

Network Embedding

Word2Vec Model

Recap the Word2Vec Model

單詞 = 向量

語義相近的單詞，在向量空間的距離相近
利用 **cosine similarity** 計算兩個單詞的關聯性

透過上下文學習 單詞的語義向量

擁有相近上下文的單詞，具有相近的語意

Word2Vec 原理

將詞彙做 **one-hot encoding**，再透過 **word2vec** 類神經網路計算，求出降維後的語義向量

實際上

Word Embedding

Word2Vec 與 auto-encoder 類似

無監督特徵學習 (Unsupervised Feature Learning)

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

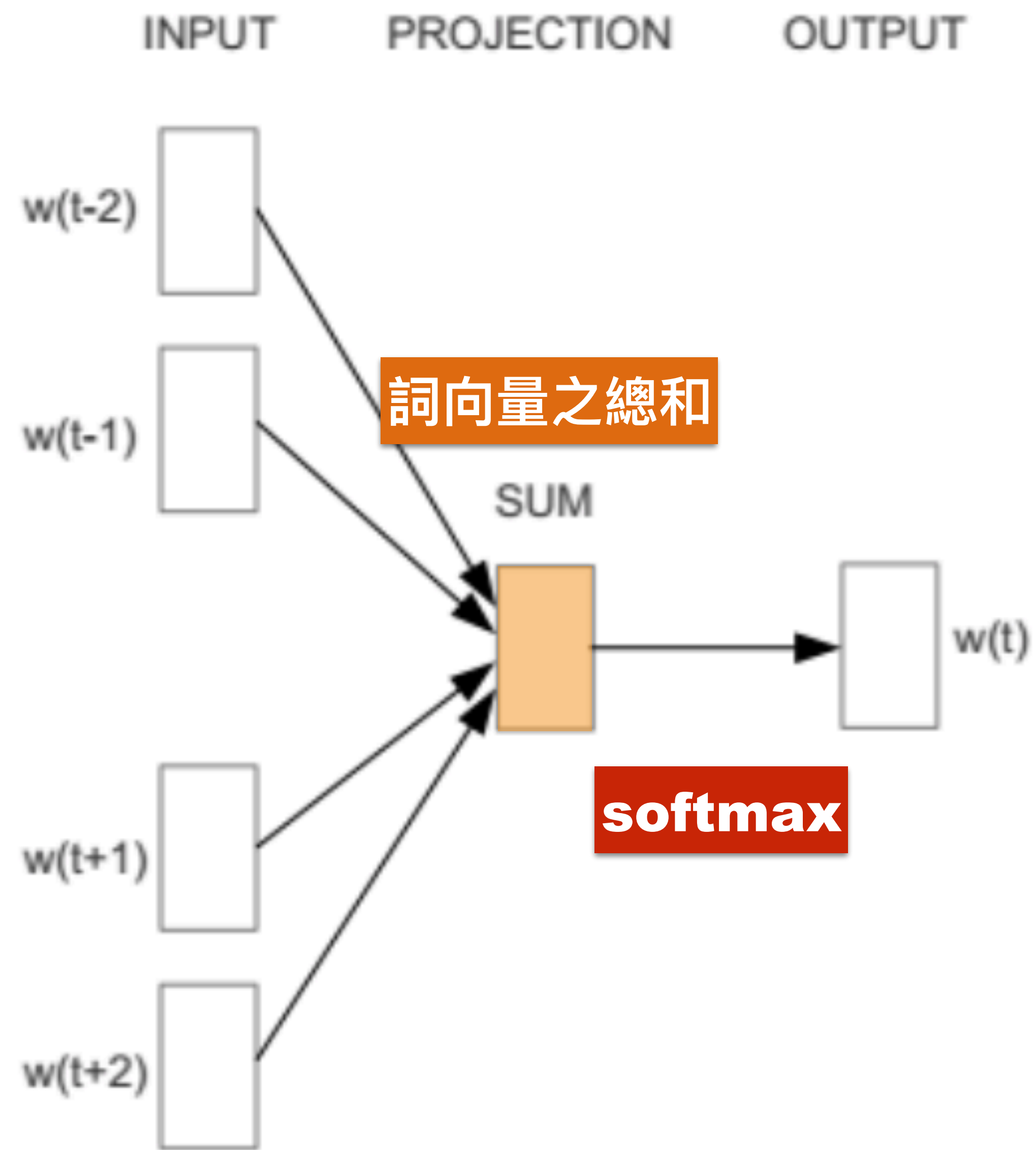
CBOW: $P(\text{word} \mid \text{context})$

輸入 **context word** ➡ 訓練 **target word**

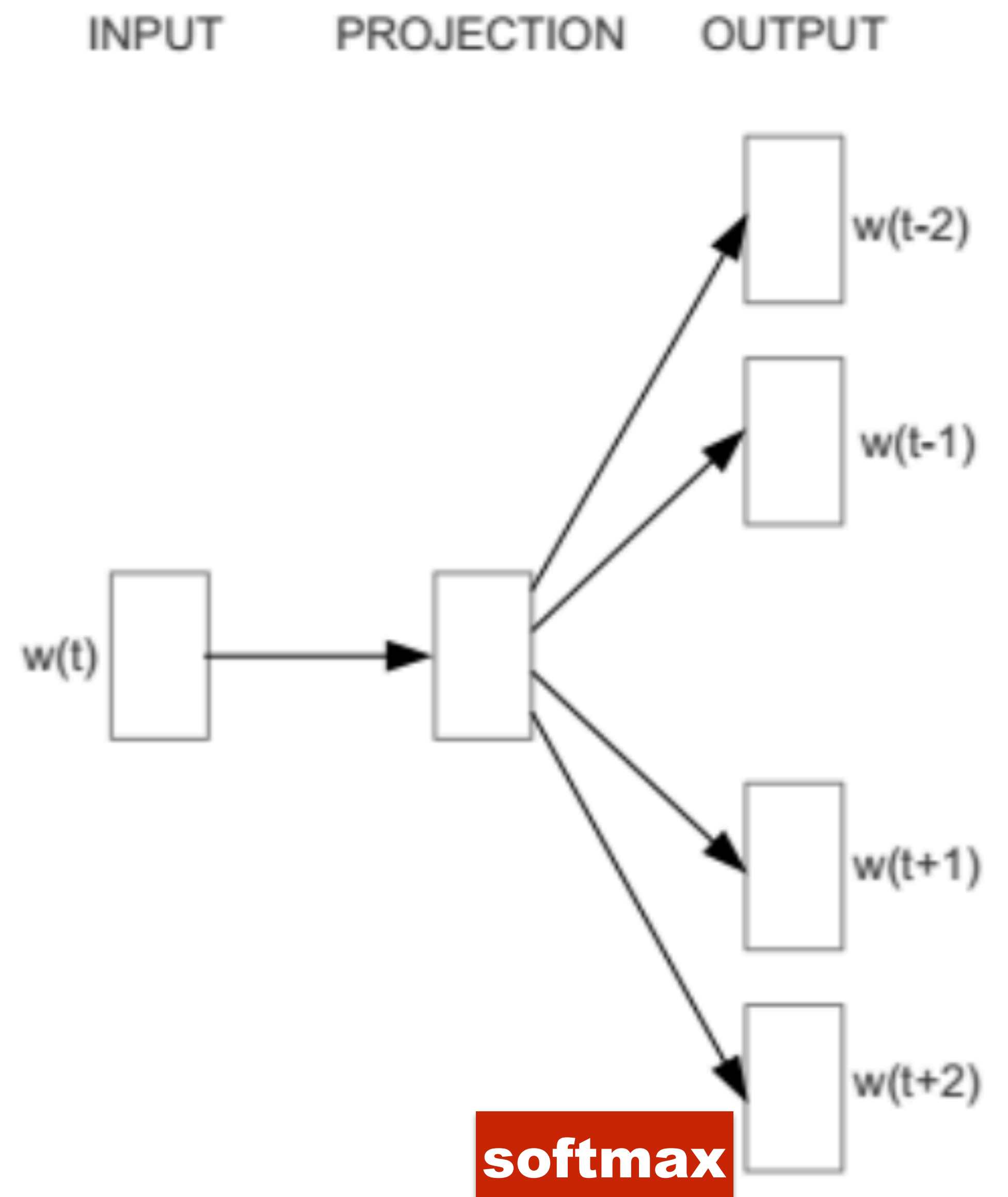


skip-gram: $P(\text{context} \mid \text{word})$

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 Model

基於成對的單詞 (**input word, output word**) 來對類神經網絡進行訓練。
input word 和 **output word** 都是 **one-hot** 編碼的向量。
最終模型的輸出是一個機率分佈。

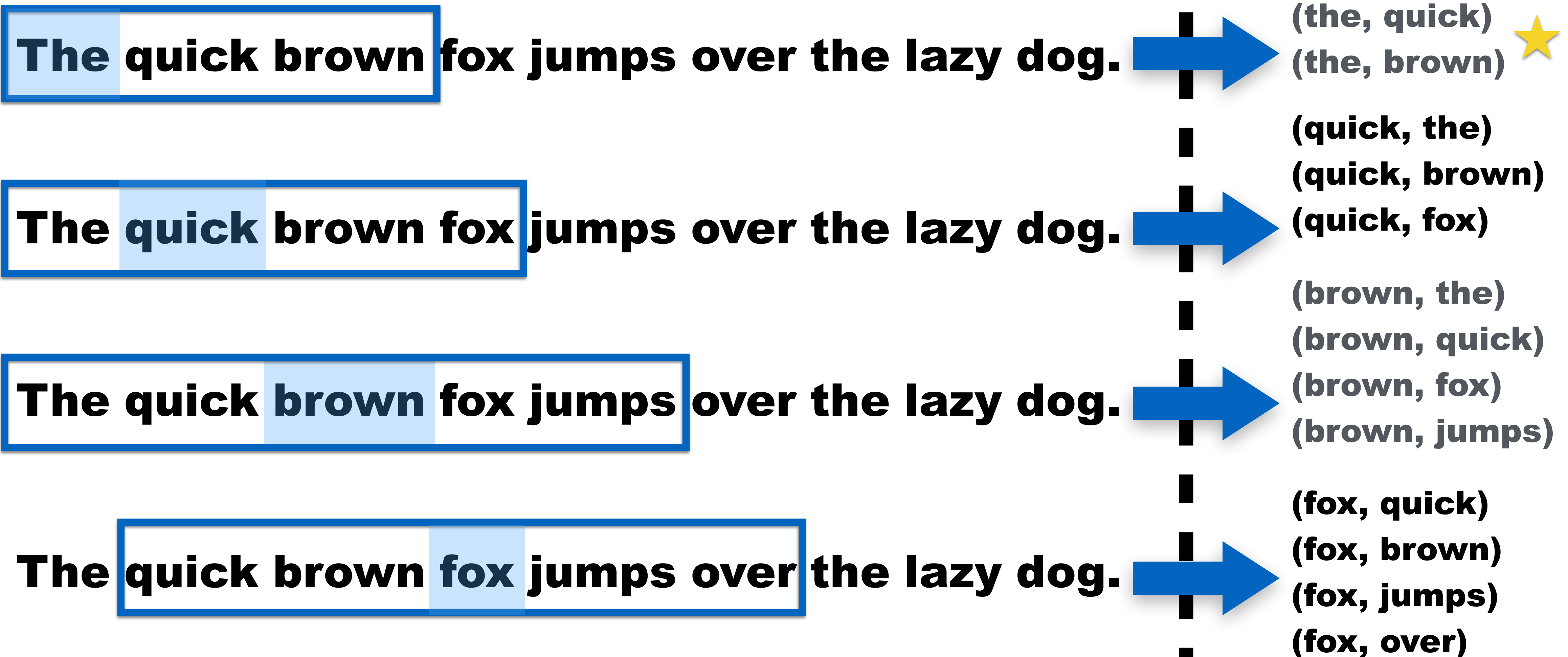
SourceText

Training Samples

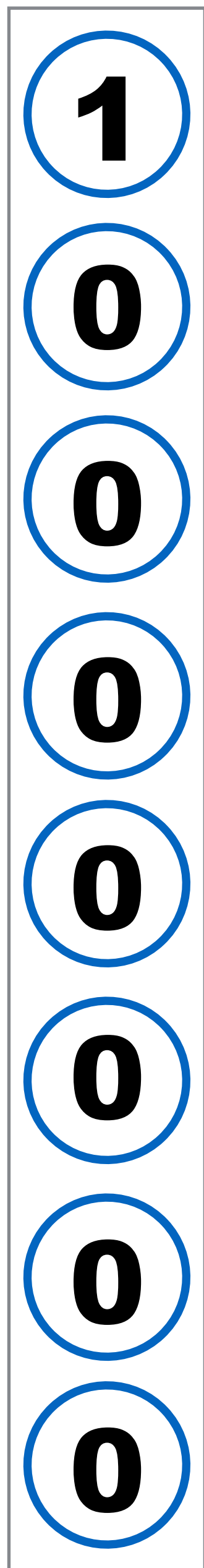
window size=2, 僅選輸入詞前後各兩個詞和輸入詞進行組合

藍色代表 input word (target word)

(input word, output word) 形式的訓練數據

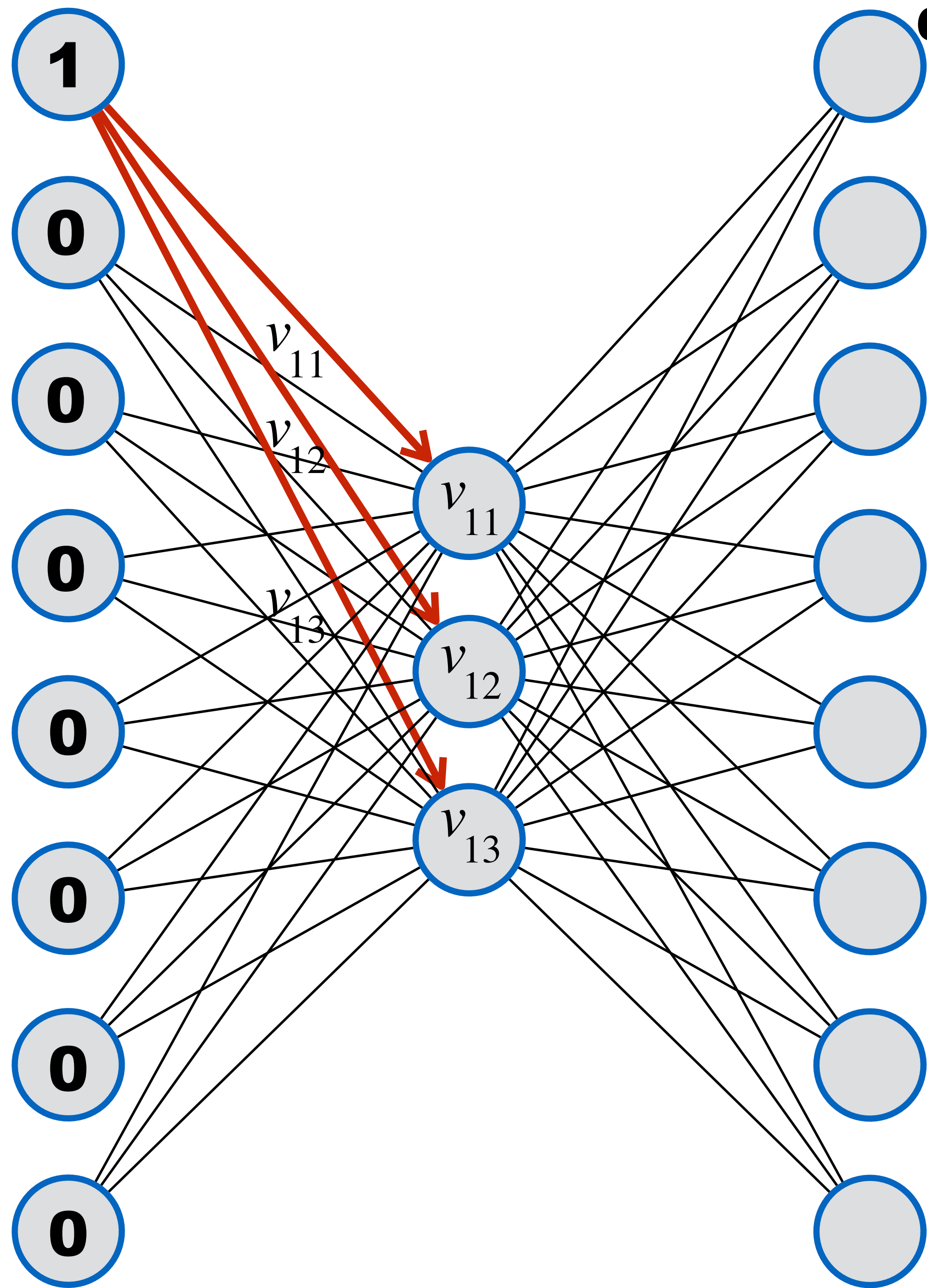
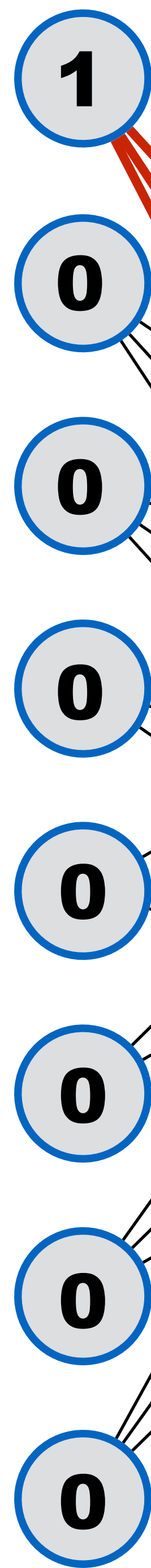


the



高維度

the



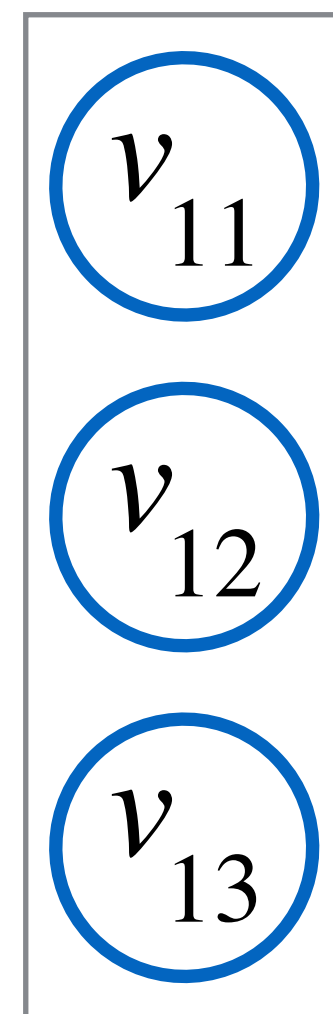
quick

brown

0
1
0
0
0
0
0
0
0
0

0
0
1
0
0
0
0
0
0
0

the
語義向量



低維度



output weights for “quick”

word vector for “the”



×

8 dimensions



softmax



=

Probability that if you randomly pick a word nearby “the”, that it is “quick”

one-hot encoding dimensions
(8 dimensions)

[1 0 0 0 0 0 0 0]

高維度

v_{11}	v_{12}	v_{13}
v_{21}	v_{22}	v_{23}
v_{31}	v_{32}	v_{33}
v_{41}	v_{42}	v_{43}
v_{51}	v_{52}	v_{53}
v_{61}	v_{62}	v_{63}
v_{71}	v_{72}	v_{73}
v_{81}	v_{82}	v_{83}

= [v_{11} v_{12} v_{13}]

低維度
語義向量

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.

透過上下文

Word Embedding



透過網絡結構的鄰居關係

Network Embedding

Network Representation Learning

概念: Network Embedding

graph representation

node representation

node embedding

graph embedding

將網絡節點投影到 低維向量空間，

用於 **node classification**、**link prediction**、**community detection**、**visualization** 等應用場景

算法分類

1. 基於矩陣特徵向量計算

2. 基於 **Random Walk** 框架計算

- **DeepWalk**
- **Node2Vec**

3. 基於 **Deep Learning** 框架計算

4....

DeepWalk

Random Walk + Word2Vec

隨機遊走序列，
涵蓋節點的順序性、鄰居關係
(**context neighborhood nodes**)

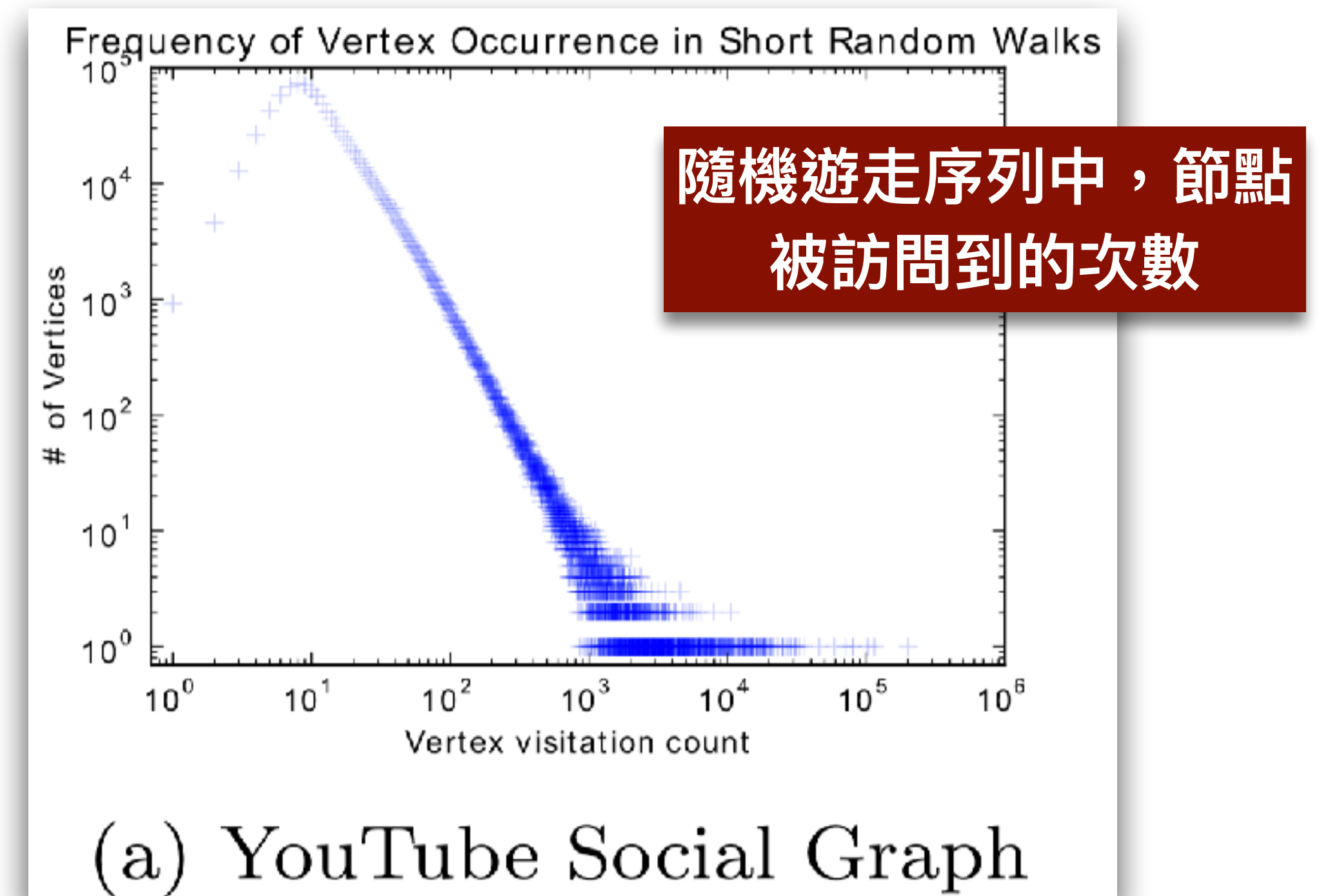
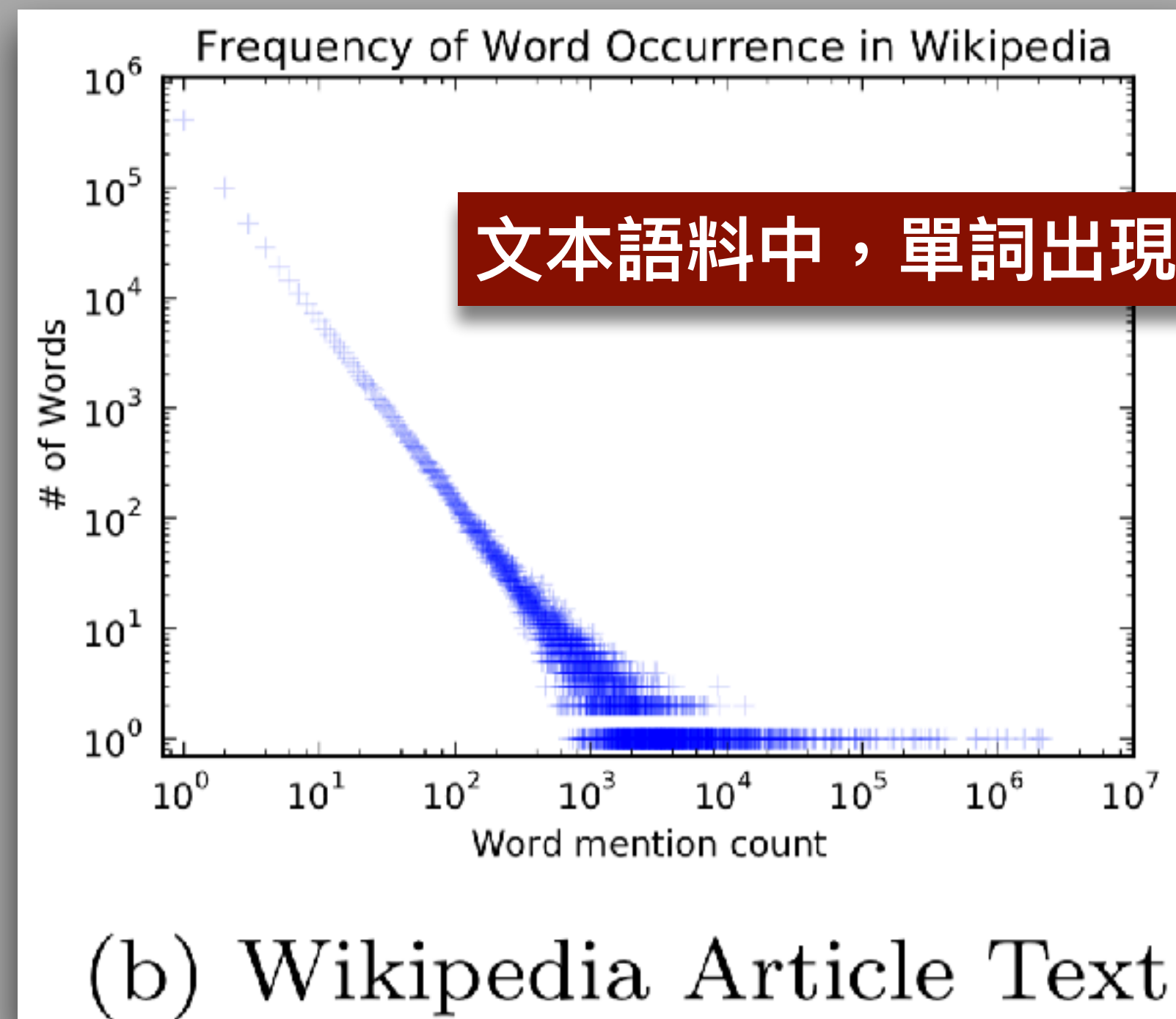
Skip-gram
Hierarchical Softmax

<https://github.com/phanein/deepwalk>

NLP: Word2Vec

SNA: DeepWalk

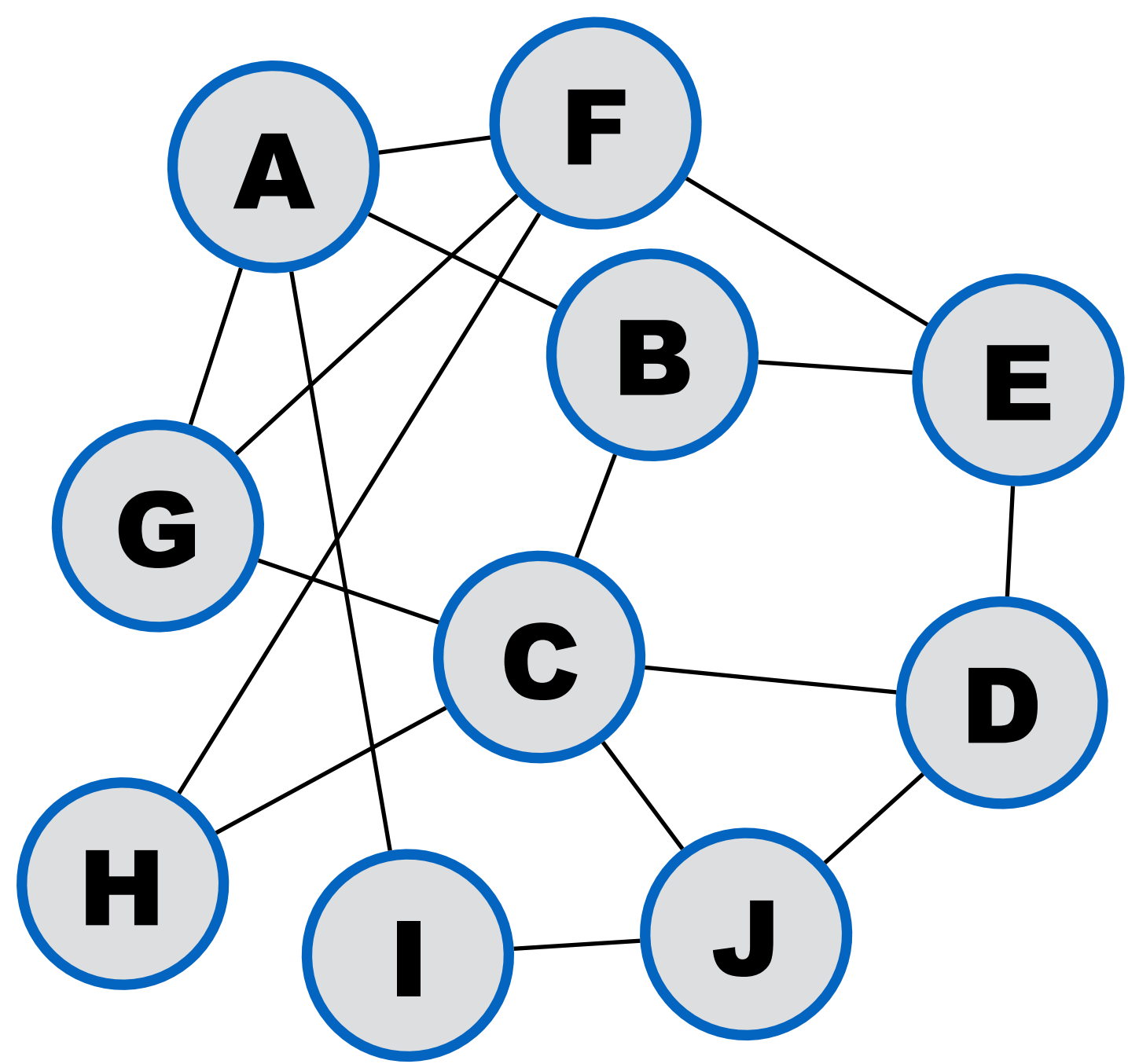
透過隨機遊走選取網絡節點，並生成**特定長度**的隨機遊走序列，將其類比為自然語言中的句子 (**節點=單詞**，**節點序列=句子**)，透過 **Word2Vec** 更新參數



網絡結構中的節點和文本中的單詞具有可類比性 **Zipf's Law**

有意義的單詞(佔多數)出現頻率低，停用詞(佔少數)的出現頻率極高。

Random Walk 策略




Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

```
1: Initialization: Sample  $\Phi$  from  $\mathcal{U}^{|V| \times d}$ 
2: Build a binary Tree  $T$  from  $V$ 
3: for  $i = 0$  to  $\gamma$  do
4:    $\mathcal{O} = \text{Shuffle}(V)$ 
5:   for each  $v_i \in \mathcal{O}$  do
6:      $\mathcal{W}_{v_i} = \text{RandomWalk}(G, v_i, t)$ 
7:      $\text{SkipGram}(\Phi, \mathcal{W}_{v_i}, w)$ 
8:   end for
9: end for
```

- 1. 遍歷網絡的每個節點，且節點可被重複訪問
- 2. 隨機採樣 1 個節點，以此為起點生成特定長度的隨機遊走序列
- 3. 遍歷整個網絡 n 輪
- 4. 原始代碼不支持帶有權重的網絡結構



window size=2

context	context	target	context	context
A	B	C	D	E
B	E	F	G	C
C	H	F	A	I
D	C	G	F	E

C	>>	A
C	>>	B
C	>>	D
C	>>	E

Skip-gram

Node2Vec

Random Walk + Word2Vec

改善隨機遊走的策略

**Skip-gram
Negative Sampling**

<http://snap.stanford.edu/node2vec/>, <https://github.com/aditya-grover/node2vec>

Random Walk 策略: 極端情形

There are two extreme sampling strategies for generating neighborhood set of k nodes:

找相鄰節點

著重局部資訊

著重全局資訊

遠離起始點

Breadth-first Sampling

廣度優先

Depth-first Sampling

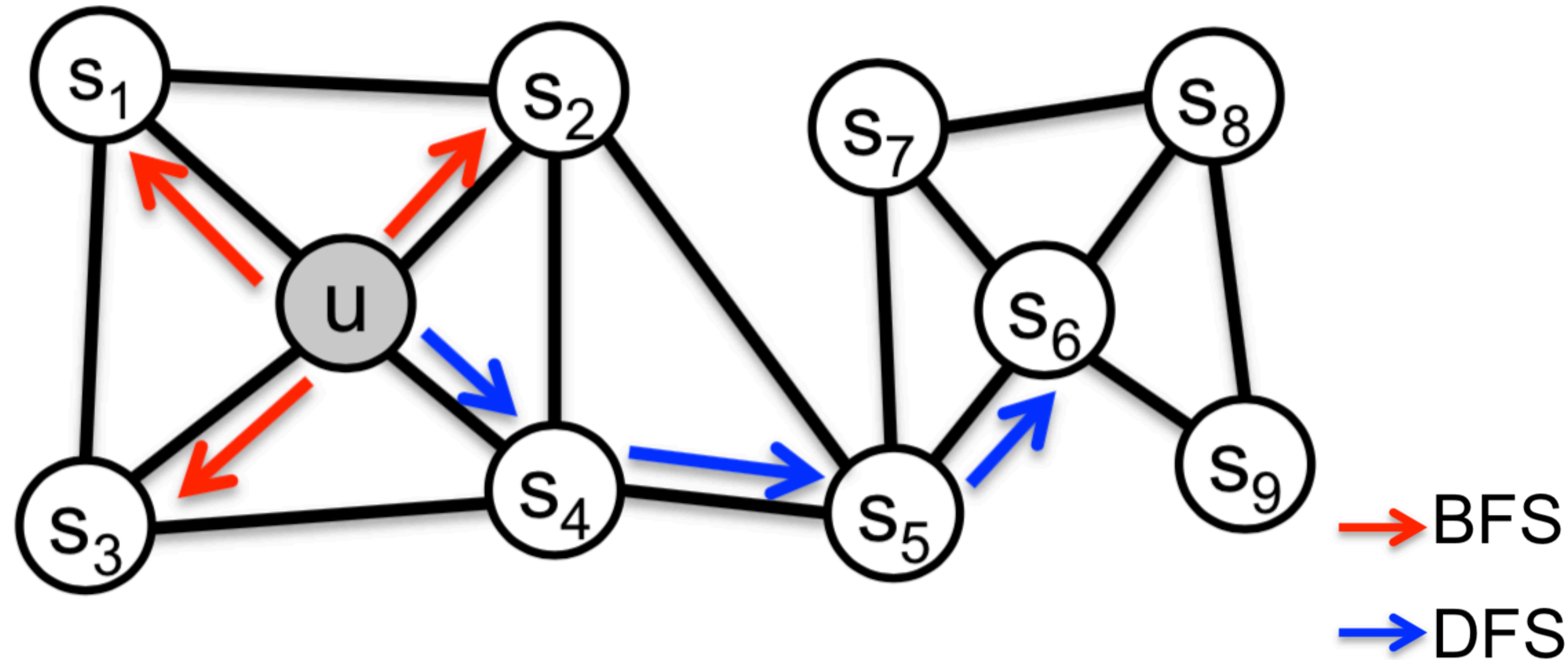


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

在 **BFS, DFS** 中取得平衡，同時考慮到局部和宏觀的資訊

Random Walk 策略: 二階隨機遊走

對當前節點的鄰域節點做 **sampling**，同時保留該節點在網絡中的位置資訊

節點 $v \rightarrow x$ 轉移機率

$v \rightarrow x$ 邊的權重

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

節點 t, x 之間的最短路徑

透過 p, q 兩個模型參數，調控隨機遊走的策略
當 $p=q=1$ 時，等同於 **DeepWalk**

將 v 的鄰居節點區分成三個類型

1. 與上個節點 t 相距 0
2. 與上個節點 t 相距 1
3. 與上個節點 t 相距 2

透過 p, q 形成一個不均勻的概率分佈，最終得到隨機遊走的路徑

根據網絡結構的不同特點，可以調整 p, q 來實現對不同 **sampling** 方式的偏好

Random Walk 策略: 二階隨機遊走

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



現在從節點 **t** 遊走到節點 **v**

首先計算其鄰居節點(下個候選節點)與上一節點 **t** 的距離 **d**

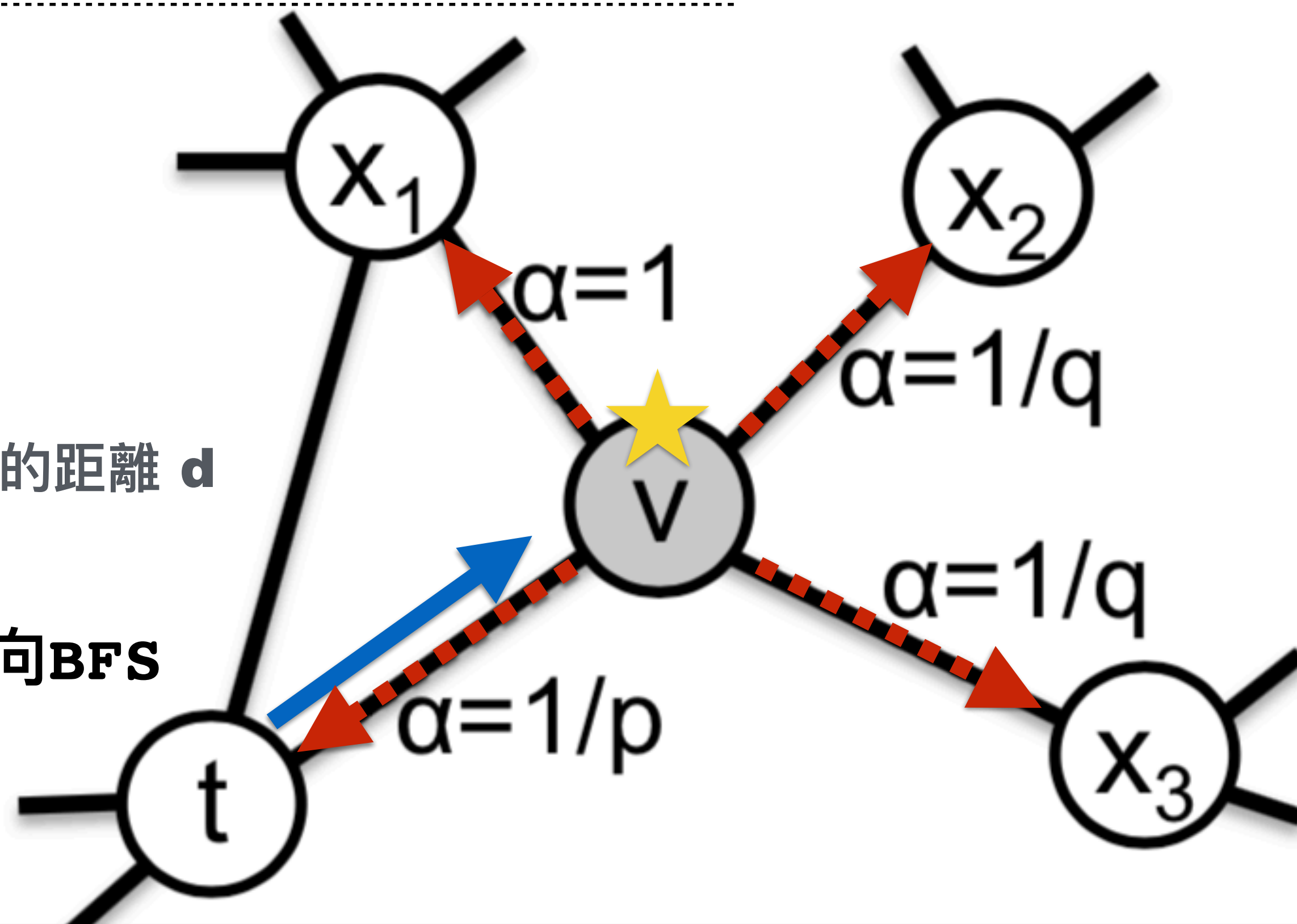
從而得到 **alpha**

t >> **v** >> **t** 立即重複拜訪相同節點, 傾向BFS

t >> **v** >> **x1** 傾向DFS, BFS

t >> **v** >> **x2** 傾向DFS

t >> **v** >> **x3** 傾向DFS



1. 返回機率參數(Return parameter) **p**, 控制回到原始節點的機率
 - 設定 $p > \max(q, 1)$, 避免立即重複拜訪相同節點
 - 設定 $p < \min(q, 1)$, 傾向立即重複拜訪相同節點
2. 離開機率參數(In-out parameter) **q**, 控制跳到其他節點的機率
 - 設定 $q > 1$, 傾向 **BFS**, 在原始節點 **t** 附近繞, 著重局部資訊
 - 設定 $q < 1$, 傾向 **DFS**, 遠離原始節點 **t**

Node2Vec

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d , Walks per node r , Walk length l , Context size k , Return p , In-out q)
 $\pi = \text{PreprocessModifiedWeights}(G, p, q)$
 $G' = (V, E, \pi)$
 Initialize $walks$ to Empty
 for $iter = 1$ **to** r **do**
 for all nodes $u \in V$ **do**
 $walk = \text{node2vecWalk}(G', u, l)$
 Append $walk$ to $walks$
 $f = \text{StochasticGradientDescent}(k, d, walks)$
 return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u , Length l)
 Initialize $walk$ to $[u]$
 for $walk_iter = 1$ **to** l **do**
 $curr = walk[-1]$
 $V_{curr} = \text{GetNeighbors}(curr, G')$
 $s = \text{AliasSample}(V_{curr}, \pi)$
 Append s to $walk$
 return $walk$

DeepWalk

DeepWalk 在下一個節點的選擇上，
只是對當前節點的所有鄰居節點中，
隨機挑選一個

Node2Vec

next_node 的選擇，需要 $\langle \text{pre_node}, \text{cur_node} \rangle$ 兩個節點的資訊：

- 初始的轉移機率空間複雜度，是 **DeepWalk** 的平方
- 當節點數較多時，可能出現 **Out of Memory**

二階隨機遊走，較複雜：

- 當網絡節點數較少時，**Node2Vec** 的效果整體優於 **DeepWalk**
- 但當節點數多到可能影響空間限制時，**DeepWalk** 效果更優

Generalization: Random Walk 策略

走在潮流尖端

ACM CIKM 2017

Conference on Information and Knowledge Management

22 篇 **Graph Mining**

11 篇 **NE**

171 篇長論文

22 篇 **Graph Mining**

<http://cikm2017.org>

主會場共設置 **49** 場專題 **Session**，**Graph Representation/Graph Mining** 佔其中的五分之一

新研究方向

1. 結合節點屬性、邊的權重、邊的屬性、高階網絡結構做
整體 **Embedding**
2. 動態網絡的增量 **Embedding**
3. 異質網絡的 **Embedding**
 - **meta-path/Multi-view/Attention**
4. 提高節點分類 / **link** 預測的成效

經典的 **DeepWalk**、**Node2Vec**、**LINE** 算法，主要基於淺層模型，而淺層模型往往收斂於局部最佳解，無法表示更複雜的非線性網絡結構

The End.