知道輸出。

因此，這才是一個 unsupervised learning 的問題。

Can Auto-encoder solve word embedding?

前面的章節有講解過一個 deep learning base 的 Dimension Reduction，叫做 Auto-encoder。

實際上需要的做法是 learn 一個 network，讓它輸入等於輸出，再將某一個 hidden layer 拿出來，作為 Dimension Reduction 的結果。

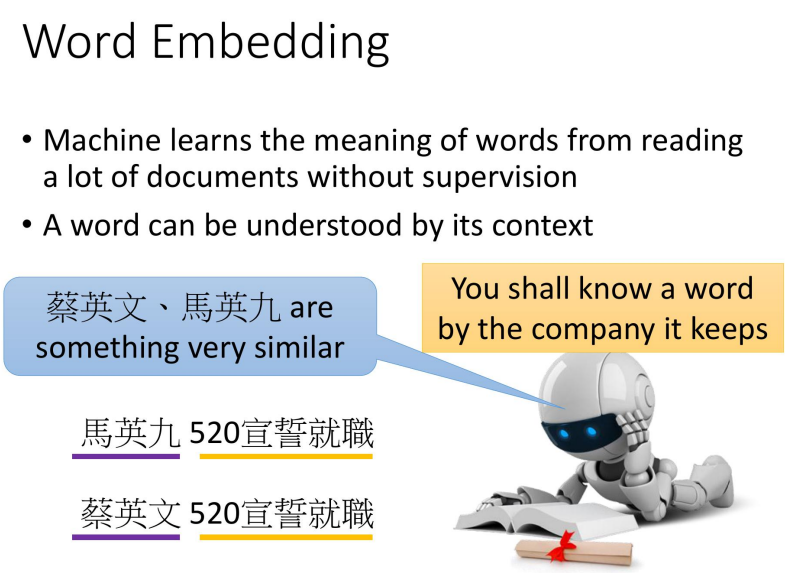
然而在這個地方，並沒有辦法用 Auto-encoder 來解

舉例來說：

如果用 1-of-N encoding 當作它的 input，對 1-of-N encoding 來說，每一個詞彙都是 independent 的。這樣子的 vector 做 Auto-encoder，沒有辦法 learn 出任何 informative 的 information。就算用character 的 n-gram 來描述一個 word，也只可以抓到一些字首、字根的含義。所以基本上，現在大家並不是這麼做。

在 Word Embedding 這個 task 裡面，沒有辦法使用Auto-encoder

Word Embedding Implementation



> Word Embedding基本的精神：每一個詞彙的含義，可以根據它的上下文來得到。

舉例來說：

假設機器讀了一段文字：『馬英九520宣誓就職』，它也讀了另外一段新聞：『蔡英文520宣誓就職』。

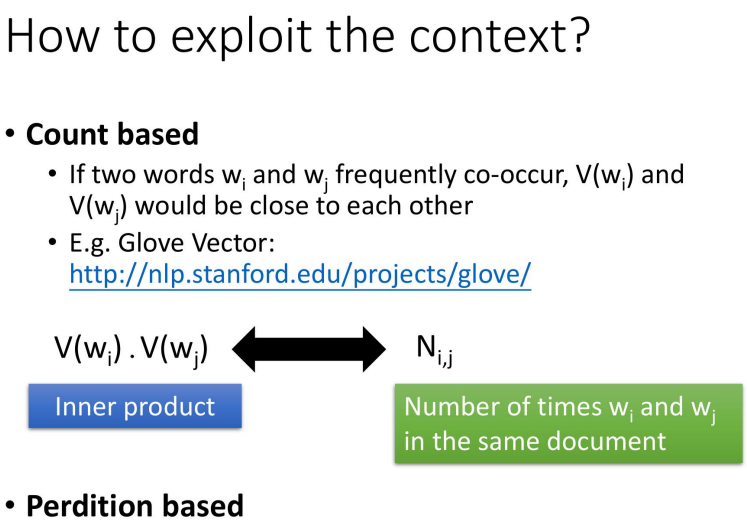
對機器來說，雖然它不知道馬英九指的是什麼，也不知道蔡英文指的是什麼，但是馬英九後面有接520宣誓就職，蔡英文的後面也有接520宣誓就職。對機器來說，只要它讀了大量的文章並且發現馬英九跟蔡英文前後都有類似的 context機器就可以推論說：「蔡英文跟馬英九代表了某種有關係的object」。它可能也不知道他們是人，但它知道，蔡英文跟馬英九這兩個詞彙代表了，同樣地位的某種東西。

那怎麼來實做這一件事呢？

怎麼用這個基本的精神來找出 Word Embedding 的 vector，有兩個不同體系的作法

- Count based 的方法
- Prediction based 的方法

Count-based method



- 概念：現在有兩個詞彙，w_i, w_j，它們有時候會在同一個文章中出現（co-occur）。我們用 V(w_i) 來代表w_i 的 word vector，我們用 V(w_j) 來代表w_j 的 word vector。如果 w_i 跟 w_j 常常一起出現的話，V(wi) 跟 V(wj) 就會比較接近。

這種方法，有一個很代表性的例子，叫做 Glove vector。

假設我們知道：wi 的 word vector 是 V(wi), wj 的 word vector 是 V(wj)。

計算它們的 inner product（假設為$N_i j$） ，則為 wi 跟 wj co-occur 在同樣的 document 裡面的次數。

Count-based的目標就是為 wi 找一組 vector，也為 wj 找一個組 vector，co-occur的次數跟inner product的直越接近越好。

這個概念與 LSA 和 matrix factorization 的概念類似。

Prediction-based method

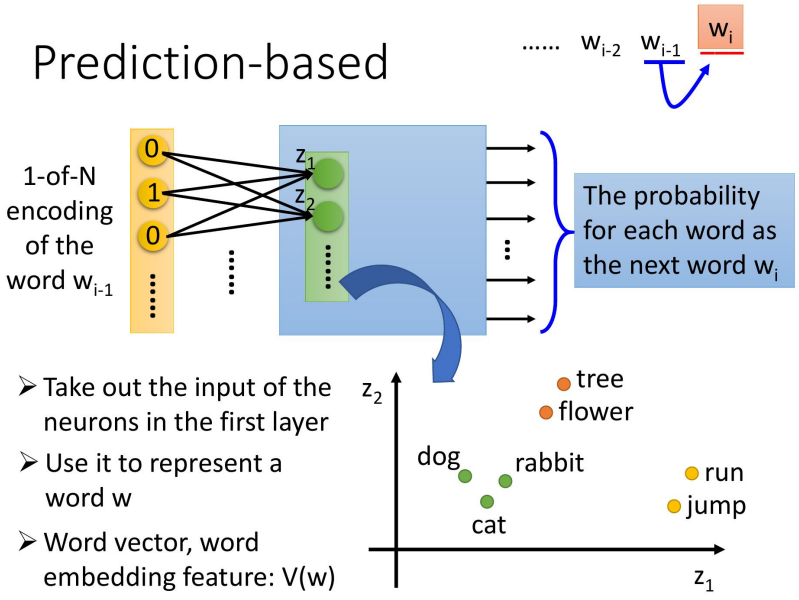
Prediction based 方法跟 Count based 的方法，並沒有誰優誰劣。

Prediction based 的基本想法：learn 一個 neural network。這邊每一個 w 代表一個 word (下標 i-1)，這個 prediction的model這個neural network，它的工作是要 predict 下一個可能出現的 word。

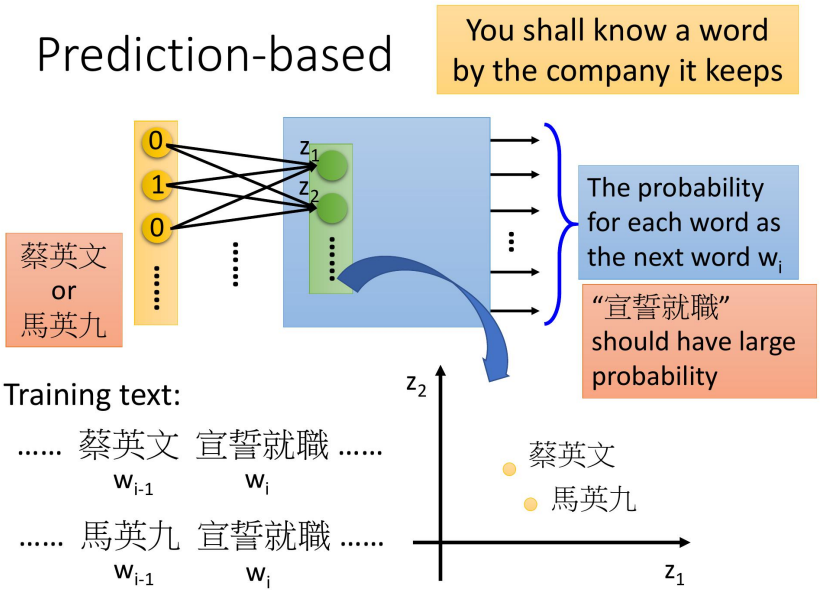
每一個 word則用 1-of-N encoding把它表示成一個 feature vector。

- input: w(下標 i-1) 的 1-of-N encoding 的 feature vector
- output: 下一個 word, wi 是某一個 word 的機率

假設現在世界上有 10 萬個 word, 這個 model 的 output 就是 10 萬維, 每一維代表了某一個 word, 是下一個 word 的機率。



Prediction Based 概念舉例



Prediction based 的方法，可以根據一個詞彙的上下文，來了解一個詞彙的涵義。

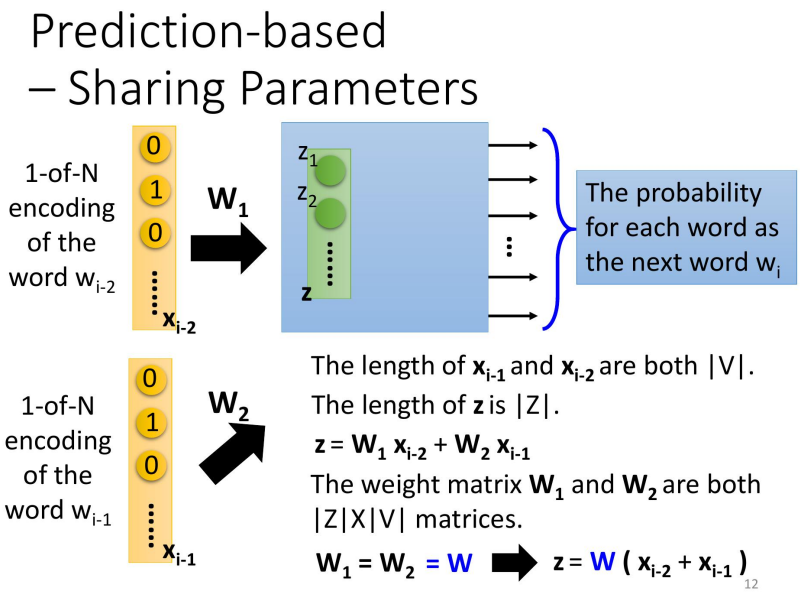
實際例子

假設我們的 training data 裡，有一個文章是馬英九跟蔡英文宣誓就職，另外一個是馬英九宣誓就職，在第一個句子裡面蔡英文是 w(下標 i-1)，宣誓就職是 w(下標 i)，在另外一篇文章裡面，馬英九是 w(下標 i-1)，宣誓就職是 w(下標 i)。在這個 Prediction model 裡，不論是 input 蔡英文，還是馬英九的 1-of-N encoding，都會希望 learn 出來的結果是宣誓就職的機率比較大，因為馬英九和蔡英文後面接宣誓就職的機率都很高。

蔡英文和馬英九雖然是不同的 input, 但是最後在 output 的地方得到了一樣的 output，代表中間的 hidden layer 必需要把他們 project 到同樣的空間。所以把這個 prediction model 的第一個 hidden layer 拿出來就可以得到這種 word embedding 的特性。

如果只用 w(下標 i-1)去 predict w(下標 i)，有時候會太弱。就算是人，你給一個詞彙要 predict 下一個詞彙，也並不簡單。

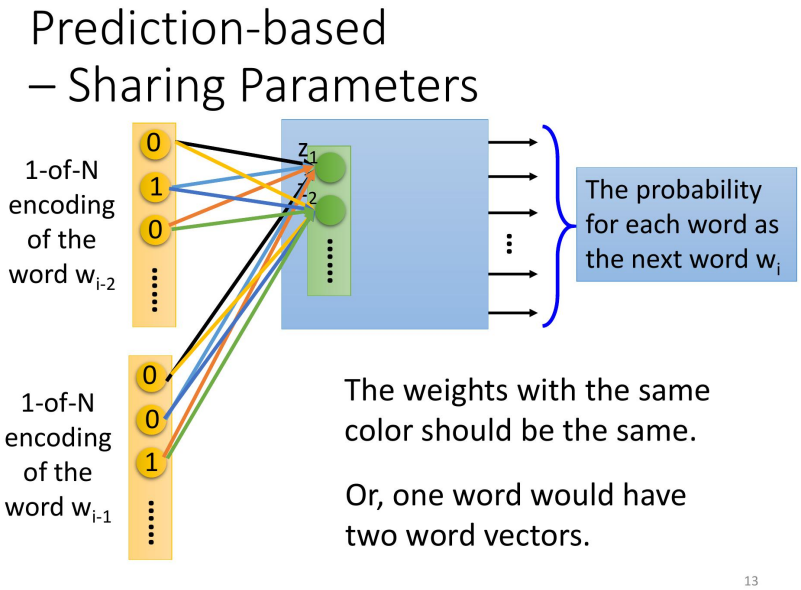
Prediction-based 延伸



因此，可以w(下標 i-2) 跟 w(下標 i-1)，predict 下一個 word, w(下標 i)，也可以把這個 model 拓展到 N 個詞彙。一般如果真的要 learn 這樣的 word vector 的話，你能會需要你input是至少 10 個詞彙，才能夠 learn 出比較 reasonable 的結果。

但是實際上，會希望跟 w(下標 i-2) 相連的 weight跟和 w(下標 i-1) 相連的 weight是被 tight 在一起的。

也就是說w(下標 i-2) 的第一個 dimension跟第一個 hidden layer 的第一個 neuron它們中間連的 weight和 w(下標 i-1) 的第一個 dimension和第一個 hidden layer 的 neuron，它們之間連的位置這兩個 weight 必須是一樣的。

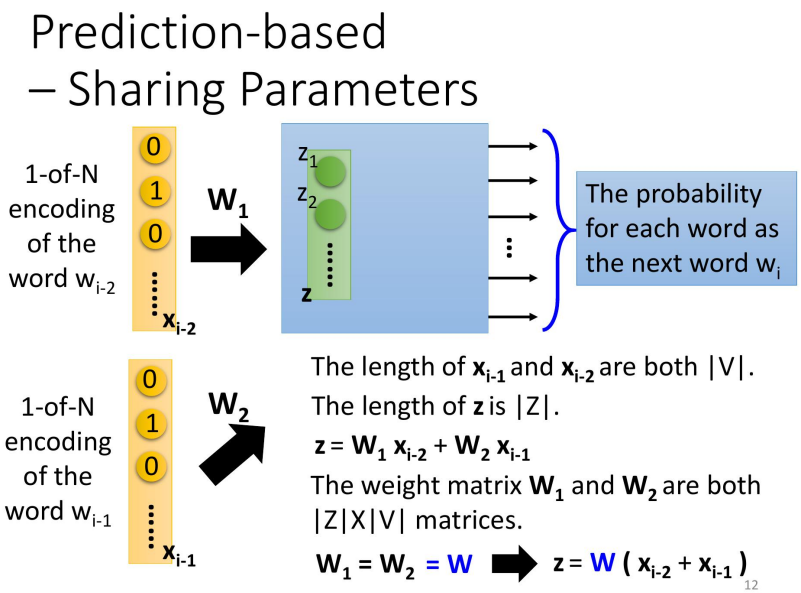


- 如果不這麼做，同一個 word 放在w(下標 i-2) 的位置跟放在 w(下標 i-1) 的位置, 通過這個

transform得到的 embedding 就會不一樣。

- 這樣做也可以減少參數量，因為 input 的 dimension 很大，是十萬維，所以 feature vector，就算是50 維它也是一個非常非常、碩大無朋的 matrix讓所有的 1-of-N encoding後面接的 weight 是一樣的，就不會隨著你的 context 的增長而需要更多的參數。

用 formulation 來解釋：



假設 w(下標 i-2) 的 1-of-N encoding 是 X2，w(下標 i-1) 的 1-of-N encoding 是 X1，它們的長度都是 V 的絕對值。

- Z 等於 X(i-2) * W1 + X(i-1) * W2，把 X(i-2) * W1 + X(i-1) * W2，就會得到 Z
- W1 跟 W2都是一個 Z 乘上一個 V dimension 的 weight matrix
- 我們強制讓 W1 要等於 W2，兩個一模一樣的 matrix, W。

所以，實際上在處理這個問題的時候，可以把 X(i-2) 跟 X(i-1) 直接先加起來。因為 W1 跟 W2 是一樣的，可以把 W 提出來，也可以把 X(i-1) 跟X(i-2) 先加起來，再乘上 W 的這個 transform，就會得到 z。

如果要得到一個 word 的 vector ，就把一個 word 的 1-of-N encoding乘上這個 W，就可以得到那一個 word 的 Word Embedding。

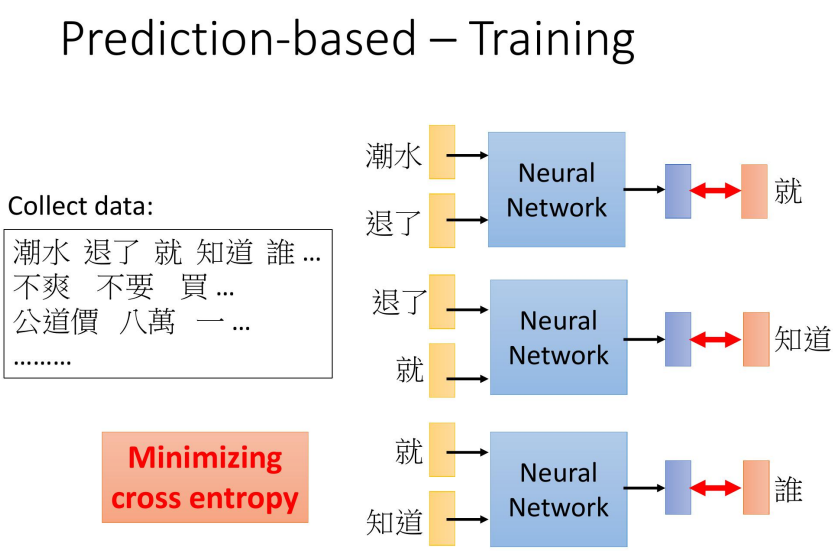
怎麼讓這個 W1 跟 W2 它們的 weight 一定一樣呢？

目標：假設我們現在有兩個 weight, wi 跟 wj，那我們希望 wi 跟 wj，它的 weight 是一樣的

解法：

- 訓練的時候要給它們一樣的初始值
- update的時候，wi 再減掉 wj 對 C 的偏微分，且 wj 再減掉 wi 對 C 的偏微分。

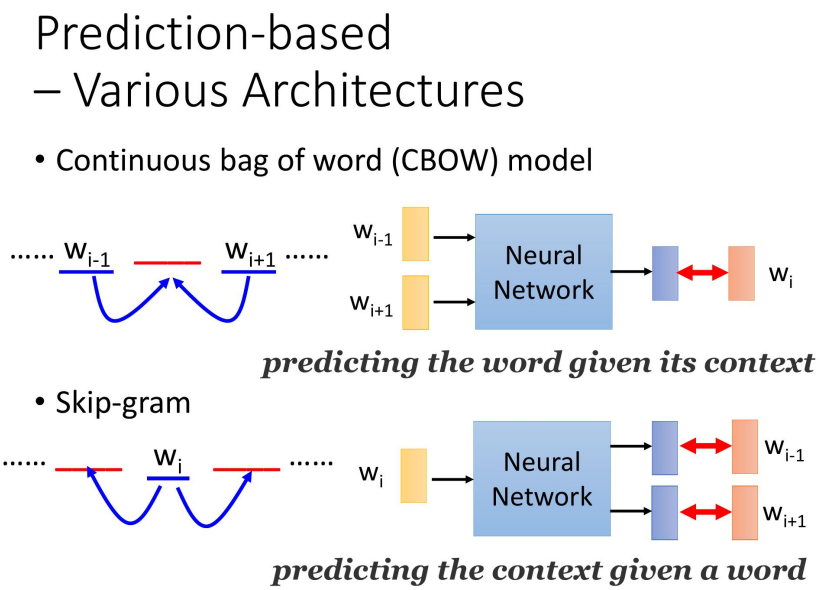
怎麼訓練 Prediction-based 的 network 呢？



- 屬於unsupervised 的 learning
- 寫一個程式爬一下八卦版的 data，就可以爬到一大堆文字 train 你的 model。
- 讓output 跟正確答案 的 cross entropy越小越好。

Ex. 讓你的 neural network input "潮水" 跟 "退了"，希望它的 output 是 "就" 這個樣子 "就" 也是一個 1-of-N encoding 來表示

其他架構



以上只是最基本的型態 Prediction based 的 model ，有種種的變形。且其performance在不同的 task上互有勝負。

以下有兩種作法：

- Continuous bag of word, (CBOW)
- Skip-gram

Continuous bag of word, (CBOW)

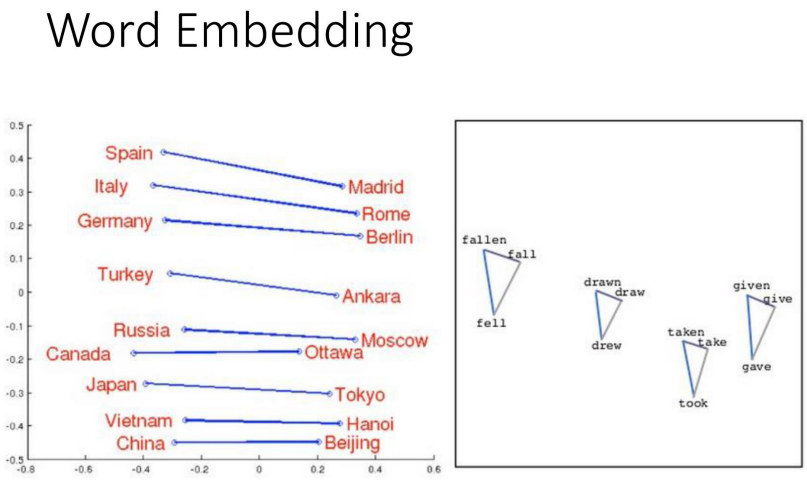
剛才是拿前面的詞彙，去 predict 接下來的詞彙，CBOW 的意思是拿某一個詞彙的 context去 predict 中間這個詞彙。也就是拿 W(i-1) 跟 W(i+1) 去 predict Wi。

Skip-gram

slip-gram是拿中間的詞彙去 predict 接下來的 context，也就是拿 Wi 去 predict W(i-1) 跟 W(i+1)。

given 中間的 word，我們要去 predict 它的周圍

Word Embedding Overview



Source: http://www.slideshare.net/hustwj/cikm-keynotenov2014

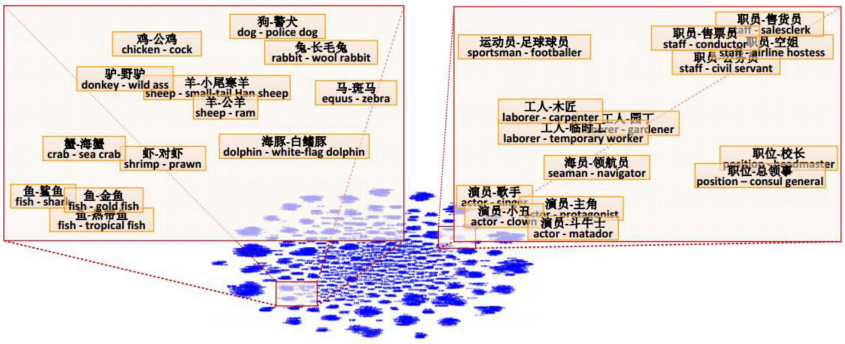
15

word vector 可以得到一些有趣的特性：

- 把同樣類型的東西的 word vector 擺在一起; 把 Italy跟它的首都 Rome 擺在一起; 把Germany 跟它的首都 Berlin 擺在一起; 我們把 Japan 跟它的首都 Tokyo 擺在一起。它們之間是有某種固定的關係的。
- 把一個動詞的三態擺在一起，會發現說同一個動詞的三態，它們中間有某種固定的關係，成為這個三角形。

所以從這個 word vector 裡面，可以 discover 我們所不知道的 word 跟 word 之間的關係。

Word Embedding



Fu, Ruiji, et al. "Learning semantic hierarchies via word embeddings."Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics: Long Papers. Vol. 1. 2014.

16

如果把兩個 word vector 和 word vector 之間，兩兩相減，然後 project 到一個 2 dimensional 的 space 上面。如果它得到的結果是落在這個位置的話，那這兩個 word vector 之間，它們就有，比如說某一個 word 是包含於某一個 word 之間的關係。

Ex. 把海豚跟會轉彎的白海豚相減，它的 vector 落在這邊; 把演員跟主角相減，落在這一邊; 把工人跟木匠相減，落在這邊; 把職員跟售貨員相減，落在這一邊; 把羊跟公羊相減，落在這邊。

如果，某一個東西是屬於另外一個東西的話，它們兩個 word vector 相減結果會是很類似的。

Word Embedding

- Characteristics

$$V(Germany) \approx V(Berlin) - V(Rome) + V(Italy)$$

$$V(hotter) - V(hot) \approx V(bigger) - V(big)$$
$$V(Rome) - V(Italy) \approx V(Berlin) - V(Germany)$$
$$V(king) - V(queen) \approx V(uncle) - V(aunt)$$

- Solving analogies

Rome : Italy = Berlin : ?

Compute $V(Berlin) - V(Rome) + V(Italy)$

Find the word w with the closest V(w)

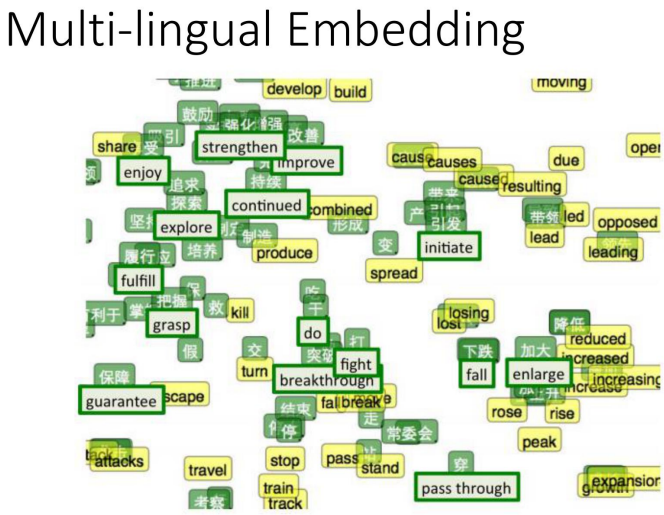
17

所用 word vector 的的概念，可以做一些簡單的推論：

舉例來說，

- hotter 的 word vector減掉 hot 的 word vector 會很接近 bigger 的 word vector 減掉 big 的 word vector 。
- Rome 的 vector 減掉 Italy 的 vector會很接近 Berlin 的 vector 減掉 Germany 的 vector 。
- King 的 vector 減掉 queen 的 vector 會很接近uncle 的 vector 減掉 aunt 的 vector。

Multi-lingual Embedding



Bilingual Word Embeddings for Phrase-Based Machine Translation, Will Zou, Richard Socher, Daniel Cer and Christopher Manning, EMNLP, 2013

那 word vector 還可以做很多其他的事情，可以把不同的語言的 word vector拉在一起，

如果今天有一個中文的 corpus，和一個英文的 corpus，各自去、分別去 train 一組 word vector。則中文跟英文的 word vector是完全沒有任何的關係的。因為你要 train word vector 的時候，憑藉的就是上下文之間的關係。所以，如果沒有中文跟英文的句子混雜在一起，machine 就沒有辦法判斷中文的詞彙跟英文的詞彙他們之間的關係。

今天假如你已經事先知道某幾個中文的詞彙和某幾個英文的詞彙，它們是對應在一起的

則先得到一組中文的 vector 再得到一組英文的 vector，可以再 learn 一個 model把中文和英文對應的詞彙。

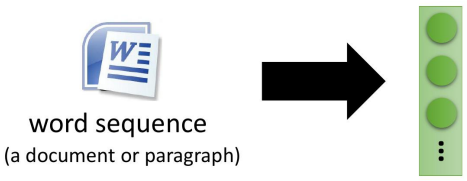
Ex. 知道 "加大" 對應到 "enlarge"，"下跌" 對應到 "fall" 把對應的詞彙，通過這個 projection 以後，把它們 project 在 space 上面的同一個點。

接下來有新的中文的詞彙和新的英文的詞彙，你都可以用同樣的 projection把它們 project 到同一個 space 上面。

Document Embedding

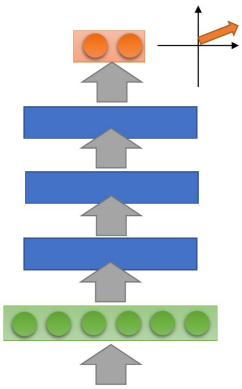
Document Embedding

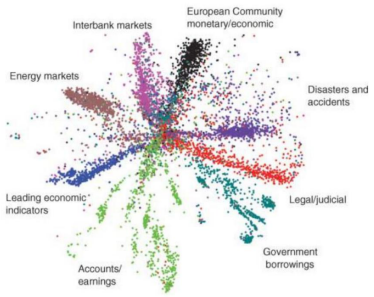
- word sequences with different lengths → the vector with the same length
  - The vector representing the meaning of the word sequence
  - A word sequence can be a document or a paragraph



不只是把一個 word 變成一個 vector，也可以把一個 document 變成一個 vector。

Semantic Embedding



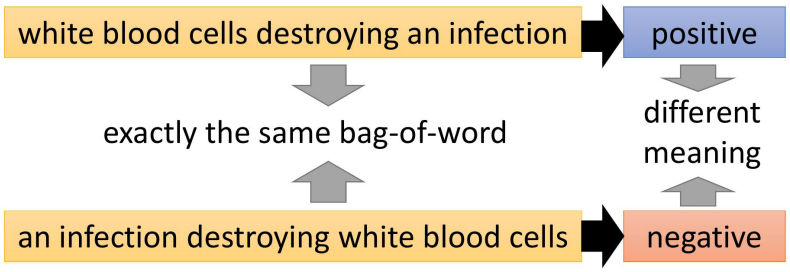


Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

就是把一個 document 變成一個 bag-of-word，然後，用 Auto-encoder learn 出這個 document 的 Semantic Embedding。

Beyond Bag of Word

- To understand the meaning of a word sequence, the order of the words can not be ignored.



23

但光用這個 word 來描述一篇 document是不夠的。

因為詞彙的順序代表了很重要的含義。

> 舉例來說，有兩個詞彙，有兩個句子
>
> 一個是： white blood cells destroying an infection
>
> 另外一個是：an infection destroying white blood cells
>
> 這兩句話，如果看它的 bag-of-word 的話，它們的 bag-of-word 是一模一樣的。它們都有出現這 6 個詞彙，但是順序是不一樣的。
>
> 雖然說，它們有同樣的 bag-of-word，它們在語意上，完全是不一樣的。所以，光只是用 bag-of-word 來描述 一篇 document 是非常不足的。

用 bag-of-word 會失去很多重要的 information。

Beyond Bag of Word

- ***Paragraph Vector***: Le, Quoc, and Tomas Mikolov. "Distributed Representations of Sentences and Documents." ICML, 2014
- ***Seq2seq Auto-encoder***: Li, Jiwei, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents." arXiv preprint, 2015
- ***Skip Thought***: Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler, "Skip-Thought Vectors" arXiv preprint, 2015.

這邊就列了 reference 給大家參考

上面這 3 個方法，它是 unsupervised，也就是說你只要 collect一大堆 document就可以讓它自己去學。

臺灣大學人工智慧中心
科技部人工智慧技術暨全幅健康照護聯合研究中心
http://aintu.tw