自然語言處理實作

向量式語意學 Vector Semantics

由次數到預測為本的向量

由詞的向量到搭配的向量

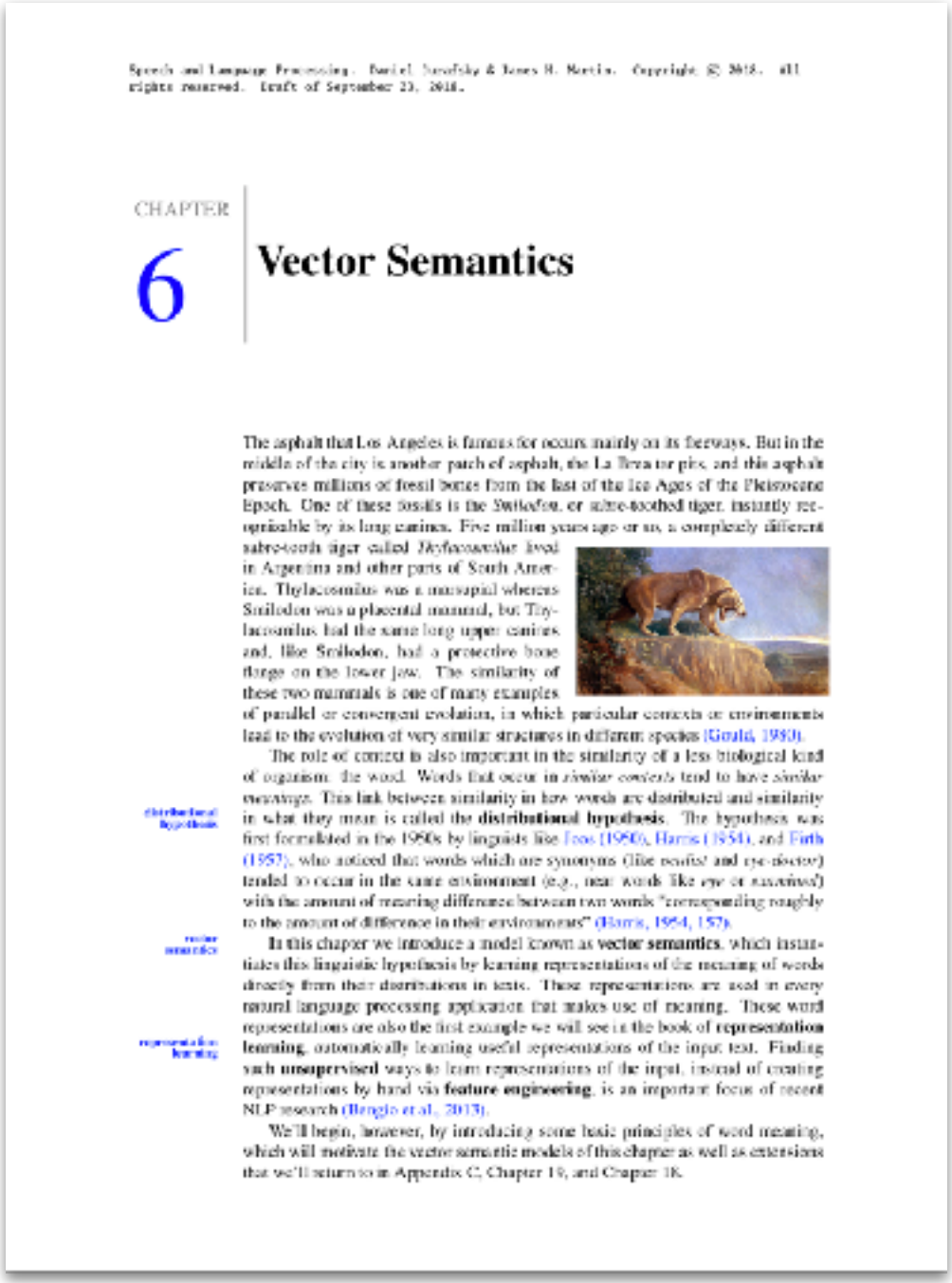
https://web.stanford.edu/~jurafsky/slp3/ Dan Jurafsky and James Martin (2018) Speech and Language Processing (draft)

第 6 章本文：

1. https://web.stanford.edu/~jurafsky/slp3/6.pdf

第 6 章投影片：

1. https://web.stanford.edu/~jurafsky/slp3/slides/vector1.pdf
2. https://web.stanford.edu/~jurafsky/slp3/slides/vector2.pdf
3. https://web.stanford.edu/~jurafsky/li15/lec3.vector.pdf



Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2018. All rights reserved. Draft of September 23, 2018.

CHAPTER

6 Vector Semantics

The asphalt that Los Angeles is famous for occurs mainly on its freeways. But in the middle of the city is another patch of asphalt, the La Brea tar pits, and this asphalt preserves millions of fossil bones from the last of the Ice Ages of the Pleistocene Epoch. One of these fossils is the Smilodon, or saber-toothed tiger, instantly recognizable by its long canines. Five million years ago or so, a completely different saber-toothed tiger called *Thylacosmilus* lived in Argentina and other parts of South America. Thylacosmilus was a marsupial whereas Smilodon was a placental mammal, but Thylacosmilus had the same long upper canines and, like Smilodon, had a protective bone flange on the lower jaw. The similarity of these two mammals is one of many examples of parallel or convergent evolution, in which particular contexts or environments lead to the evolution of very similar structures in different species (Gould, 1980).

The role of context is also important in the similarity of a less biological kind of organism: the word. Words that occur in *similar contexts* tend to have *similar meanings*. This link between similarity in how words are distributed and similarity in what they mean is called the **distributional hypothesis**. The hypothesis was first formalized in the 1950s by linguists like Joos (1950), Harris (1954), and Firth (1957), who noticed that words which are synonyms (like *oculist* and *eye-doctor*) tended to occur in the same environment (e.g., near words like *eye* or *examined*) with the amount of meaning difference between two words "corresponding roughly to the amount of difference in their environments" (Harris, 1954, 157).

distributional hypothesis

In this chapter we introduce a model known as **vector semantics**, which instantiates this linguistic hypothesis by learning representations of the meaning of words directly from their distributions in texts. These representations are used in every natural language processing application that makes use of meaning. These word representations are also the first example we will see in the book of **representation learning**, automatically learning useful representations of the input text. Finding such **unsupervised** ways to learn representations of the input, instead of creating representations by hand via **feature engineering**, is an important focus of recent NLP research (Bengio et al., 2013).

vector semantics

representation learning

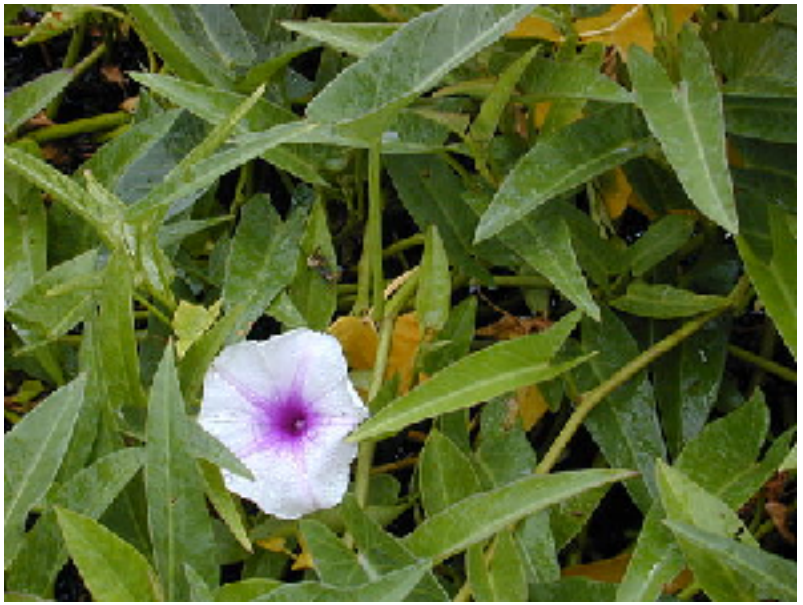
We'll begin, however, by introducing some basic principles of word meaning, which will motivate the vector semantic models of this chapter as well as extensions that we'll return to in Appendix C, Chapter 10, and Chapter 18.

用詞的使用情境定義詞意

- 動物在環境中演化成不同的樣子
- 詞也依其環境 (上下文) 發展出不同詞意
- Zellig Harris (1954)
  - 如果 A 和 B 有幾乎一模一樣的環境
  - 那我們就可以說 A 和 B 是同義詞 synonyms

例子：Ong choi *"Water Spinach"* 蕹菜

https://en.wikipedia.org/wiki/Ipomoea_aquatica



Yamaguchi, Wikimedia Commons, public domain

用上下文來定義 ongchoi 的詞意

假設我們有下列的句子

- Ong choi is delicious **sautéed with garlic**.
- Ong choi is superb **over rice**
- Ong choi **leaves** with salty sauces

而且，我們也看到

- … spinach **sautéed with garlic over rice**
- Chard stems and **leaves** are **delicious**
- Collard greens and other **salty** leafy greens

結論

- Ongchoi 是多葉的綠色蔬菜
- 類似 spinach, chard, collard greens

如何建立向量式語意模型

——用模型來計算相似度 similarity

每一個詞 = 一個向量

- 不再是字串 "word" 或編號 word45

相似詞在此向量空間彼此鄰近

詞意向量又稱為「詞內嵌」

NLP 過去稱之為「詞向量」最近有新作法，改名為「詞內嵌」

因為把詞嵌入「向量空間」又稱「詞內嵌」可代表細微語意和相似度

「詞內嵌」的優點h

- 如果用「詞內嵌」，那只需**相似詞**在訓練與測試資料，不需要 用「詞」本身出現在訓練資料

「詞內嵌」在 NLP 的任何應用，都可以用得上

- Word Sense Disambiguation, WSD
- Question Answering, QA
- Machine Translation, MT
- Chatbots
-

介紹兩種詞內嵌 embeddings

次數為本（詞對文件，詞對詞）

- 計算方式：count, tf-idf, PMI
- 常用基線模型 baseline model
- 稀疏向量 Sparse vectors
- 用鄰近的詞、次數所形成的向量來表示詞意（經驗法則）

Word2vec（預測詞旁邊的鄰居詞）

- 密集向量 Dense vectors
- 透過機器學習的方法，訓練一個分類器來學習最佳的向量 (用學習而非經驗法則)

計算向量的方法：次數、tf-idf、PMI

- 用次數的缺點：超高頻詞如 *the, it, they* 不太有資訊
- 用 tf-idf 可以反應頻率，但又避免超高頻詞的問題
- 用 Positive Pointwise Mutual Information (PPMI) 比較可以反應資訊量
  - 但對低頻詞有偏差（罕見詞反而有高PMI 值）
- 解決方案
  - 指定給罕見詞稍高的機率
  - 使用 加 1 平滑化策略 (效果差不多)

稀疏向量 vs 密集向量

- Tf-idf and PPMI兩種向量 V 都是
  - 長、稀疏向量 (|V|= 20,000 to 50,000)—分量大都值為 0
- 需要替代方案的向量
  - 短、密集向量 (|V|= 50-1000) —分量大都為非0)
- 為何需要短而密集的向量?
  - 在機器學習中是比較好的特徵值 (較少權重需要調整)
  - 比較更有一般性， 能掌握同義詞
    - 用詞當維度 car 和 automobile 是同義詞，但是代表不同維度，兩者的鄰居應該相似，但向量卻不相似
- 實驗證實短而密集的向量效果較佳

有現成的「詞內嵌」可以下載

- Word2vec (Mikolov et al.) https://code.google.com/archive/p/word2vec/
- Fasttext http://www.fasttext.cc/
- Glove (Pennington, Socher, Manning) http://nlp.stanford.edu/projects/glove/

簡介 word2vec

- 最普及、知名的詞內嵌方法
- 訓練時間短
- 程式碼公開
- 想法：**predict** rather than **count**
  - 不去數詞 w 在 “apricot" 旁邊出現幾次
  - 而是訓練二元分類器
    - 預測 w 是否容易出現在 "apricot"旁邊
  - 最終，並非要用這個分類器
  - 而是取分類器內的權重，作為「詞內嵌」

自我學習 Self Training—文章是預測下個字的訓練資料，不需要標註

- 靠近 *apricot* 在文章內有某一個鄰近詞 *s*
  - *s* 就是「w 是否常出現在 *apricot* 旁邊？」的標準答案
  - 不需要人工標註資料
- 觀念源自類神經網路，來取代次數、機率為本的語言模型
  - Bengio et al. (2003)
  - Collobert et al. (2011)

skip gram + 假性負面資料樣本 SGNS

- Word2vec 提供更簡單作法 "Skip-Gram +Negative Sampling”
  - 把 (目標詞, 鄰近詞) 當做正面資料實例
  - 隨機選取詞，把 (目標詞, 隨機詞) 當做負面資料實例
  - 使用迴歸分析來訓練二元分類器，區分兩者
  - 用分類器權重作為「詞內嵌」

Skip-Gram 訓練資料

訓練資料的句子：

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 target c3 c4

假設上下文為 +/- 2 詞的範圍 c1 c2 c3 c4

Skip-Gram 方法的目標

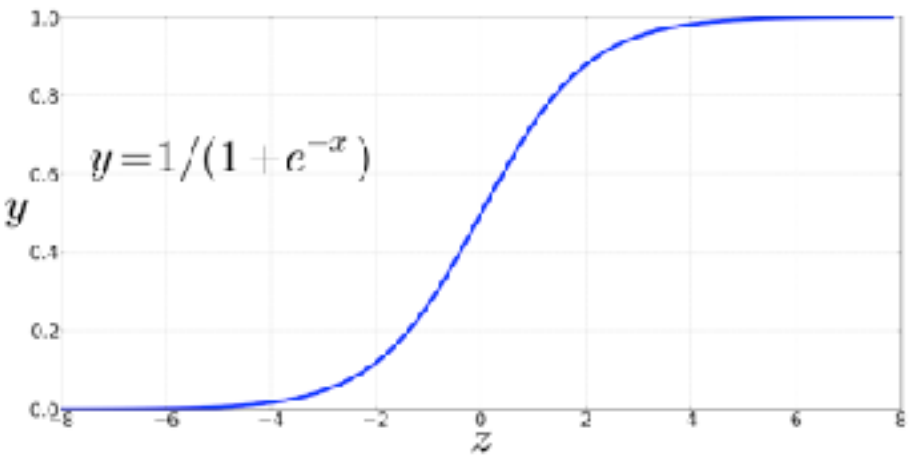
- 給予 (t, c) t = target, c = context 例如
  - (t, c) = (*apricot*, *jam*)
  - (t, c) = (*apricot*, *aardvark*)
- 訓練一個數學模型 $P(+|t,c)$
  - $P(+|apricot, jam) = 0.8$
  - $P(+|apricot, aardvark) = 0.001$
- $P(-|t,c) = 1 - P(+|t,c)$

如何計算 $P(+|t, c)$？

- 直覺
  - 詞容易出現在「相似詞」的附近
  - 「相似度」即向量內積 dot-product—Similarity(t,c) $\propto t \cdot c$
- 問題
  - 向量內積值的範圍為 [0, ∞] 不是機率的範圍 [0, 1]
- 解答
  - 代入sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$

$$P(-|t,c) = 1 - P(+|t,c) = \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}}$$



考慮所有的鄰近詞

假設所有鄰近詞互相獨立（機率相乘）

$$P(+|t, c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1 + e^{-t \cdot c_i}}$$

$$\log P(+|t, c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1 + e^{-t \cdot c_i}}$$

Skip-Gram 訓練（兩倍負面資料）

Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

k=2

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

如何隨機選擇雜訊詞 noise words

- 可以依照單詞機率 *P*(*w*) 的比例選取
- 但是，一般會在調整 *P*(*w*) 成為 $P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$
- α= .75 效果比較好，因為罕見詞的機率稍微提高
- 例如 (0.99, 0.01) -> (0.97, 0.03)

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

word2vec 演算法

- 隨機初始化所有 V 個詞彙 *w* 的向量為定長 (300) 向量
  - 開始有 300 * V 個隨機參數
- 訓練過程，我們會調整這些詞彙向量
- 最大化正面資料內目標詞, 鄰近詞配對 (*t*, *c*) 的相似度
- 最小化負面資料內目標詞, 鄰近詞配對 (*t*, *c*) 的相似度

word2vec 演算法：目標函數

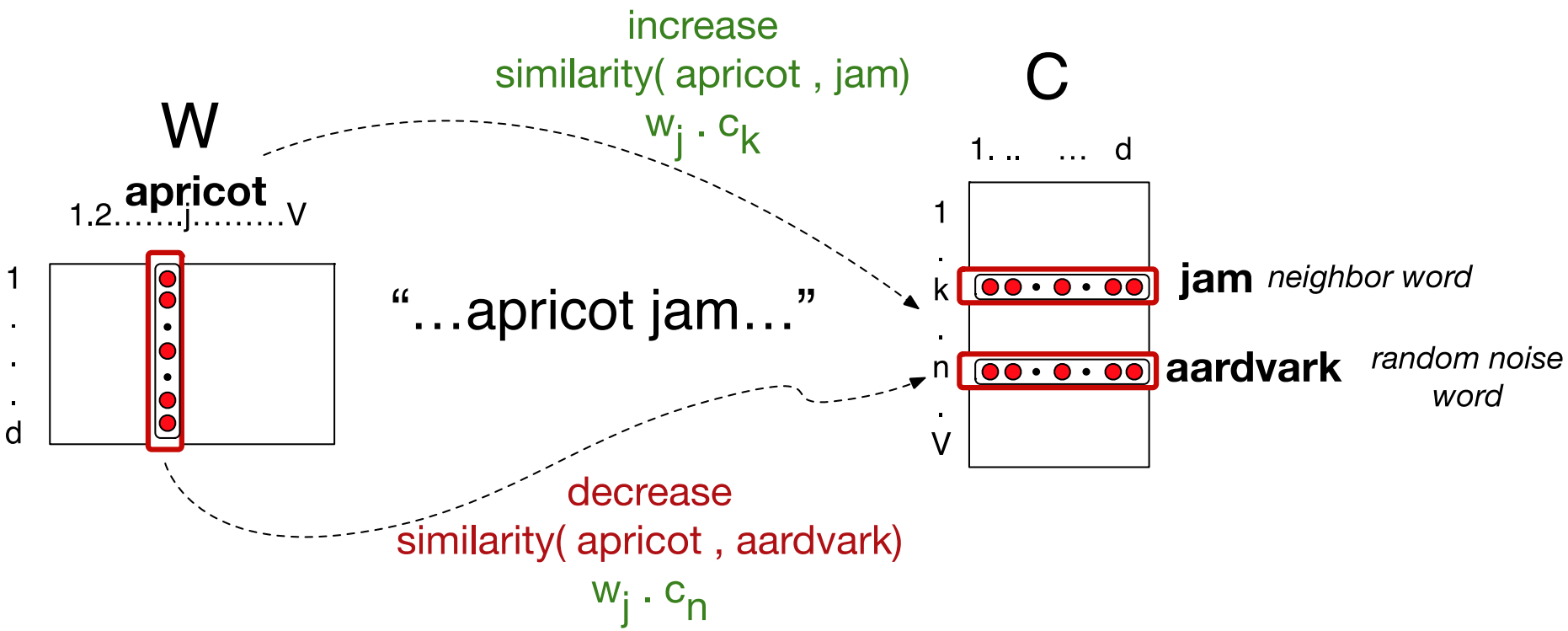
- 我們要最佳化以下的目標函數

$$L(\theta) = \sum_{(t,c)\in +} \log P(+|t,c) + \sum_{(t,c)\in -} \log P(-|t,c)$$

- 針對 + 號標籤的 (*t*, *c*) 配對和 - 號標籤的 (*t*, *c*) 來最大化
- 若針對一組配對 (*t*, *c*) 與其 *k* 個負面資料 n_i，則最大化

$$\begin{aligned} L(\theta) &= \log P(+|t,c) + \sum_{i=1}^{k} \log P(-|t,n_i) \\ &= \log \sigma(c \cdot t) + \sum_{i=1}^{k} \log \sigma(-n_i \cdot t) \\ &= \log \frac{1}{1+e^{-c \cdot t}} + \sum_{i=1}^{k} \log \frac{1}{1+e^{n_i \cdot t}} \end{aligned}$$

word2vec 演算法



用梯度下降 gradient descent 最大化目標

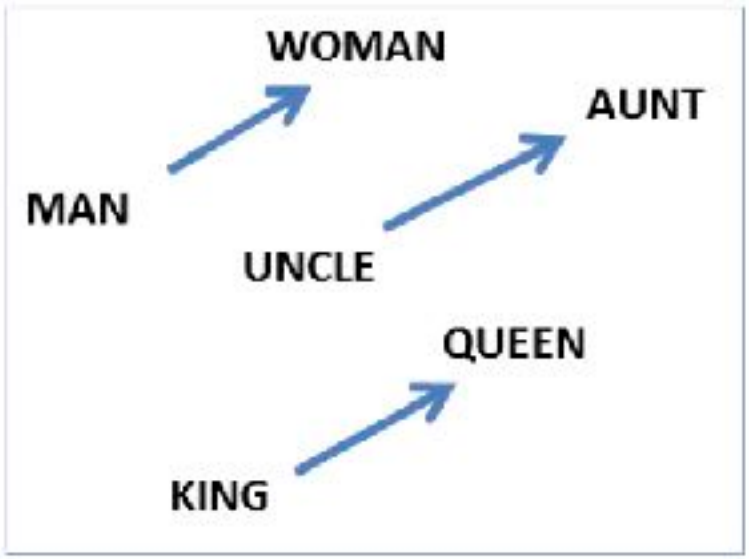
- 實際上，學習兩組矩陣 W 和 C
- W 代表目標詞的所有向量
- C 代表鄰近詞的所有向量
- 訓練結束之後
  - 留下 W 丟掉 C
  - 使用合併的 W 和 C

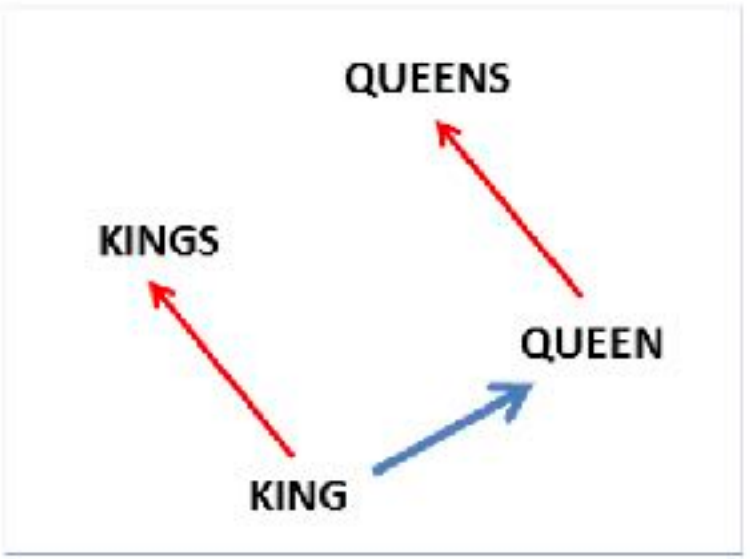
小結：word2vec (skip-gram) 詞內嵌

- 由 V 個隨機產生的300-維向量開始
- 用迴歸分析（最基本的機器學習法）
  - 取輸入語料庫的共同出現的兩個詞作為正面資料
  - 取不曾共同出現的兩個詞作為負面資料
  - 透過慢慢調整向量值訓練一個區分兩者的分類器
- 丟掉分類器，留下詞向量（分類器特徵值）

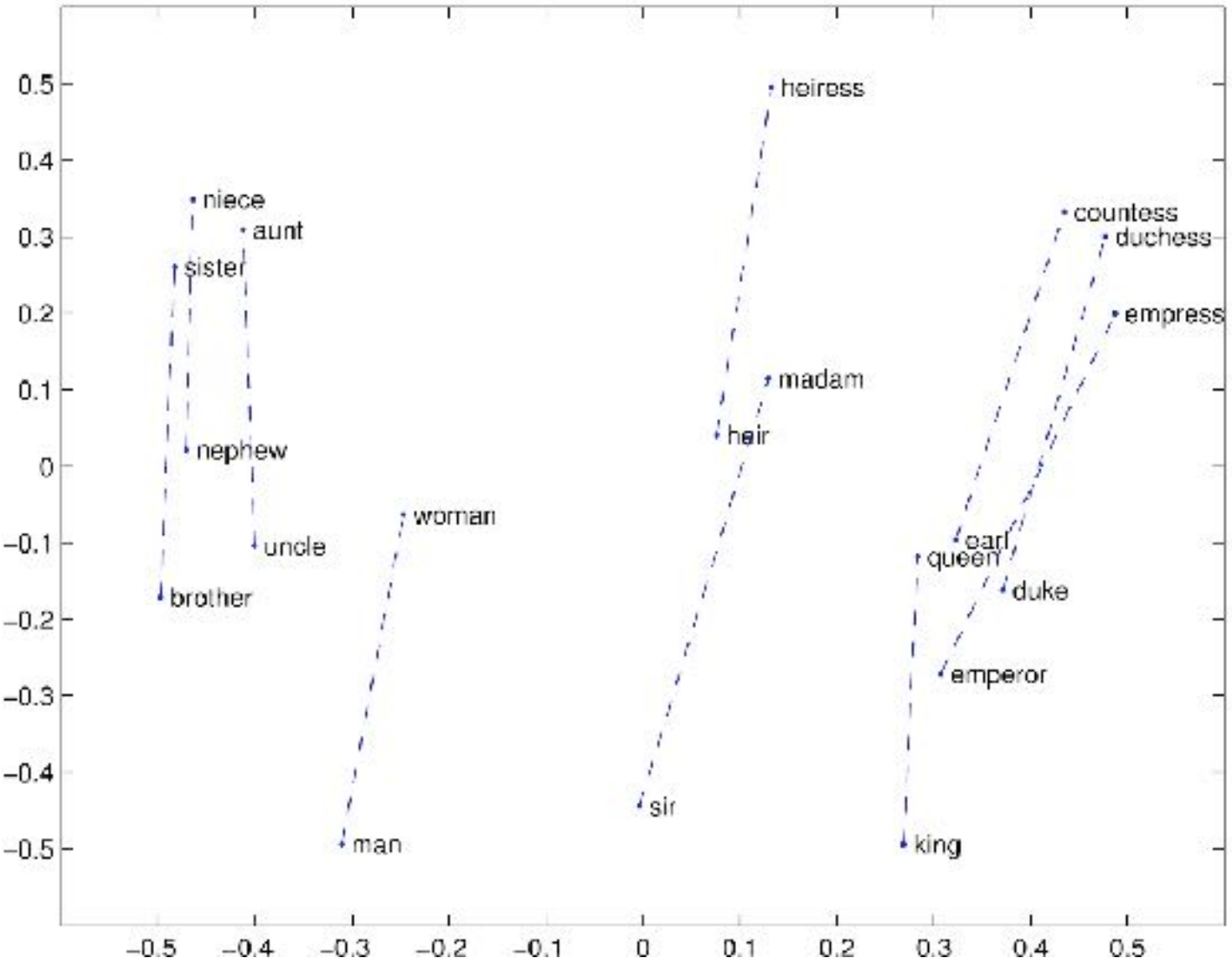
詞內嵌掌握了詞意關係：類比

- vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) vector('queen')
- vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) vector('Rome')

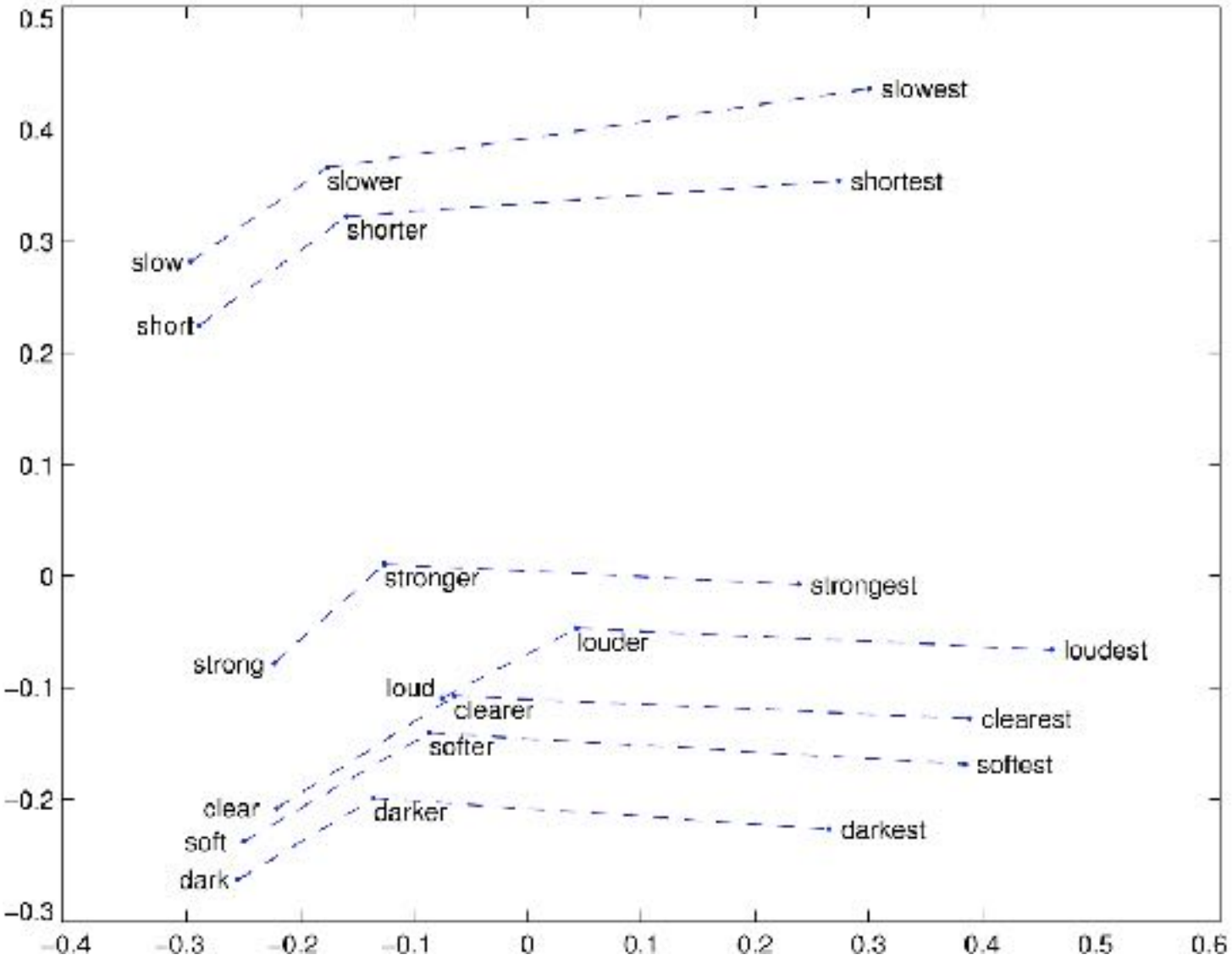




詞內嵌掌握了詞意關係：性別



關係：形容詞比較、最高級


slowest
slower
shortest
shorter
slow
short
stronger
strongest
louder
loudest
strong
loud
clearer
clearest
softer
softest
clear
darker
darkest
soft
dark
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
0.6

如何視覺化高維度向量空間?

- T-Distributed Stochastic Neighbouring Entities (t-SNE)
  - Visualizing Data Using t-SNE (GoogleTechTalks) June 24, 2013—https://www.youtube.com/watch?v=RJVL80Gg3lA&list=UUtXKDgv1AVoG88PLl8nGXmw
  - https://lvdmaaten.github.io/tsne/
- 動手做
  - Introduction to t-SNE (https://www.datacamp.com/community/tutorials/introduction-t-sne)
- 論文
  - http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

t-SNE 概念

- 非線性維度降低的統計模型
  - 讓相似點靠近機率值高
  - 讓相異點靠近機率值低
- 演算法

1. 定義挑選點配對的機率分布
   - 相似點被挑中的機率高
   - 相似點被挑中的機率低
2. 定義高、低維度分布
   - 最小化高維度、低維度分布之 KL divergence

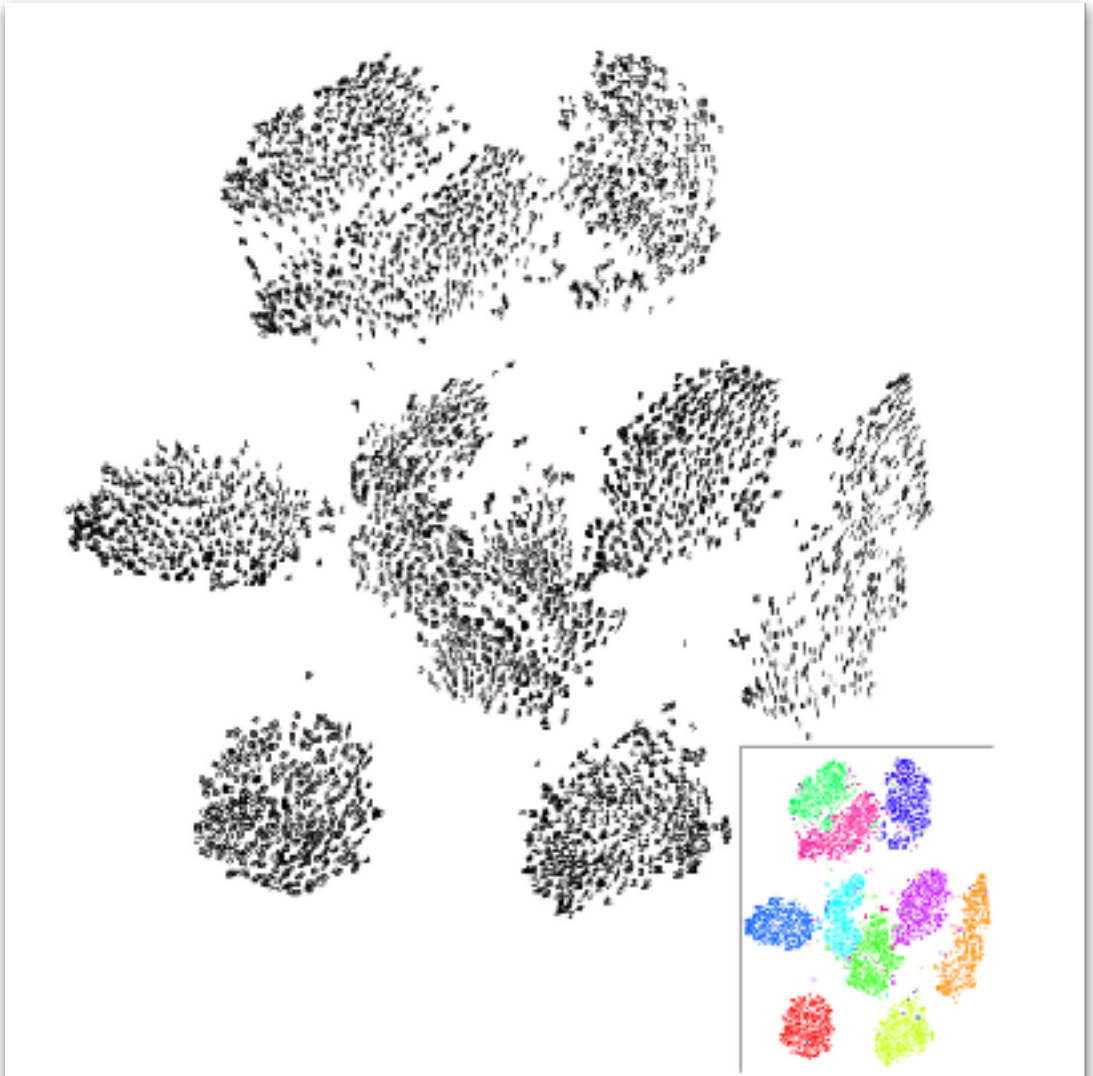


Figure 7: Visualization of 6,000 digits from the MNIST data set produced by the random walk version of t-SNE (employing all 60,000 digit images).