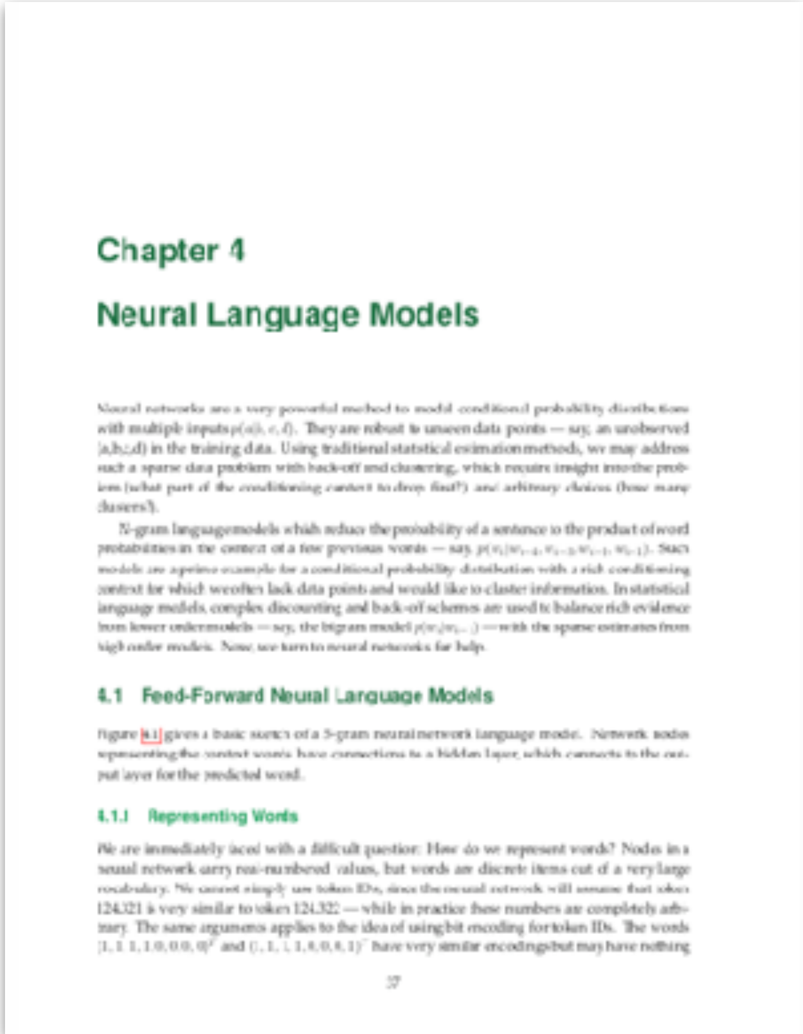
NMT 第 4 章 類神經網路語言模型

教科書與課程網站：mt-class.org/jhu/syllabus.html （草稿）

2018 0930

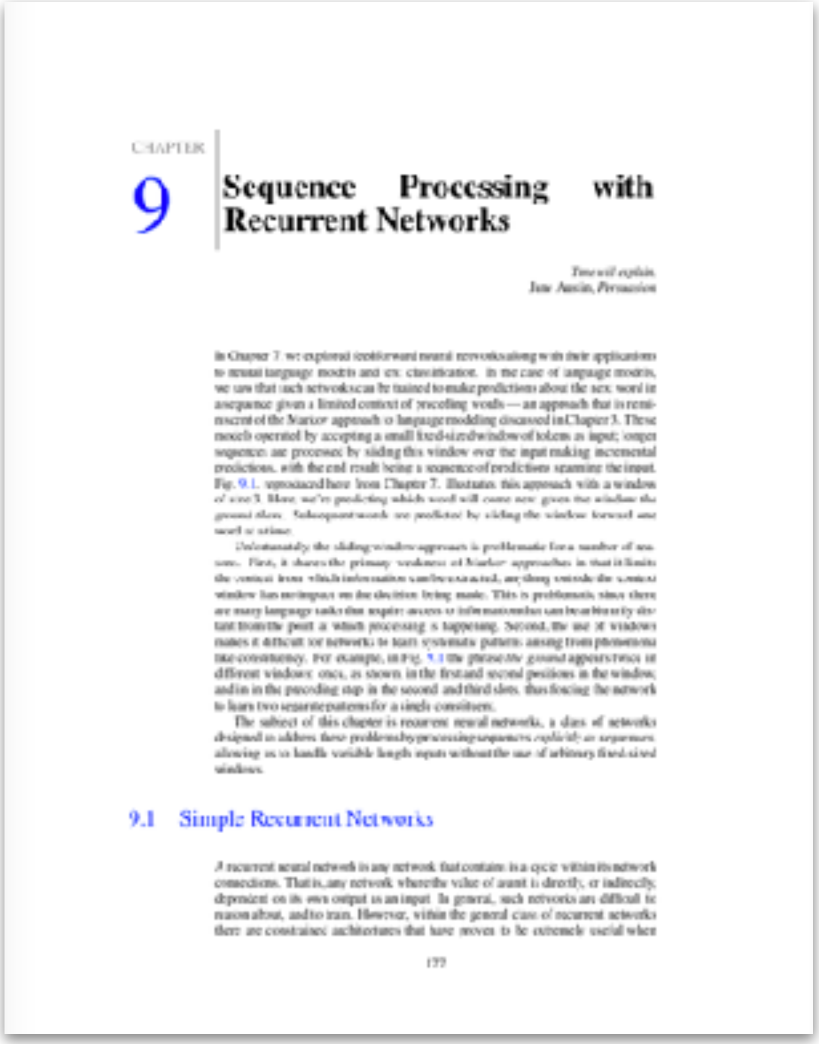
教科書相關章節


Chapter 4

Neural Language Models

4.1 Feed-Forward Neural Language Models

4.1.1 Representing Words


CHAPTER

9 Sequence Processing with Recurrent Networks

9.1 Simple Recurrent Networks

來源：mt-class.org/jhu/assets/nmt-book.pdf

web.stanford.edu/~jurafsky/slp3/９.pdf (Draft of September 23, 2018)

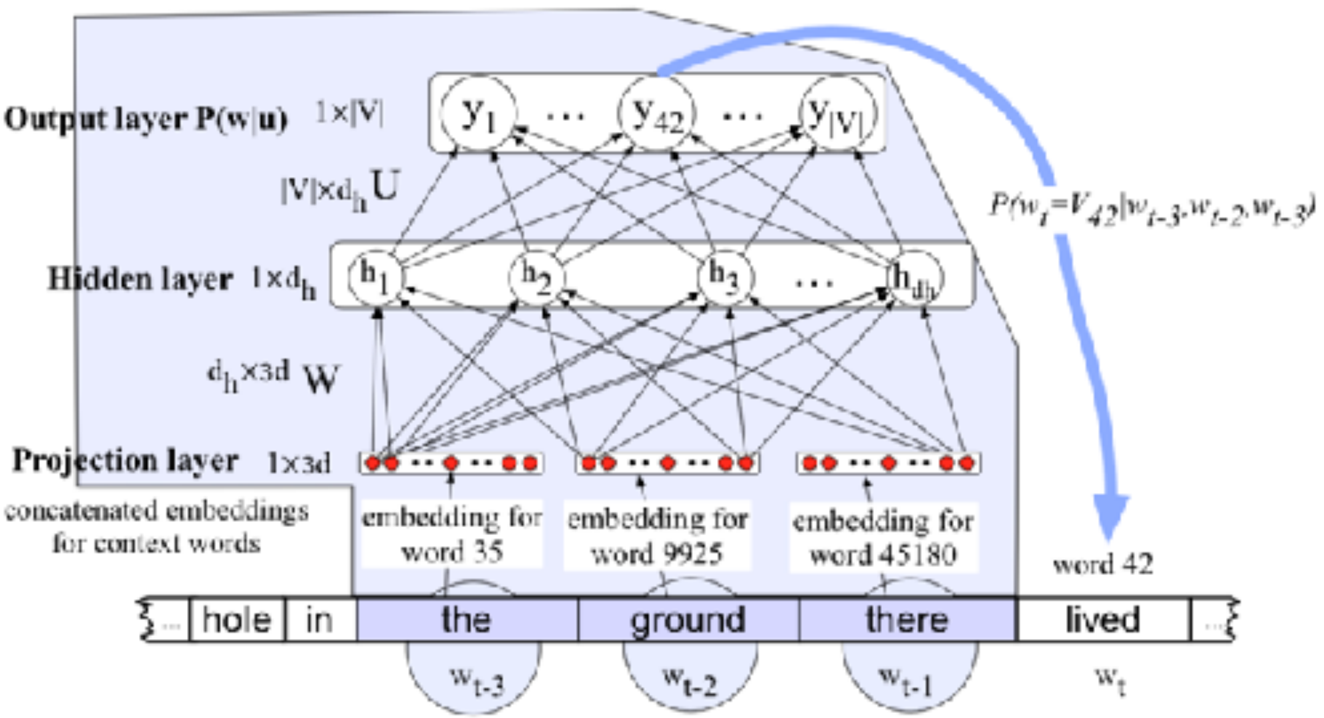
大綱

- 9.1 簡單遞迴網路 Simple Recurrent Networks
- 9.2 RNNs 應用
- 9.3 深度網路：堆疊和雙向 RNNs
- 9.4 在 RNNS 中管控文脈：LSTMs 和 GRUs
- 9.5 表達輸入：詞、字母、Byte-Pairs
- 9.6 結語

來源：http://www.cse.chalmers.se/~coquand/AUTOMATA/mcp.pdf

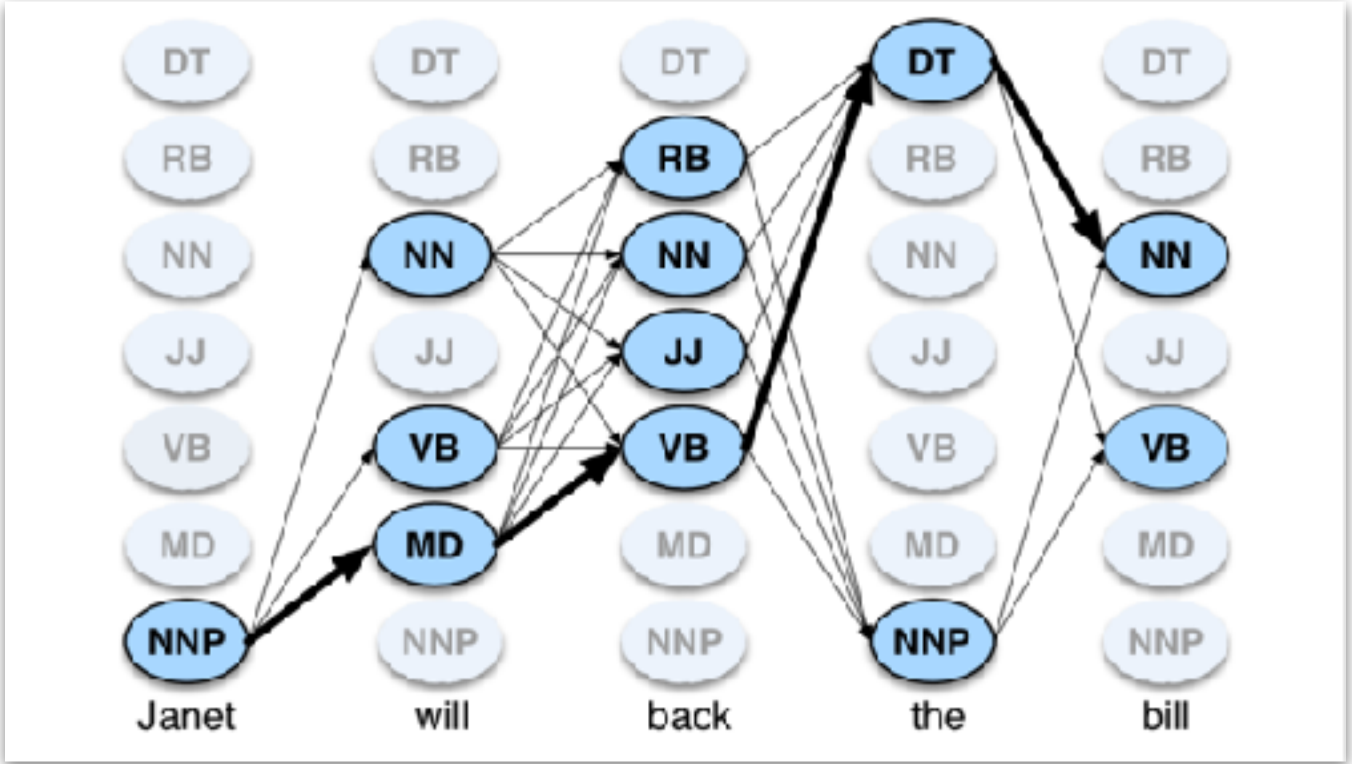
9 簡介

- HMM 和 FFNN 問題
  - 假設輸入為固定長度的詞 (Markov Approach)
  - 把輸入切割成幾段重疊的固定長度的片段
  - 逐次往前移動一段一段處理
- FFNN



9 簡介

- 假設輸入為固定長度的詞 (Markov Approach)
- 把輸入切割成幾段重疊的固定長度的片段
- 逐次往前移動一段一段處理
  - HMM

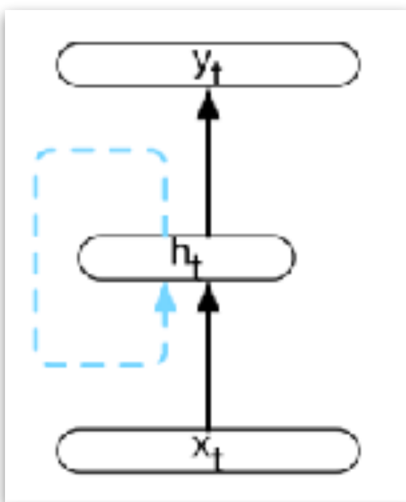


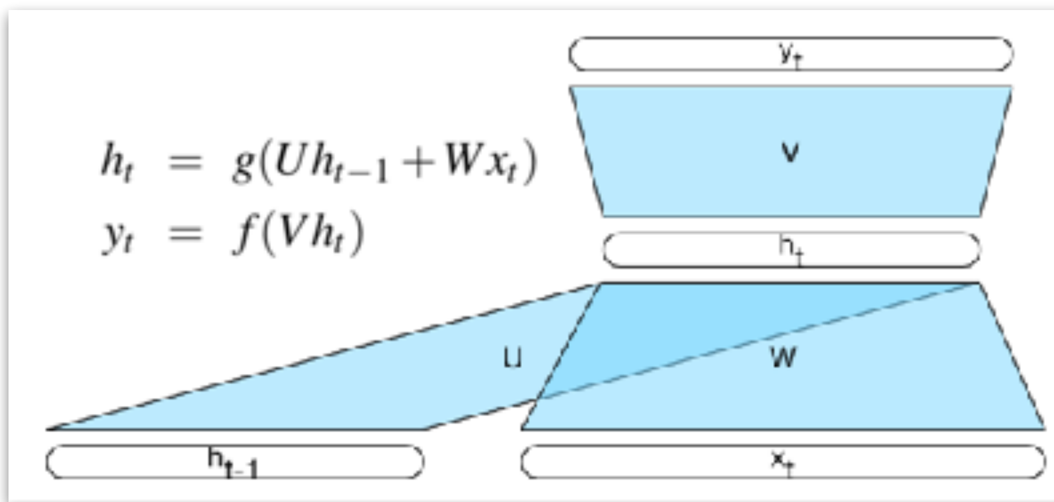
問題與解決方案

- 問題
  - 限制一次處理輸入的「視窗」——限制遠距離資訊
  - 無法處理語言的結構 (constituents)—結構通常是不固定長度
- 解決方案
  - 遞迴網路 Recurrent Networks
  - 設定輸入 ＝ 不定長的輸入 sequence
  - 輸入是一個整體，而非一段
  - 像是 recursive program (動態) 而非 for loop (固定次數)

9.1 簡單遞迴網路 Simple Recurrent Networks

- 和 FFNN 不同點
  - 輸入增加 $h_{t\text{-}1}$ 的隱藏層狀態 (就像是「歷史、記憶」)
  - 增加 U 來表示如何運用「歷史」來計算 h_t 和 y_t
- 和 FFNN 一樣　(雖然表面上很不一樣)
  - 在每個時間點 i ，用狀態 $h_{t\text{-}1}$ 和輸入 x_t，往前計算 h_t 輸出 y_t

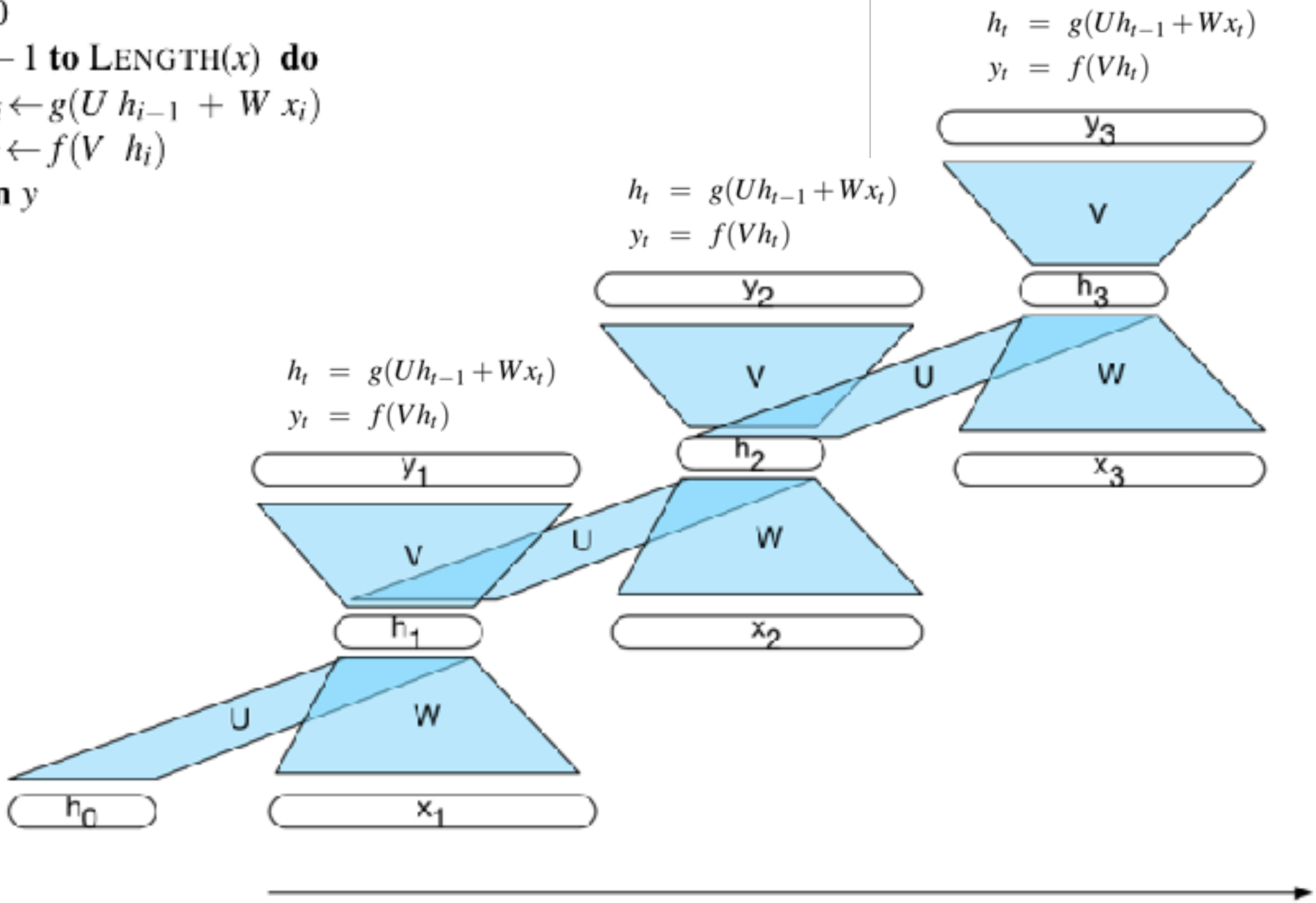




9.1.1 計算

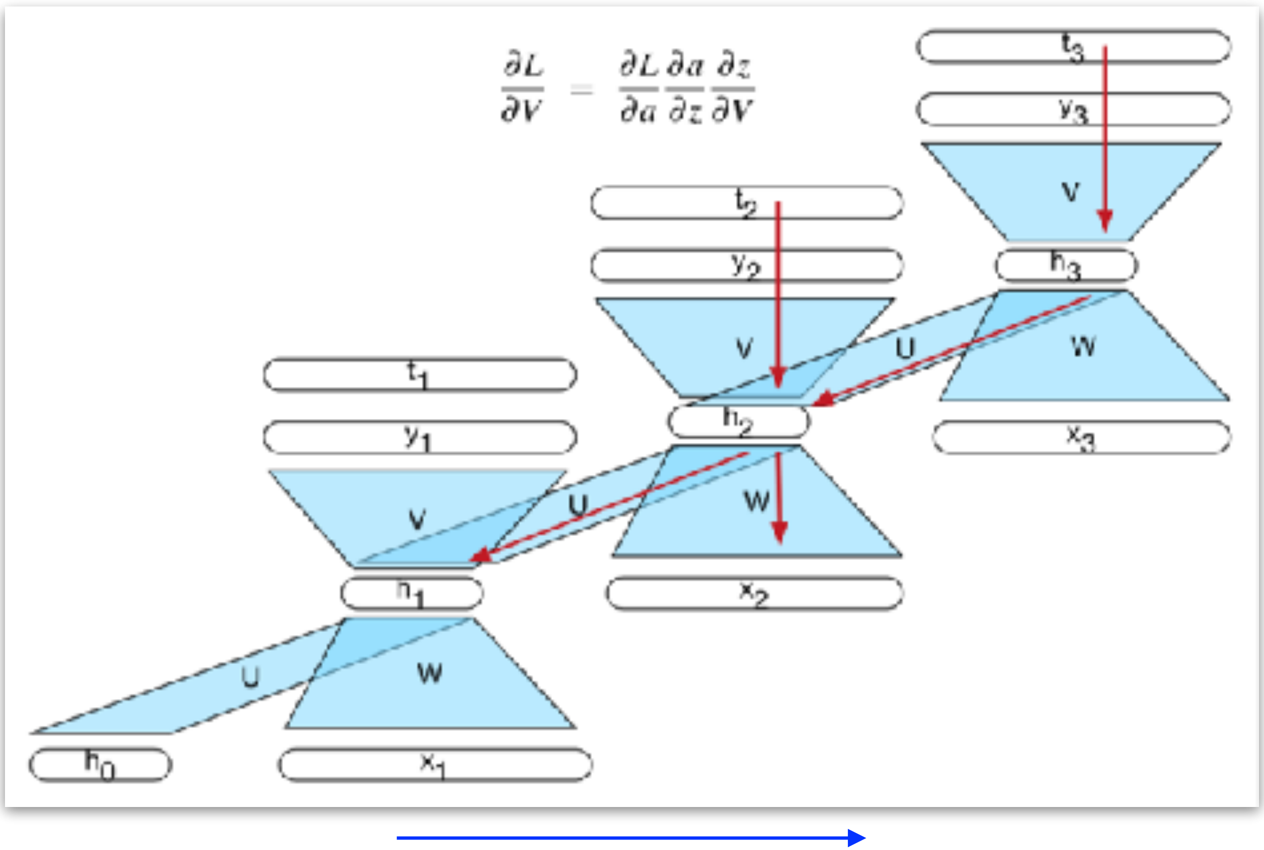
function FORWARDRNN(x, *network*) **returns** output sequence y

$h_0 \leftarrow 0$
for $i \leftarrow 1$ **to** LENGTH(x) **do**
  $h_i \leftarrow g(U\ h_{i-1}\ +\ W\ x_i)$
  $y_i \leftarrow f(V\ \ h_i)$
return y



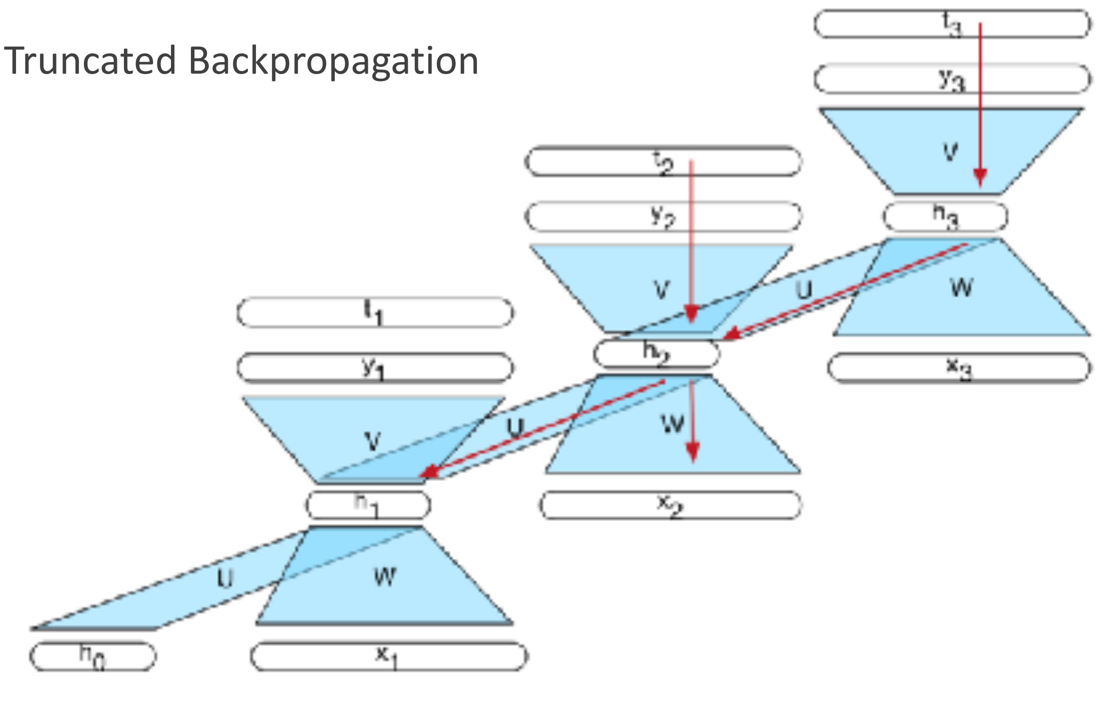
9.1.2 訓練

- 不同點
  - 輸入增加了前一時間 $h_{t\text{-}1}$ 的隱藏層狀態值 (就像是「歷史、記憶」)
  - 用 U 來表示如何運用「歷史」來計算輸出



9.1.3 把 RNN 展開為計算圖 computation graph

- 可以把 RNN 展開成為深度 FFNN 計算圖
  - 訓練資料可以看成同時輸入
  - 編譯對於該輸入的 FFNN
  - 做向前計算和向後擴散的訓練
- 有時候展開不實際 (太長)
  - 分段展開、分段訓練
  - 切斷式時序向後擴散 Truncated Backpropagation Through Time (TBTT).



9.2 RNNs 應用

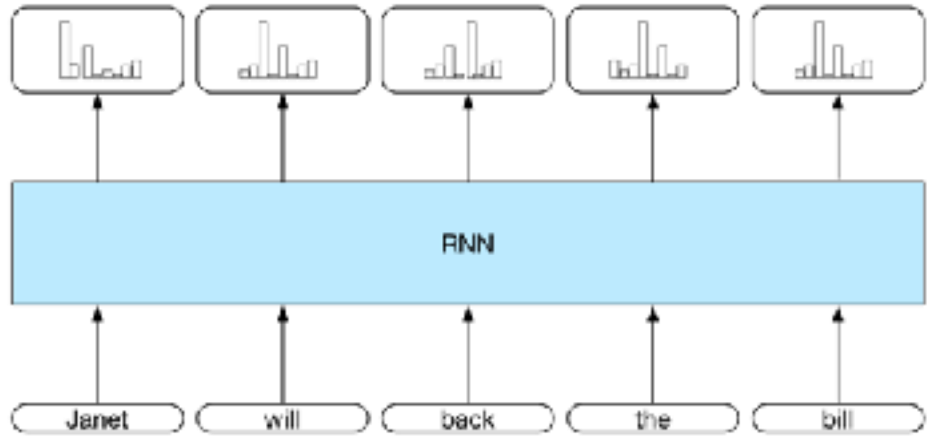
- 9.2.1 RNNs 語言模型
- 9.2.2 RNNs 序列標示
- 9.2.3 條件隨機域 (CRFs)
- 9.2.4 RNNs 序列分類器

9.2.1 語言模型

來源：https://github.com/fbchow/keras-rnn-demo
http://cs.rochester.edu/nlp/rocstories/
https://github.com/tensorflow/models
https://www.tensorflow.org/tutorials/sequences/recurrent

9.2.2 序列標示

- 詞性標註 POS tagging
  - 輸入：一串詞
  - 輸出：一串詞性的機率
    - 由最後的 softmax層產生
    - 每一詞性都有機率值
  - 訓練：用 cross entropy loss
- 專有名詞實體辨識 NER
  - IOB 編碼
  - IOB+NER分類
    - 人名、地點、組織
    - B-PER, I-PER, O
    - B-LOC, I-LOC
    - B-ORG, I-ORG



United	*cancelled*	*the*	*flight*	*from*	*Denver*	*to*	*San*	*Francisco.*
B	O	O	O	O	B	O	B	I

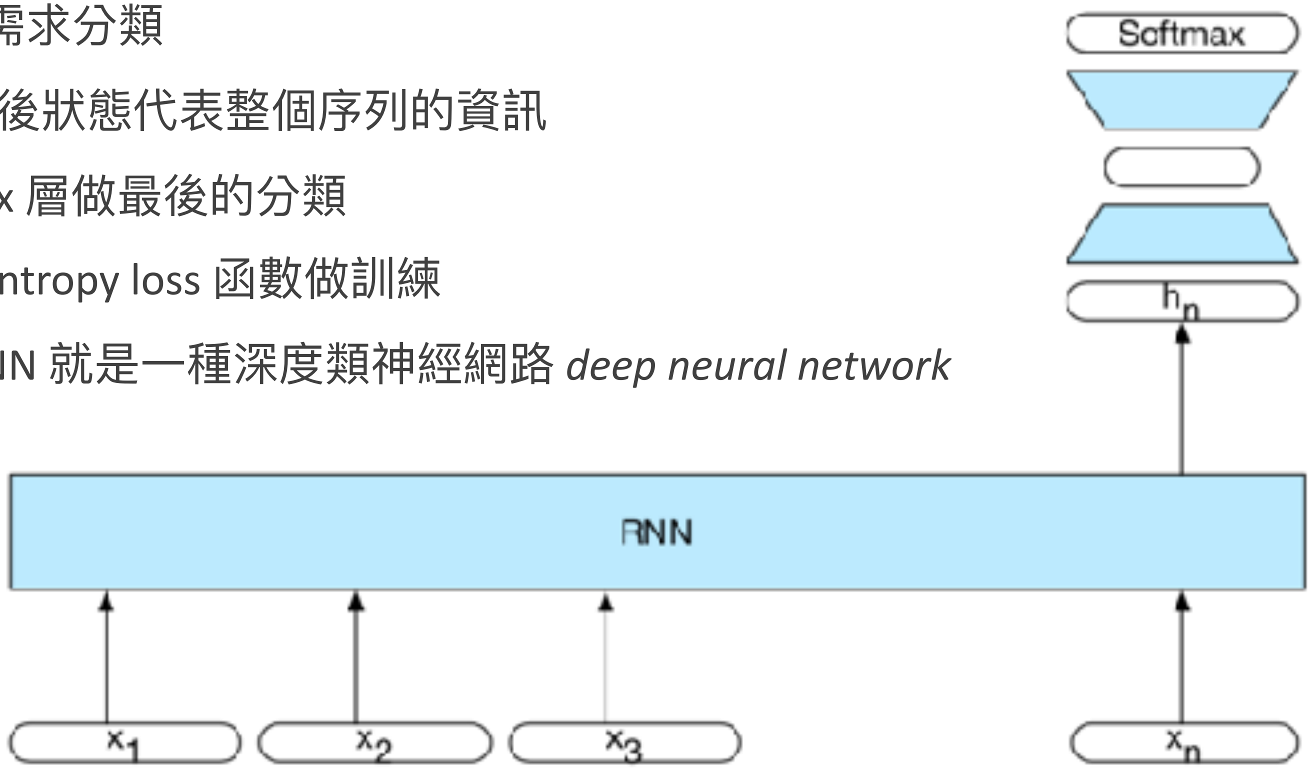
United	*cancelled*	*the*	*flight*	*from*	*Denver*	*to*	*San*	*Francisco.*
B-ORG	O	O	O	O	B-LOC	O	B-LOC	I-LOC

9.2.3 RNN+條件隨機場 (CRFs)

- 輸出有可能不合理
  - O I (必須是 O B 才合理)
  - B-PER I-LOC (必須是 B-PER I-PER 才合理)
- 用 RNN 的 softmax 輸出，然後再
  - 用詞的 LM 的 MEMM 做最後的決定
  - 用詞的 LM 的 CRF 做最後的決定

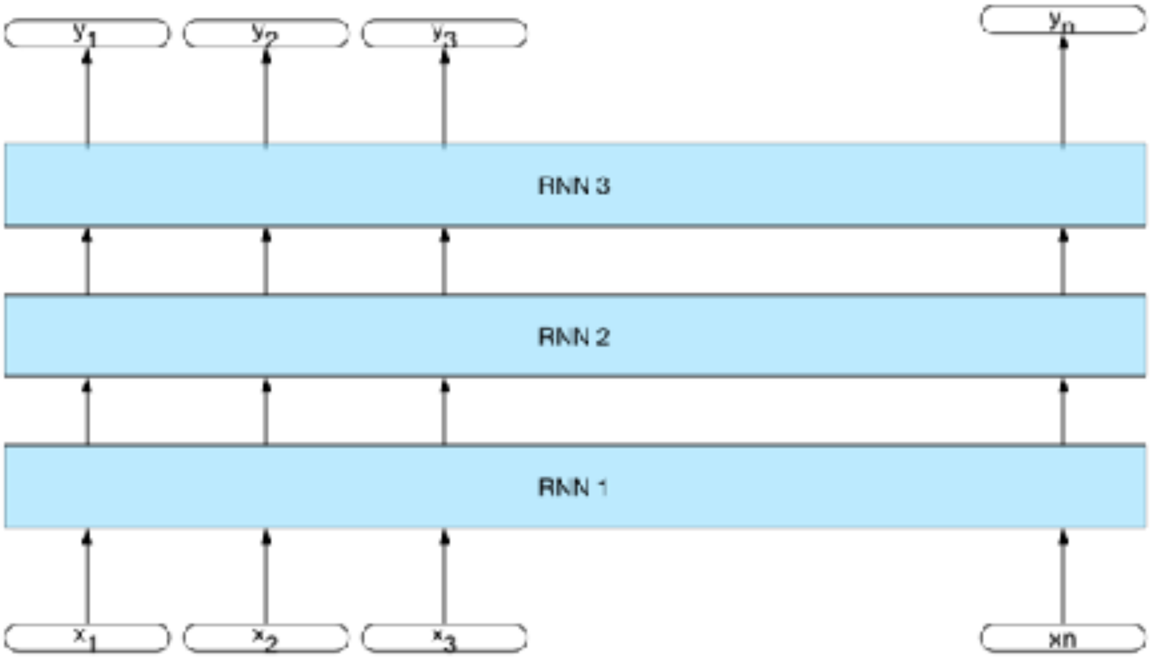
9.2.4 RNNs 序列分類器

- 可以做系列標註，也可以做序列的分類
  - 情意、意見分析 (電影評論的正反意見)
  - 文件主題分類
  - 垃圾郵件、 詐騙分類
  - 客服需求分類
- RNN 的最後狀態代表整個序列的資訊
- 用 softmax 層做最後的分類
- 用 cross-entropy loss 函數做訓練
- RNN + FFNN 就是一種深度類神經網路 *deep neural network*



9.3 深度網路：堆疊 RNNs

- 單層的 RNN 效能不如多層的堆疊 RNNs，因為多層 RNN 可
  - 學習推導不同層次的抽象特徵值
  - 很像人類視覺系統的前階段
    - 先辨識邊緣、輪廓
    - 然後用來辨識區域、形狀



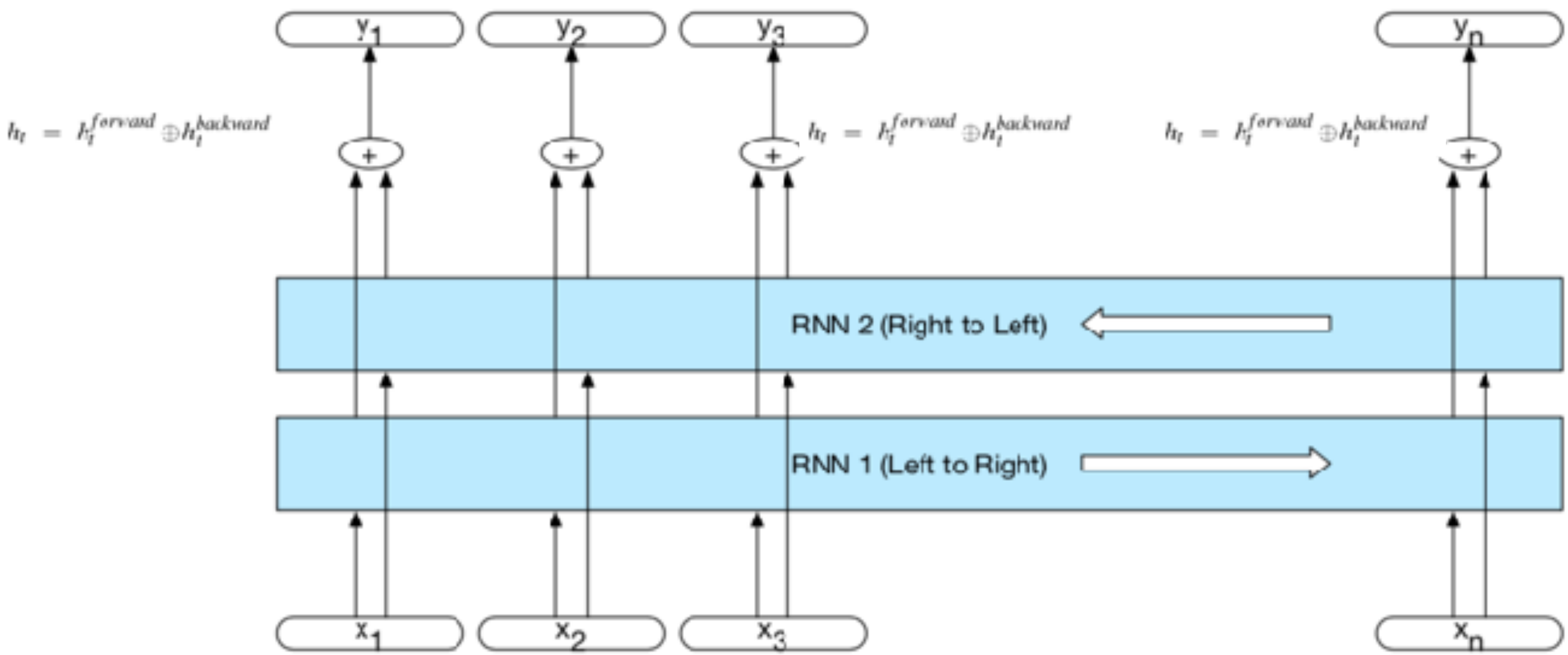
9.3 深度網路：雙向 RNNs 序列標註

- 實驗證實「雙向 RNNs」對序列分類有比較好的效果
- 單向 RNNs 的問題：最後狀態比最初狀態，有較多資訊
- 雙向 RNNs 結合向前與向後的資訊
  - 接續向前與向後的資訊
  - 逐一相加、相乘、平均

$$h_t^{forward} = SRN_{forward}(x_1 : x_t)$$

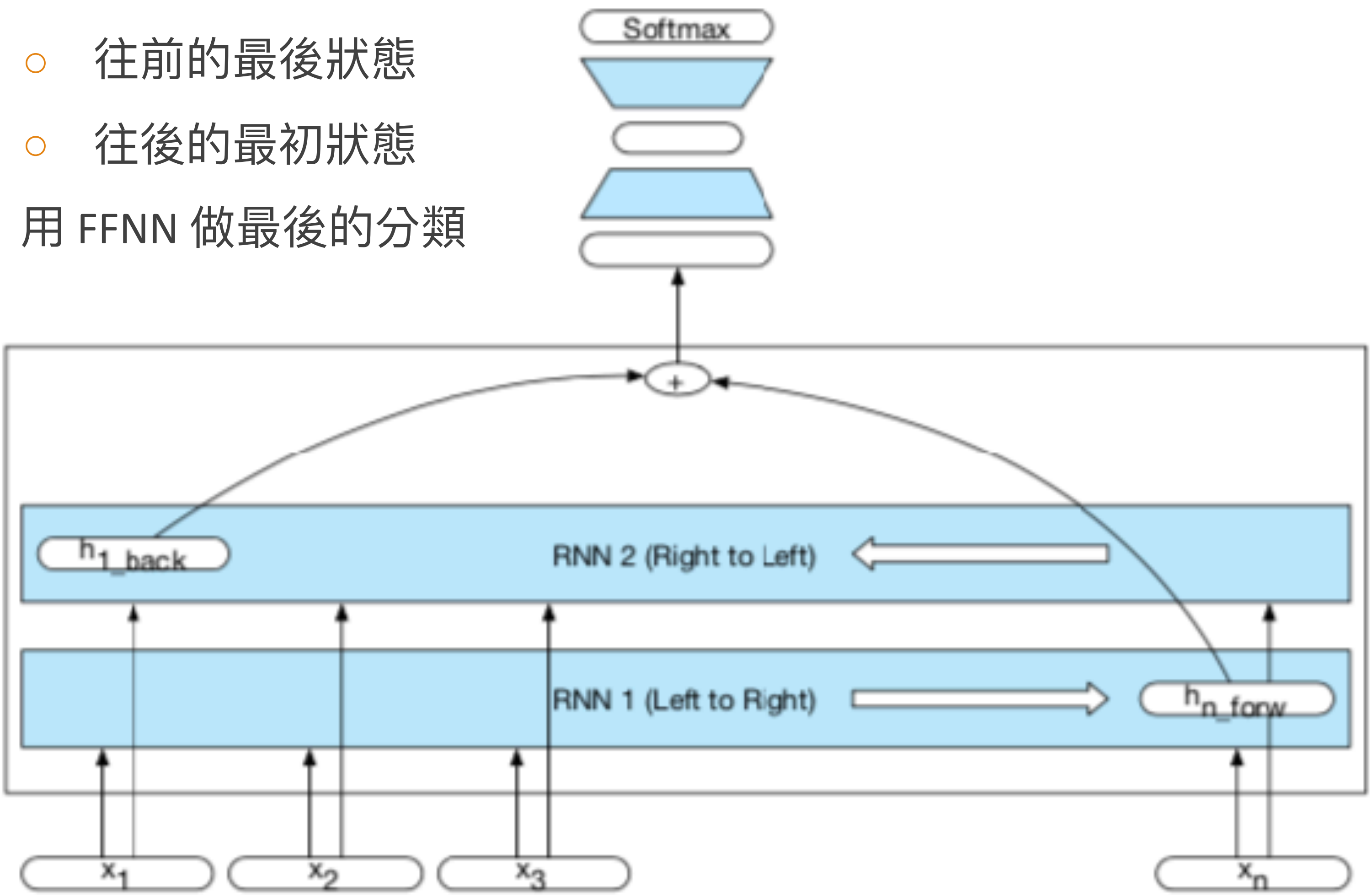
$$h_t^{backward} = SRN_{backward}(x_n : x_t)$$

$$h_t = h_t^{forward} \oplus h_t^{backward}$$



9.3 深度網路：雙向 RNNs 分類器

- 加總雙向資訊
  - 往前的最後狀態
  - 往後的最初狀態
- 用 FFNN 做最後的分類

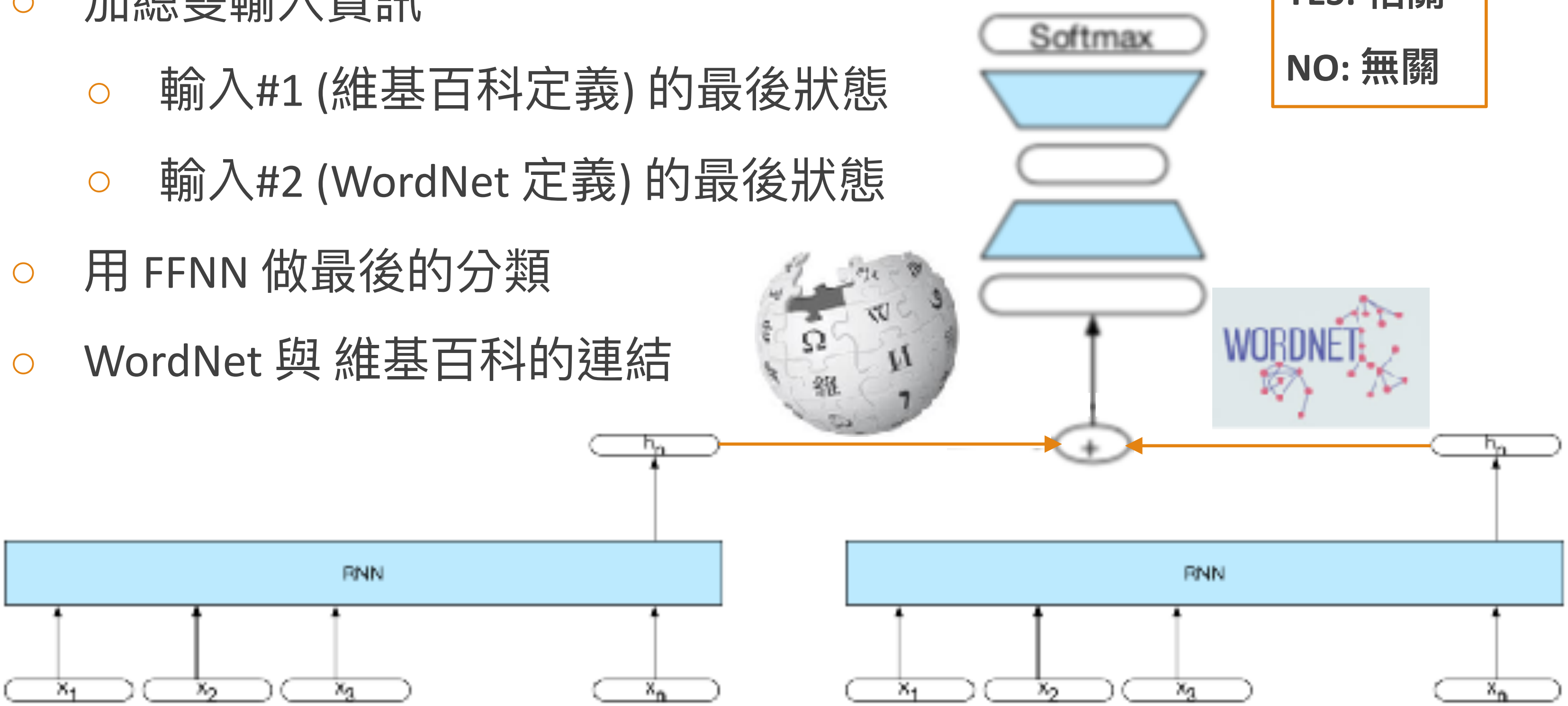


9.3 深度網路：雙輸入 RNNs 分類器

- 加總雙輸入資訊
  - 輸入#1 (維基百科定義) 的最後狀態
  - 輸入#2 (WordNet 定義) 的最後狀態
- 用 FFNN 做最後的分類
- WordNet 與 維基百科的連結

YES: 相關

NO: 無關



Plants are mainly multicellular, predominantly photosynthetic eukaryotes of the kingdom **Plantae** 植物

a living organism lacking the power of locomotion

buildings for carrying on industrial labor

an actor situated in the audience whose acting is rehearsed but seems spontaneous to the audience

something planted secretly for discovery by another

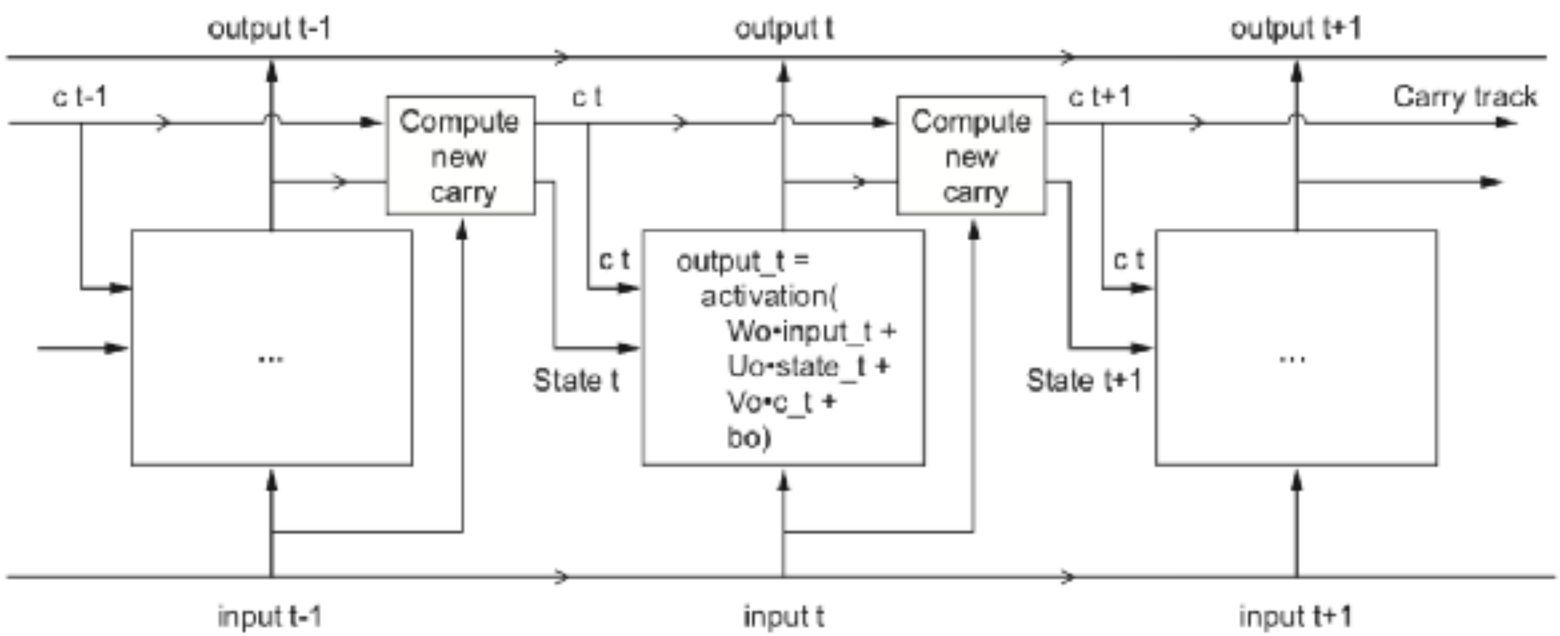
9.4 在 RNNS 中管制文脈：LSTMs 和 GRUs

- 9.4.1 長短期記憶 Long Short-Term Memory
- 9.4.2 閘門遞迴單元 Gated Recurrent Units
- 9.4.3 閘門單元,、層次、網路

9.4.1 長短期記憶 Long Short-Term Memory

- 用 LSTM 網路學習如何控管文脈
  - 移除不需要的資訊 (忘記) $c_t \; = \; f_t * c_{t-1}$
  - 記住需要的新資訊 (記憶) $c_t \mathrel{+}= i_t * g_t$
- LSTMs單元「像」用閘門控制文脈的資訊流動⟹增加權重
- 新權重 *g, i, f* 是對搜尋 (學習) 的限制 (不想成「設計」)
  - 讓過去資訊可由「輸送帶」重新加入對抗消失的梯度

$$
\begin{aligned}
g_t &= tanh(U_g h_{t-1} + W_g x_t) \\
i_t &= \sigma(U_i h_{t-1} + W_i x_t) \\
f_t &= \sigma(U_f h_{t-1} + W_f x_t) \\
o_t &= \sigma(U_o h_{t-1} + W_o x_t) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$



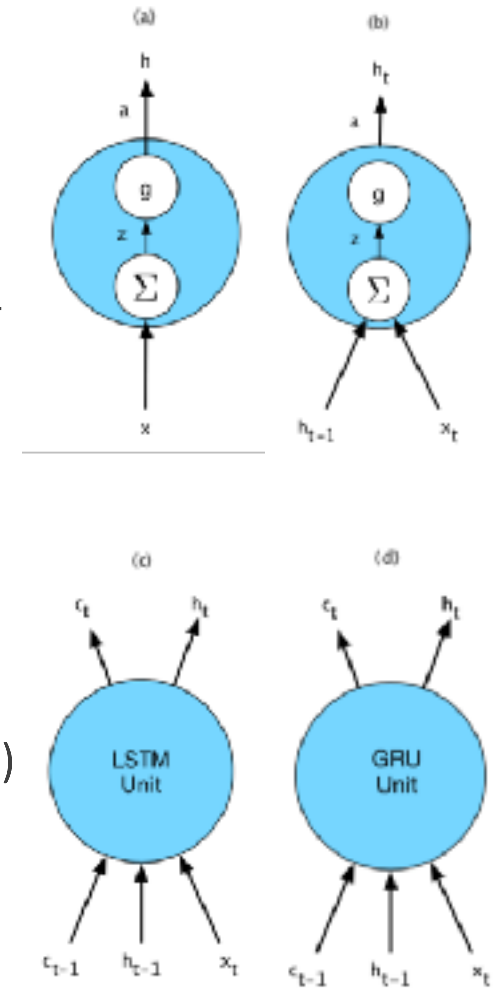
來源： Deep Learning with Python, p. 204

9.4.2 閘門遞迴單元 Gated Recurrent Units

- LSTMs的問題：更多的權重，更長的訓量時間
- 解決方案： 閘門遞迴單元 Gated Recur-rent Units (GRUs)
  - 合併遺忘閘門和累加閘門＝更新閘門
  - 降低權重數量

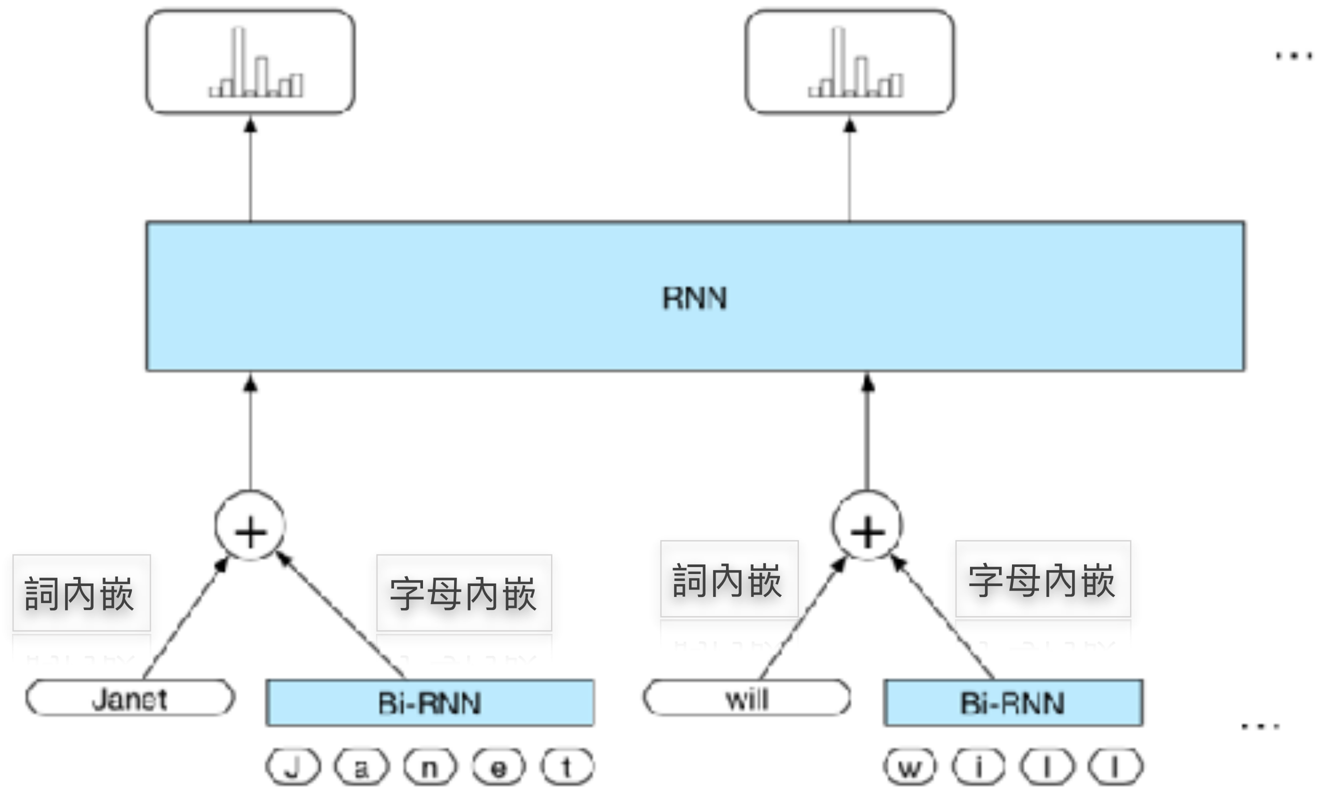
9.4.3 閘門單元,、層次、網路

- LSTMs 和 GRUs 用比較複雜的計算單元
- 複雜度限制在單元內不需調整，即可支持不同的架構 (模組化)
  - 只要把每個單元的輸出接到下一個端元的輸入，就可以了
- 這些單元可以用輸入、輸出來定義
  - (a) FF 單元 unit：$h = g(W\,x + b)$
  - (b) RRN 單元：2 輸入+激化函數 + 輸出 (加文脈記憶)
  - (c, d) LSTM, GRU 單元：3 輸入 + 2 輸出 (加短長期記憶)
- (b) 可代入 RNN，(c), (d) 可代入有文脈的 LSTM
- 多層的網路可展開成 FFNN，然後向後擴散訓練網路



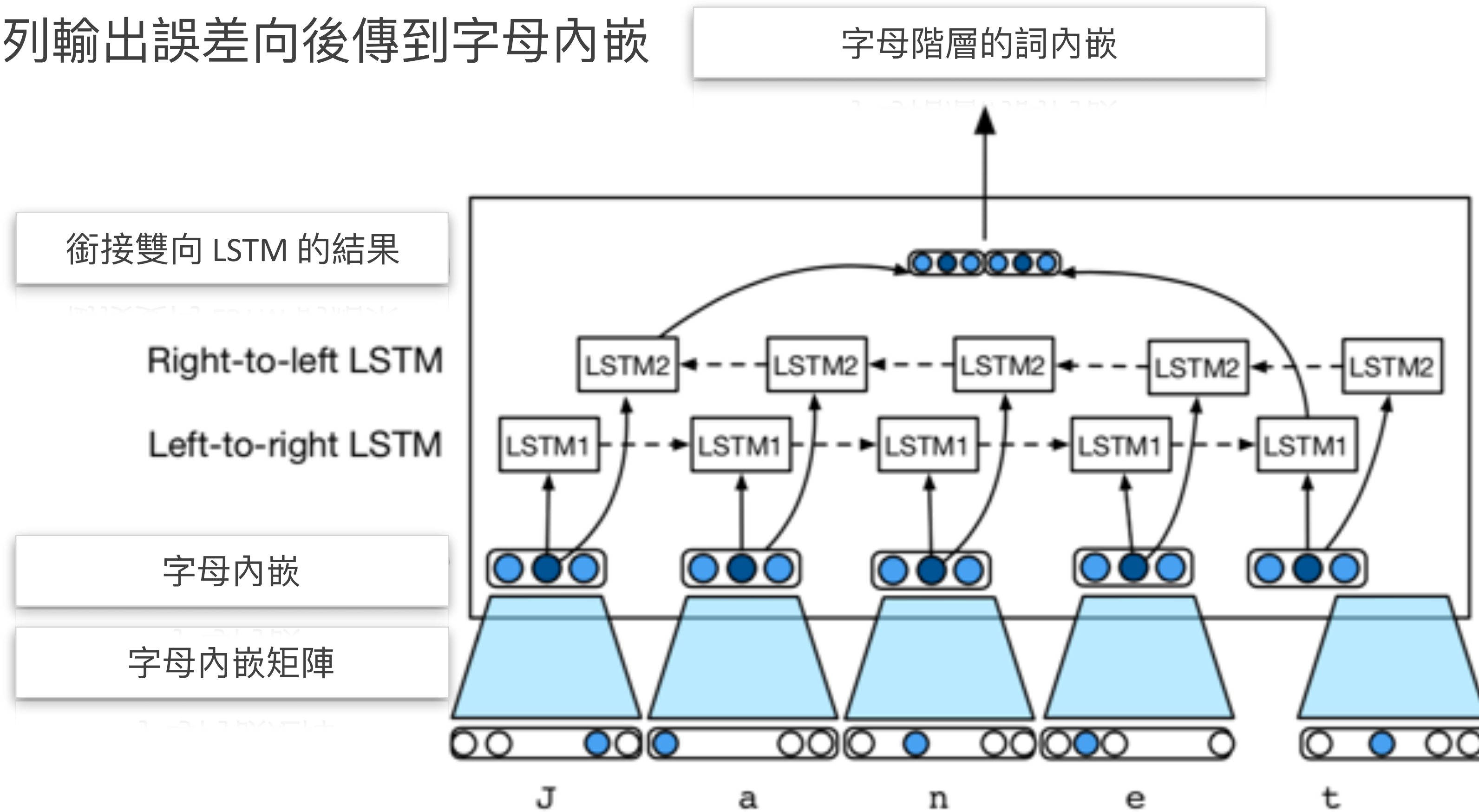
9.5 詞、字母、Byte-Pairs

- 用預先訓練或自行訓練詞內嵌都有困難—詞彙表通常很大
  - 限制詞彙範圍、應付未知詞
- 用一般的詞內嵌再加上字母資訊



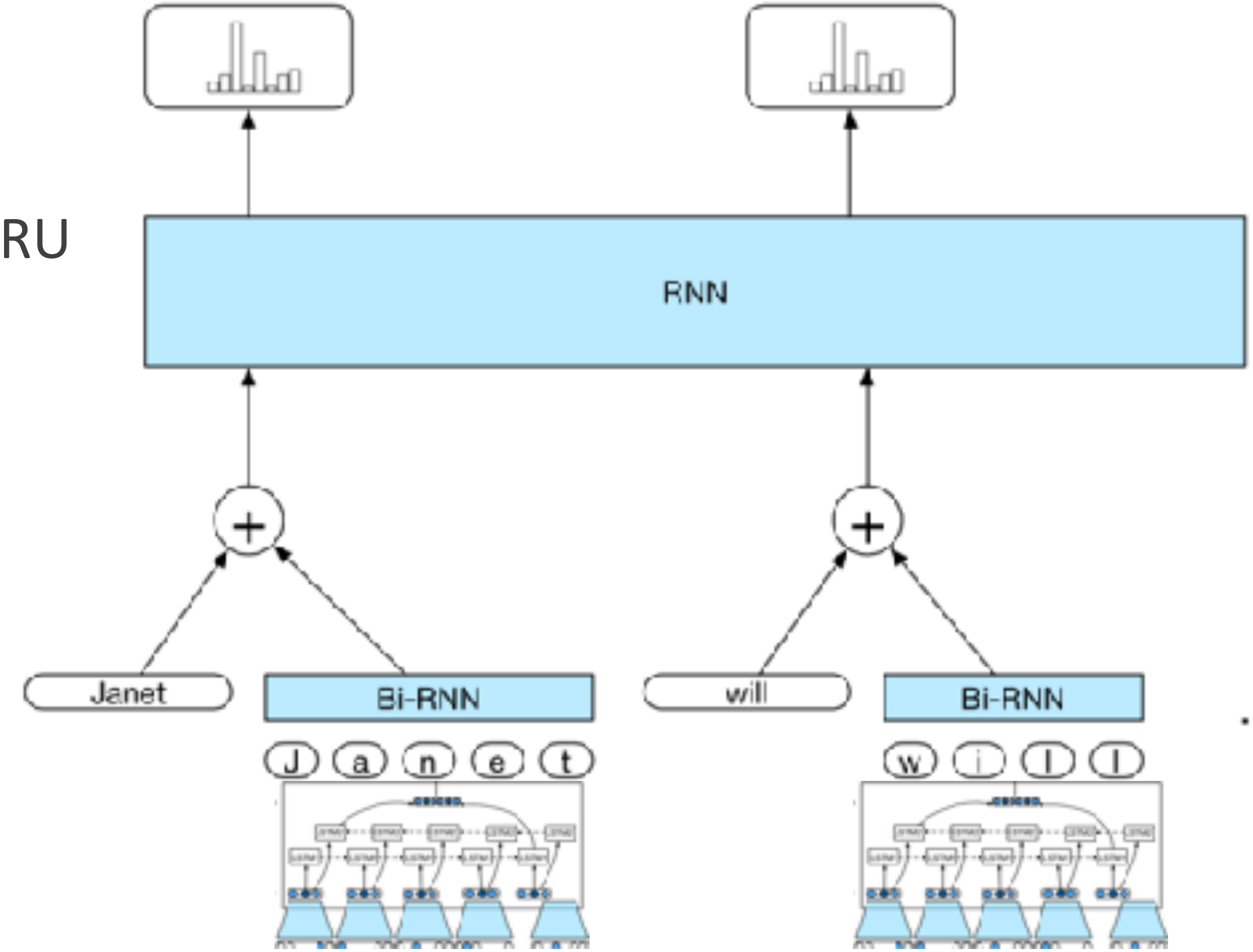
9.5 字母內嵌、字母階層詞內嵌

- 用雙向 LSTM 訓練字母內嵌 (如 J, a, n, e, t)
- 產生字母階層的詞內嵌 (如 Janet)
- 用任務的資料訓練字母內嵌
- 序列輸出誤差向後傳到字母內嵌



9.5 結合詞和字母內嵌

- 結合「詞內嵌」和「字母詞內嵌」
- 最後再匯入 RNN
    - 簡單 RNN
    - LSTM
    - LSTM with GRU



9.6 結語

- 簡單遞迴網路
- 執行和訓練 RNN
- 使用 RNN 的標註和分類應用
  - 序列到序列標註
  - 序列分類
- LSTMs and GRUs 可以處理
  - 消失的梯度
  - 長距離關係
- 用字母代表輸入
  - 應付大量詞彙表
  - 處理未知詞問題