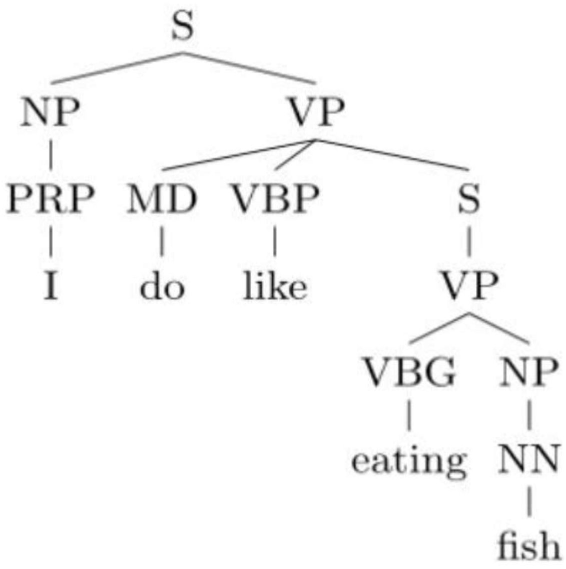
Neural Language Modeling by Jointly Learning Syntax and Lexicon

Yikang Shen, *Zhouhan Lin*, *Chin-wei Huang*, *Aaron Courville*
Universit de Montral

ICLR 2018

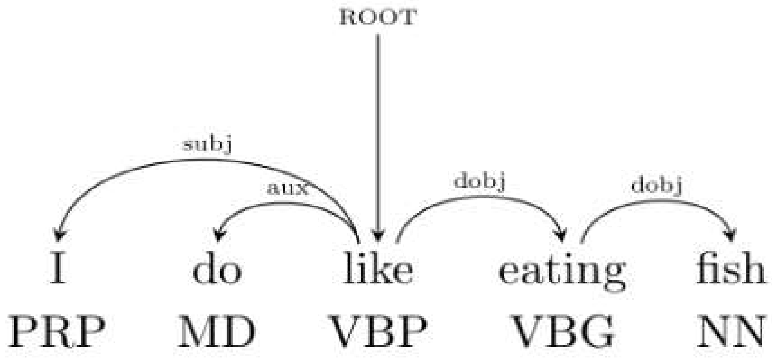
2018/10/29 莊永松

Syntax Parsing



Constituency Parsing

這篇要討論的



Dependency Parsing

Related Work and Contribution

- 原本普遍作法: 人工手標一堆 Supervised data，再用 Recursive model 做
  Socher et al., 2010; Alvarez-Melis & Jaakkola, 2016;
  Zhou et al., 2017; Zhang et al., 2015
- 這篇的貢獻: **Unsupervised** parsing without label data 表現還OK
- 並且順便在 Word/Char-level language modeling 拿下了 state of the art

Parsing-Reading-Predict Networks (PRPN)

- Parsing Network
- Reading Network
- Predict Network

不是純 RNN 或 RecursiveNN，而像是兩者結合
Unsupervised training target: **Language Model**

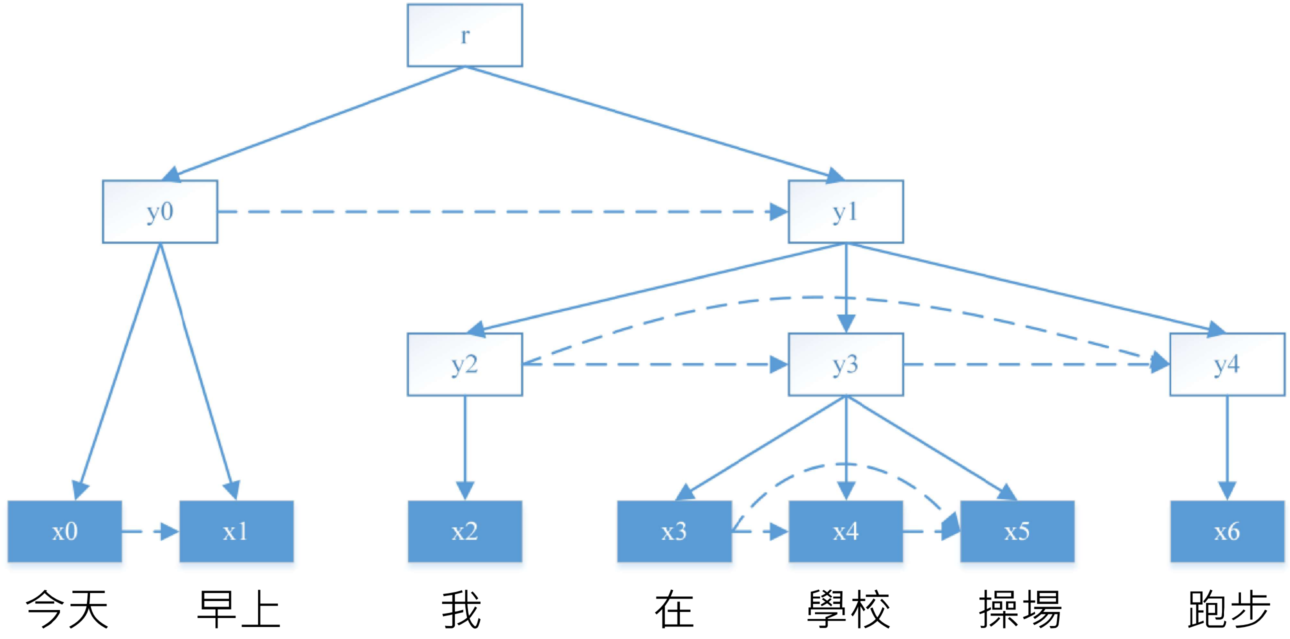
Tasks:

- Word-level language modeling
- Character-level language modeling
- **Unsupervised constituency parsing**

Before Intro the Model…

- **Hard arrow**: syntactic tree structure and parent-to-child dependency relation
- **Dash arrow:** dependency relation between siblings

假設：此處的dependency relation，存在於任意<u>left-to-right</u> 的sibling之間。



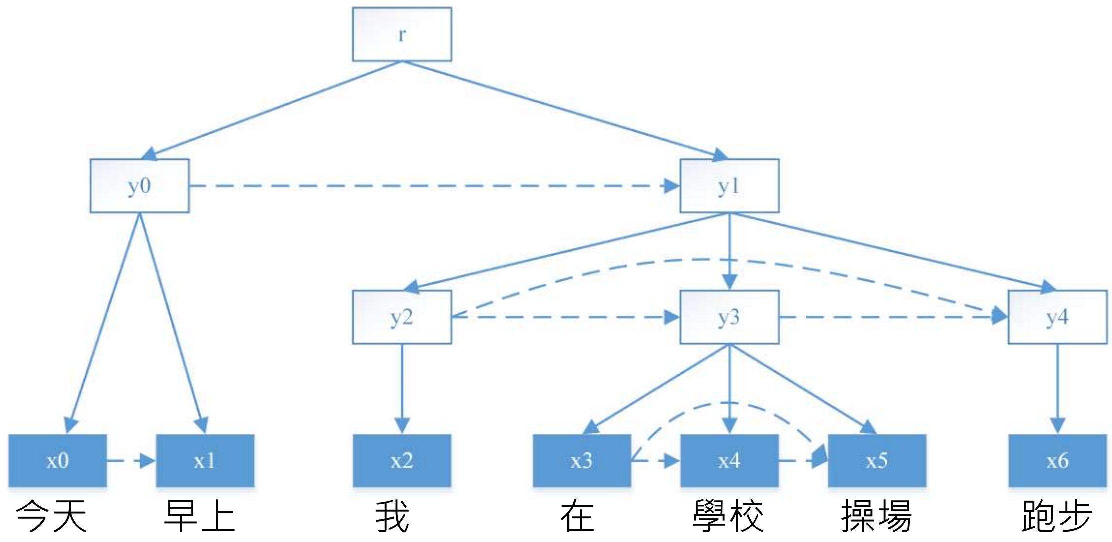
整篇的Basic idea

在做LM預測next word的時候
誰最重要呢？

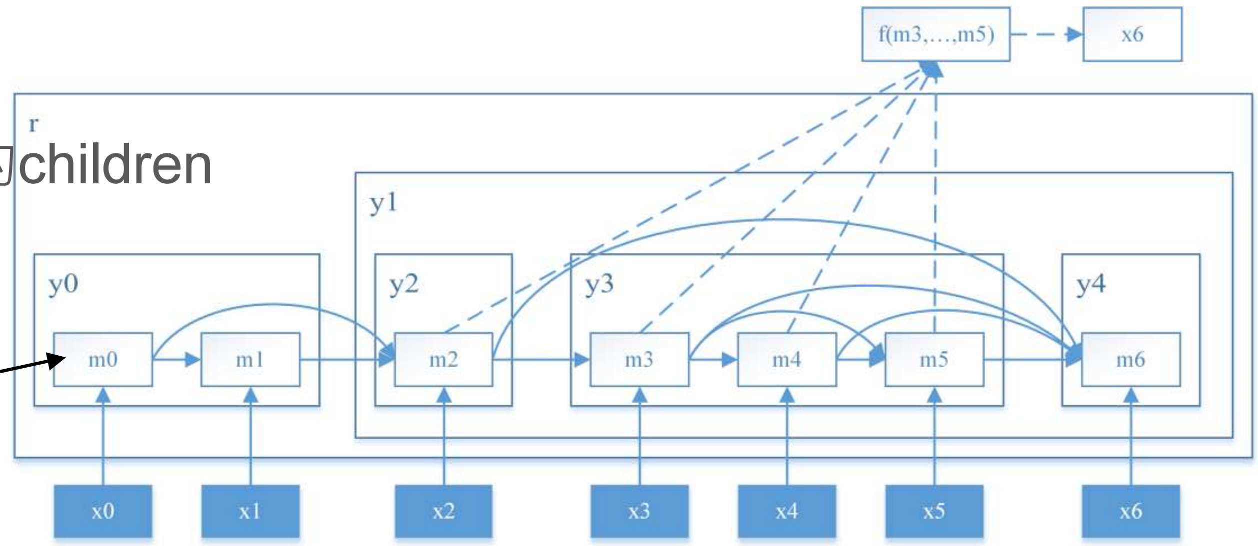
Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要

Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

→ 圖中的Hard arrow就是了



m_i: hidden state



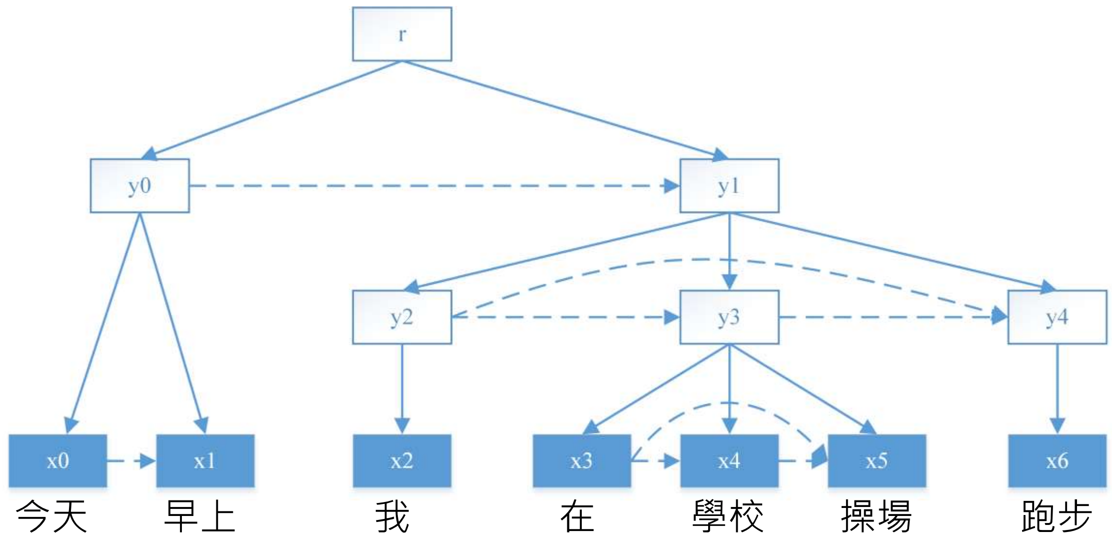
整篇的Basic idea

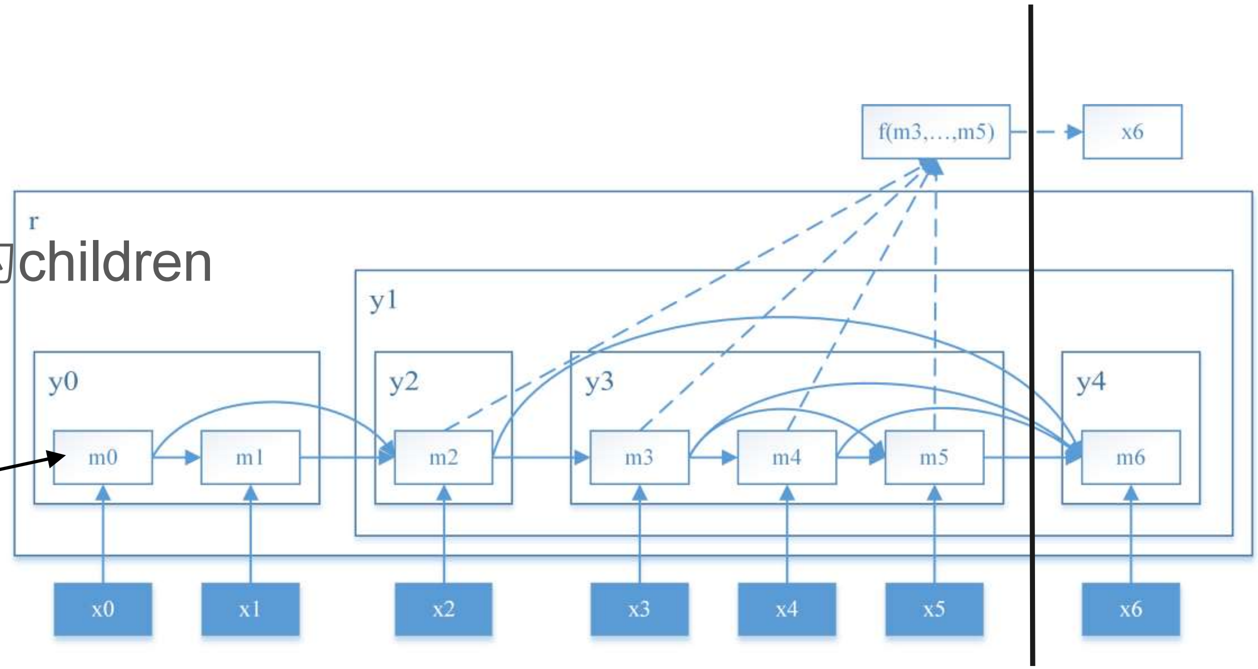
在做LM預測next word的時候
誰最重要呢？

Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要

Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

→ 圖中的Hard arrow就是了





m_i: hidden state

整篇的Basic idea

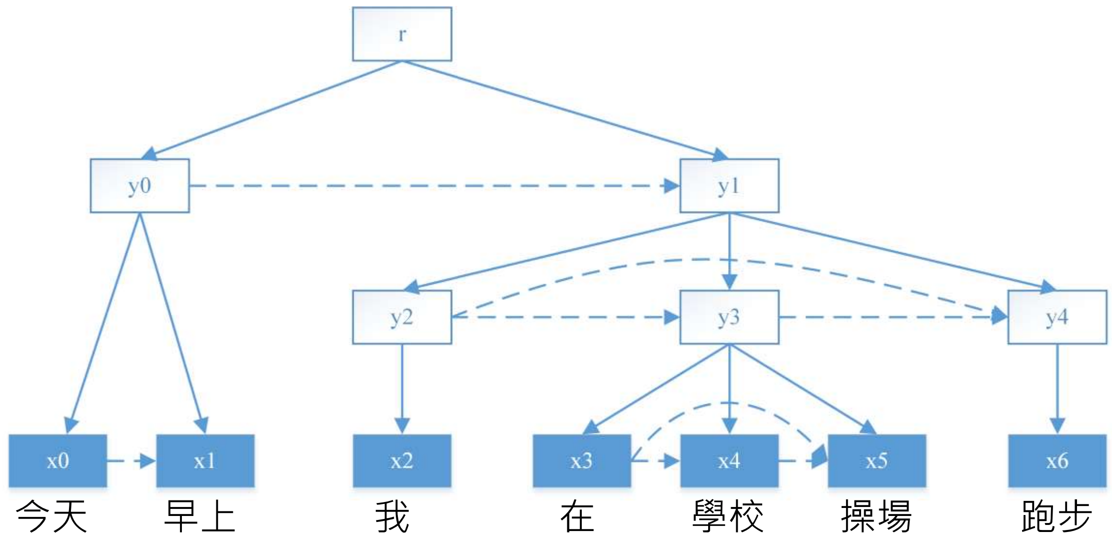
在做LM預測next word的時候
誰最重要呢？

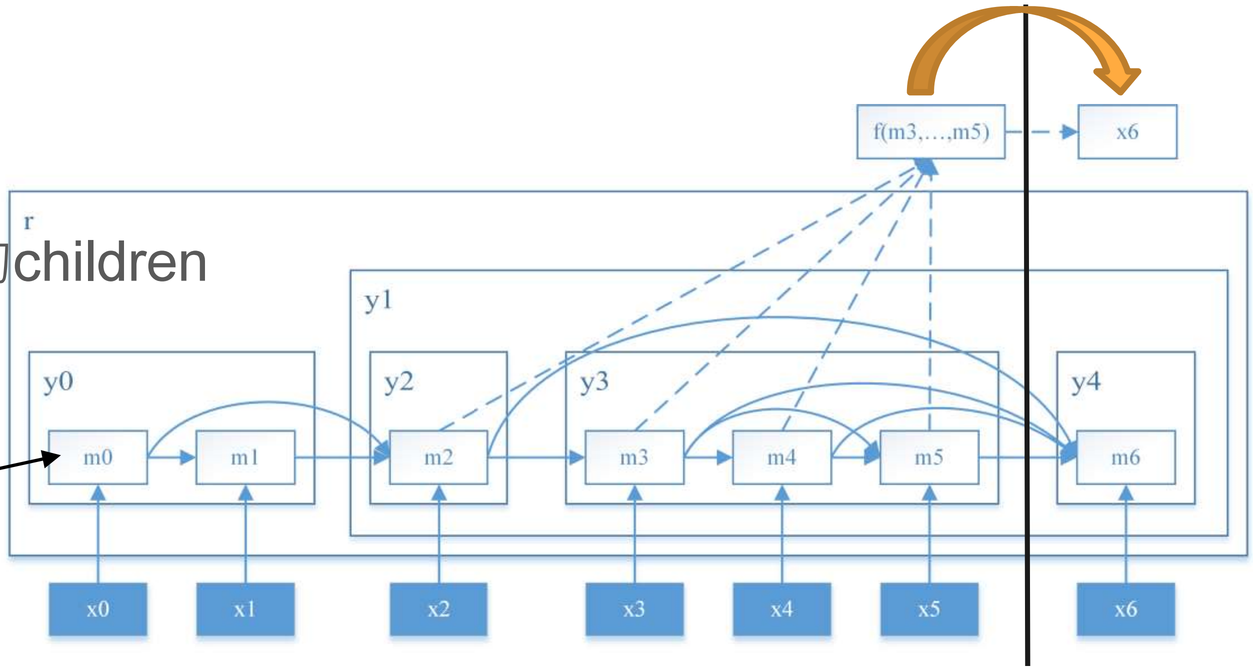
Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要

Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

→ 圖中的Hard arrow就是了

m_i: hidden state





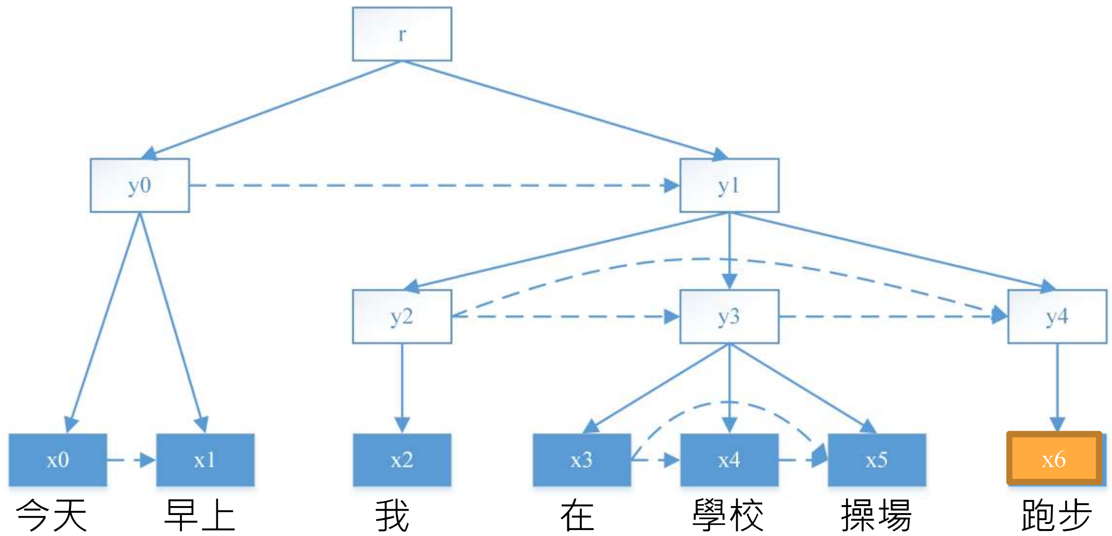
整篇的Basic idea

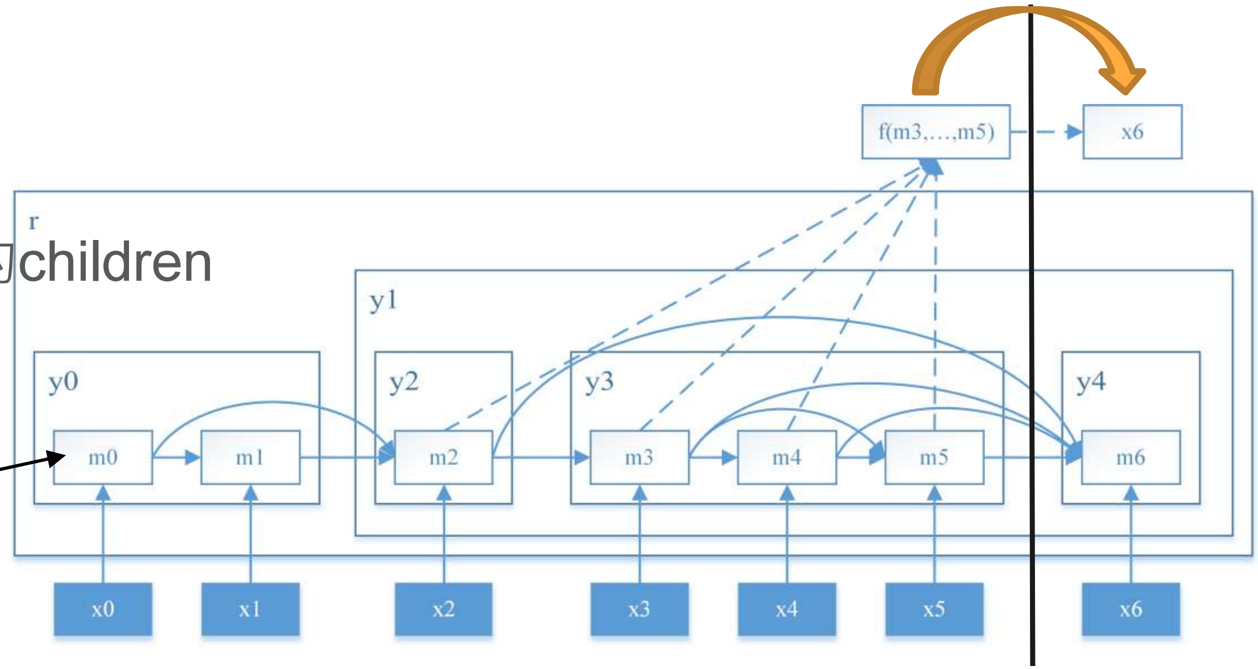
在做LM預測next word的時候
誰最重要呢？

Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要

Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

→ 圖中的Hard arrow就是了





m_i: hidden state

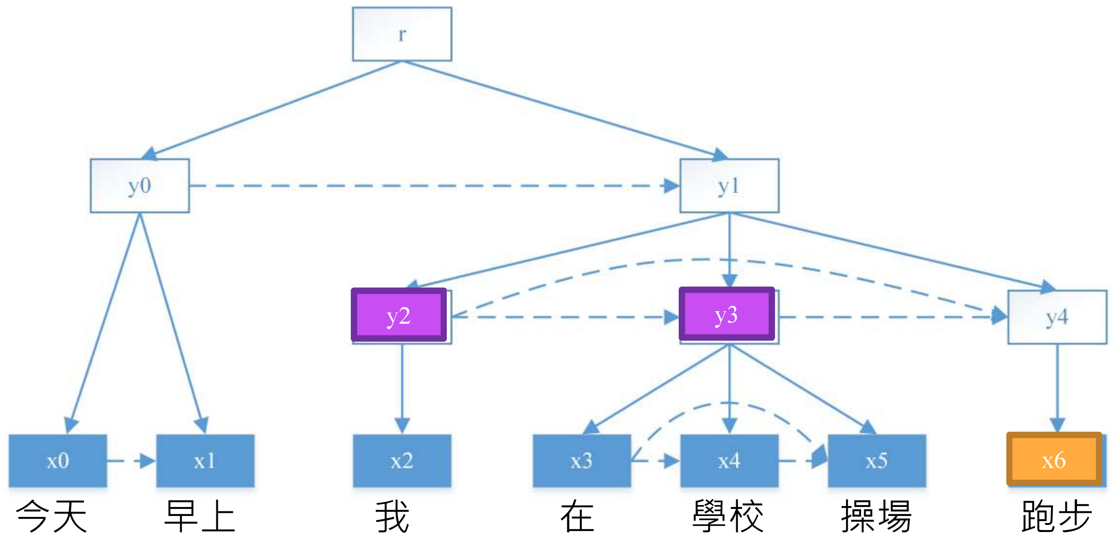
整篇的Basic idea

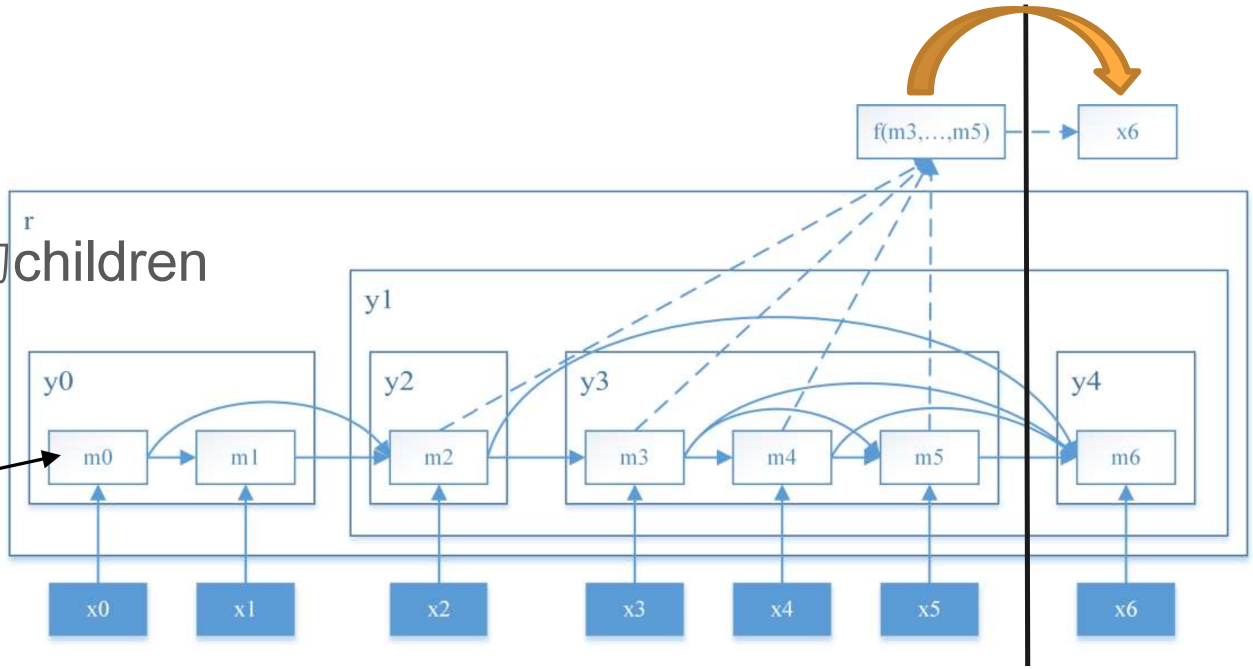
在做LM預測next word的時候
誰最重要呢？

Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要

Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

→ 圖中的Hard arrow就是了



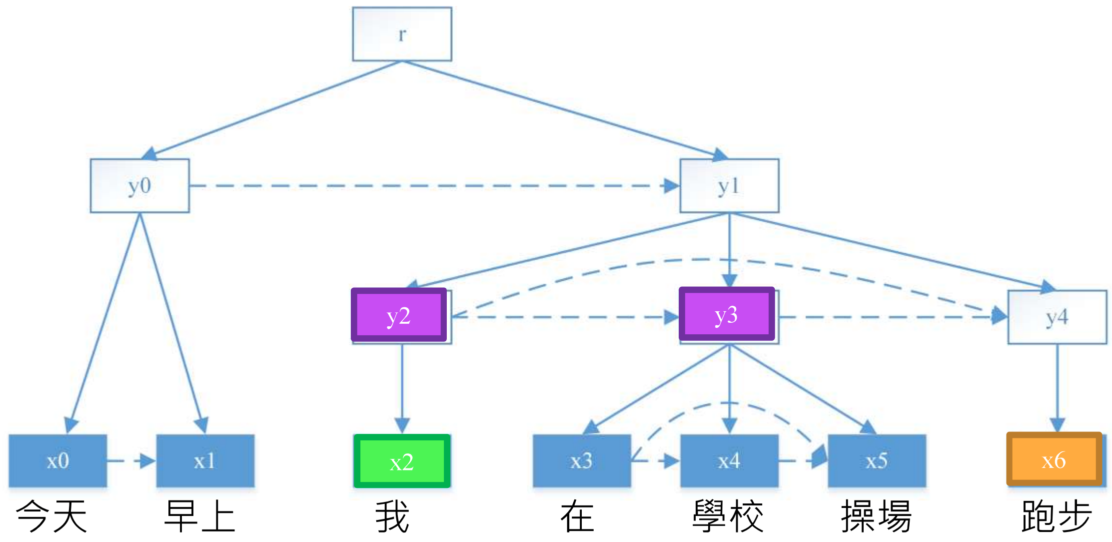


m_i: hidden state

整篇的Basic idea

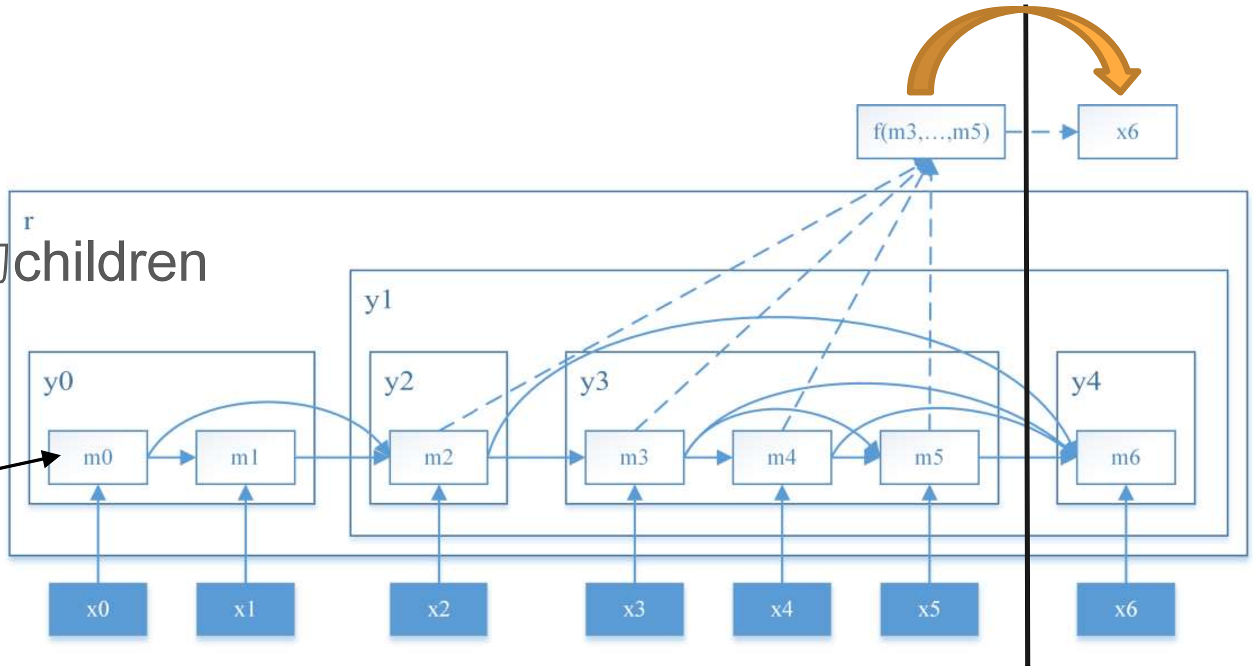
在做LM預測next word的時候
誰最重要呢？

Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要



Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

→ 圖中的Hard arrow就是了



m_i: hidden state

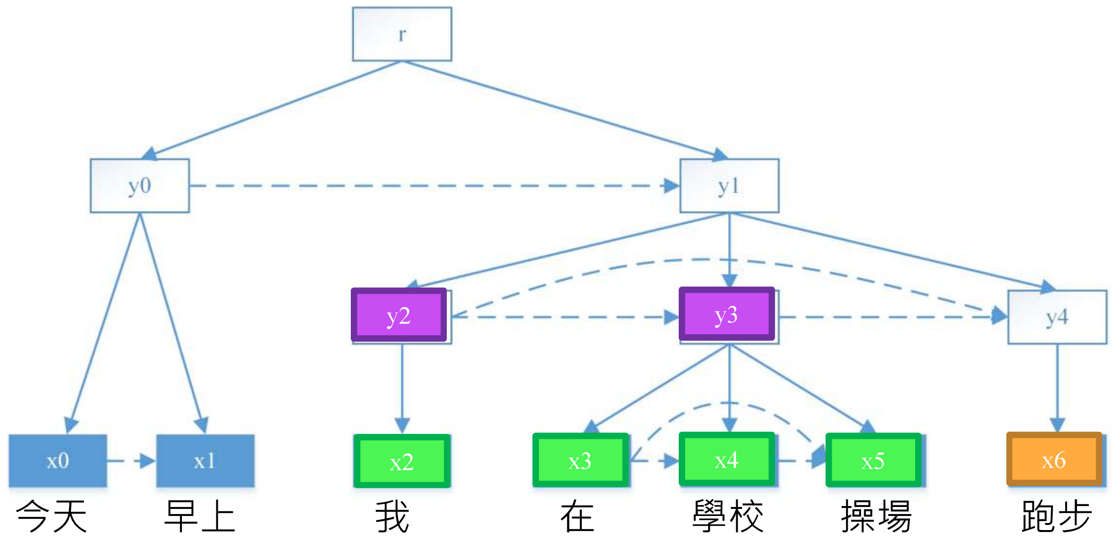
整篇的Basic idea

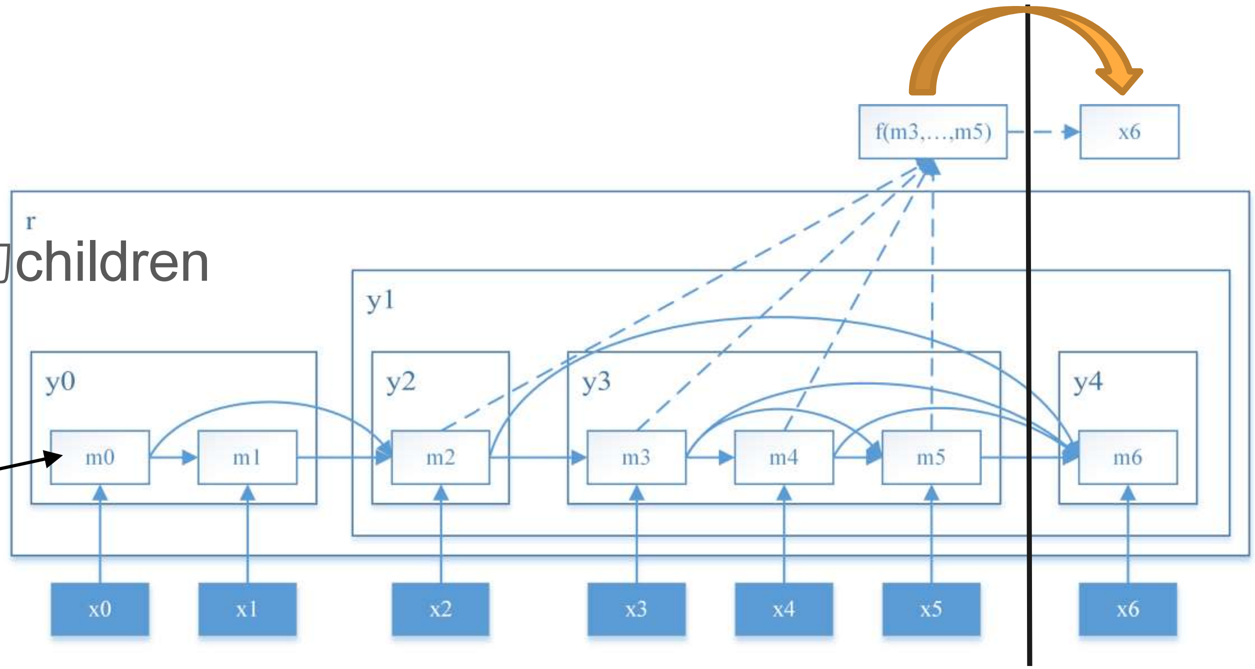
在做LM預測next word的時候
誰最重要呢？

Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要

Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

→ 圖中的Hard arrow就是了





m_i: hidden state

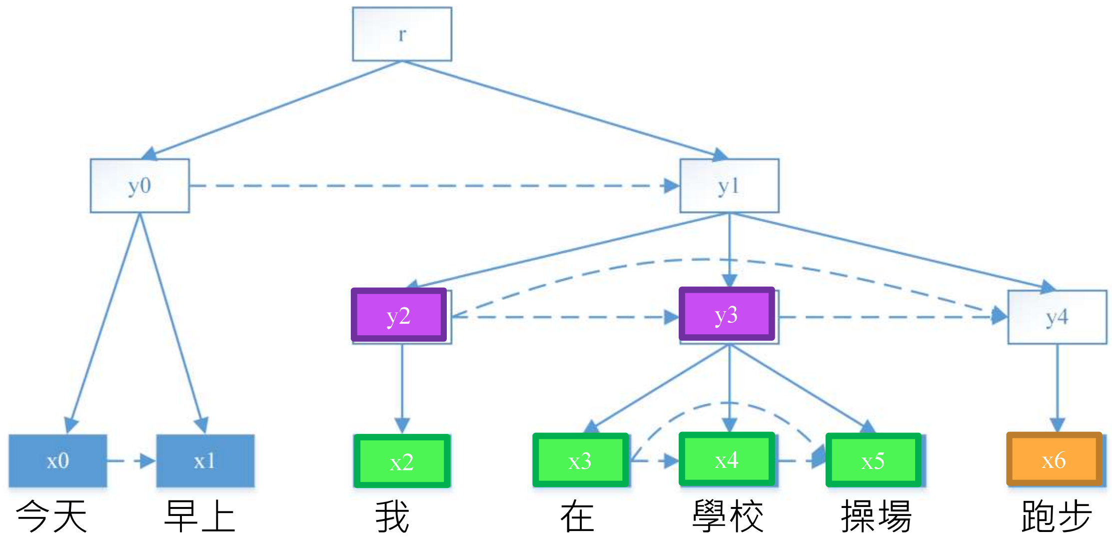
整篇的Basic idea

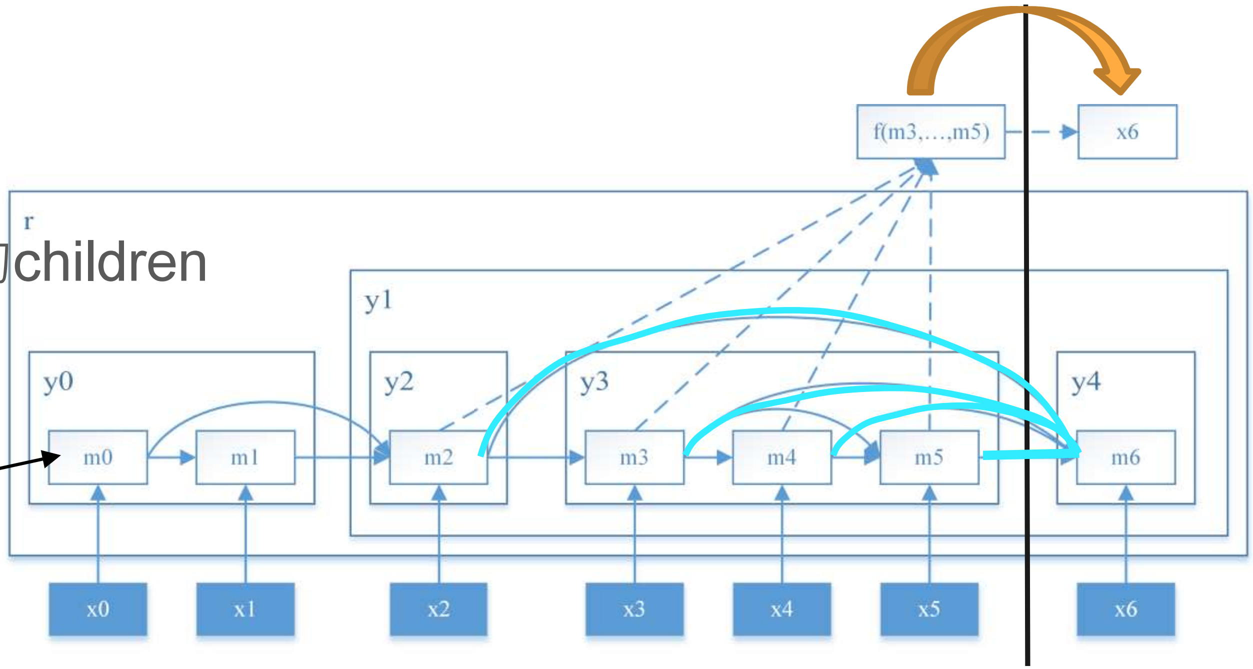
在做LM預測next word的時候
誰最重要呢？

Case 1: 要預測的word不是left-most-child
→ 他左邊的sibling最重要

Case 2: 要預測的word是left-most-child
→ 他的parent的左邊的sibling下面的children

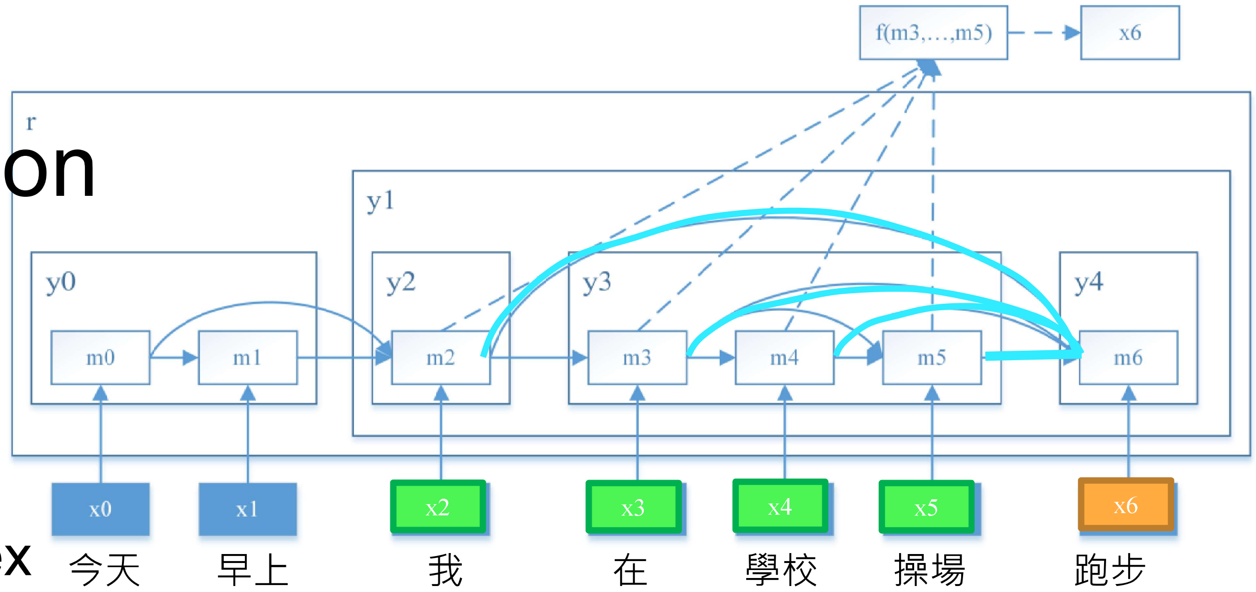
→ 圖中的Hard arrow就是了





m_i: hidden state

Discrete skip-connection

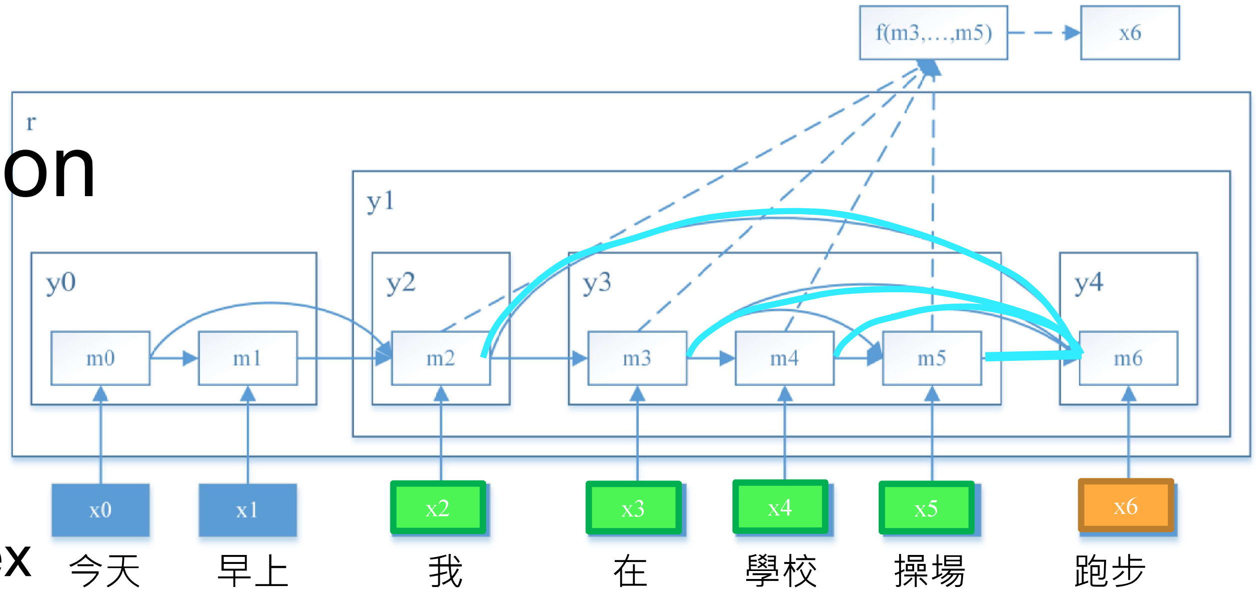


用一個 gate function 做 skip-connection，達到上一頁所說的效果

- l_t: 所依賴的word中最遠的index
- $$g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 < i < l_t \end{cases}$$
- $m_t = h(x_t, m_0, ..., m_{t-1}, g_0^t, ..., g_{t-1}^t)$
- $$p(x_{t+1}|x_0, ..., x_t) \approx p(x_{t+1}; f(m_0, ..., m_t, g_0^{t+1}, ..., g_t^{t+1}))$$

上面兩件事是分開做的，當你要預測x6的時候，你還沒有m6，只有m5

Discrete skip-connection



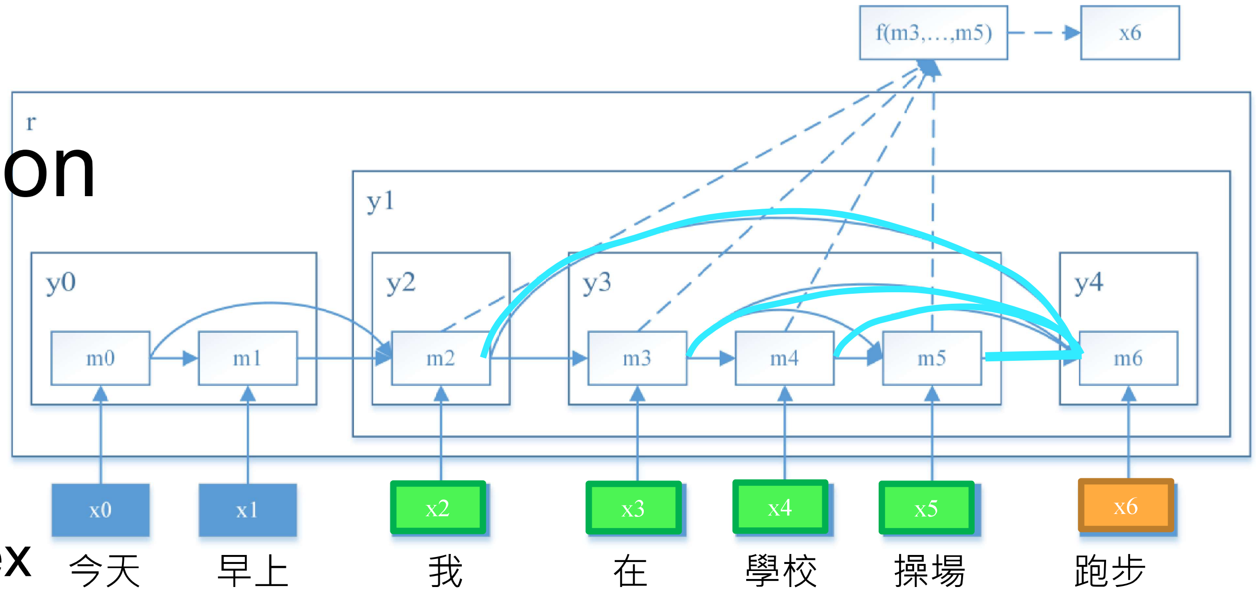
$l_6 = 2$

用一個 gate function 做 skip-connection，達到上一頁所說的效果

- l_t: 所依賴的word中最遠的index
- $$g_i^t = \begin{cases} 1, & l_t \le i < t \\ 0, & 0 < i < l_t \end{cases}$$
- $m_t = h(x_t, m_0, ..., m_{t-1}, g_0^t, ..., g_{t-1}^t)$
- $$p(x_{t+1} | x_0, ..., x_t) \approx p(x_{t+1}; f(m_0, ..., m_t, g_0^{t+1}, ..., g_t^{t+1}))$$

上面兩件事是分開做的，當你要預測x6的時候，你還沒有m6，只有m5

Discrete skip-connection



$l_6 = 2$

用一個 gate function 做 skip-connection，達到上一頁所說的效果

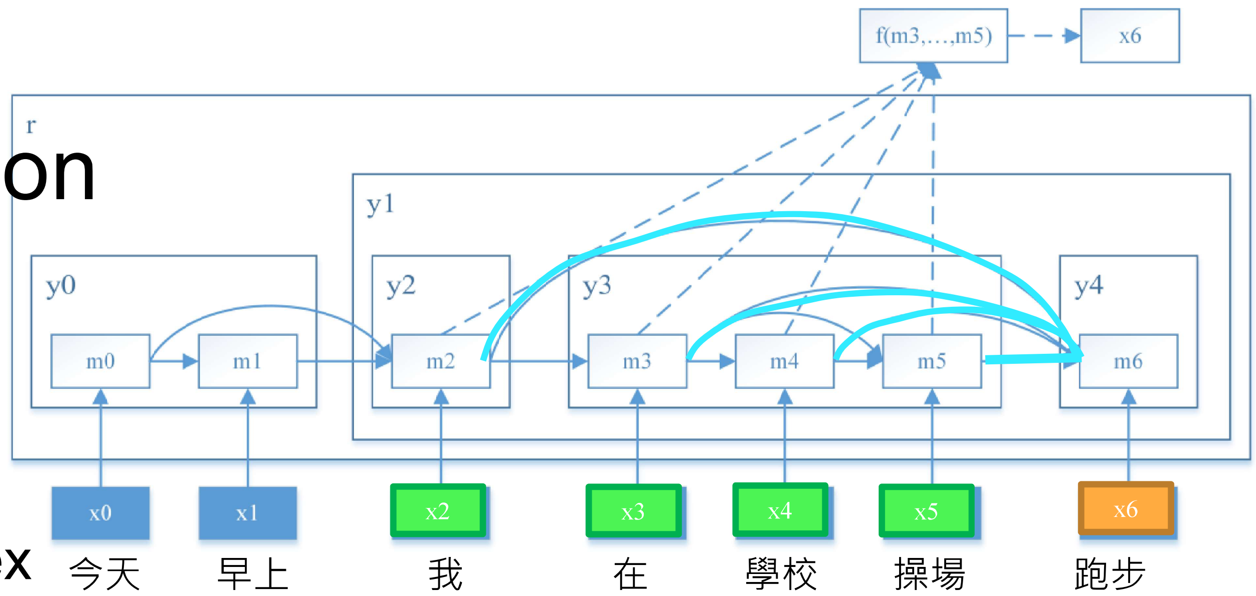
- l_t: 所依賴的word中最遠的index
- $$g_i^t = \begin{cases} 1, & l_t \le i < t \\ 0, & 0 < i < l_t \end{cases}$$

$g_0^6 = 0 \quad g_1^6 = 0$

- $m_t = h(x_t, m_0, \ldots, m_{t-1}, g_0^t, \ldots, g_{t-1}^t)$
- $$p(x_{t+1}|x_0, \ldots, x_t) \approx p(x_{t+1}; f(m_0, \ldots, m_t, g_0^{t+1}, \ldots, g_t^{t+1}))$$

上面兩件事是分開做的，當你要預測x6的時候，你還沒有m6，只有m5

Discrete skip-connection



用一個 gate function 做 skip-connection，達到上一頁所說的效果

- l_t: 所依賴的word中最遠的index
- $$g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 < i < l_t \end{cases}$$
- $m_t = h(x_t, m_0, ..., m_{t-1}, g_0^t, ..., g_{t-1}^t)$
- $p(x_{t+1}|x_0, ..., x_t) \approx p(x_{t+1}; f(m_0, ..., m_t, g_0^{t+1}, ..., g_t^{t+1}))$

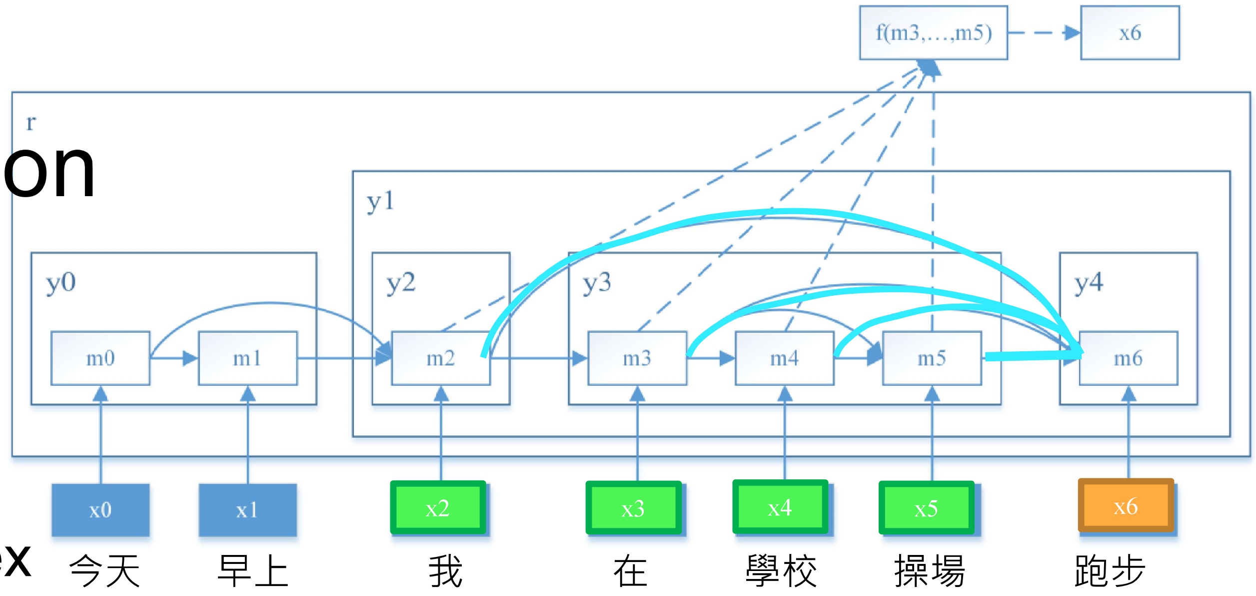
$l_6 = 2$

$g_0^6 = 0$ $g_1^6 = 0$ $g_2^6 = 1$ $g_3^6 = 1$ $g_4^6 = 1$ $g_5^6 = 1$

上面兩件事是分開做的，當你要預測x6的時候，你還沒有m6，只有m5

Discrete skip-connection

用一個 gate function 做 skip-connection，達到上一頁所說的效果



$l_6 = 2$

$g_0^6 = 0 \quad g_1^6 = 0 \quad g_2^6 = 1 \quad g_3^6 = 1 \quad g_4^6 = 1 \quad g_5^6 = 1$

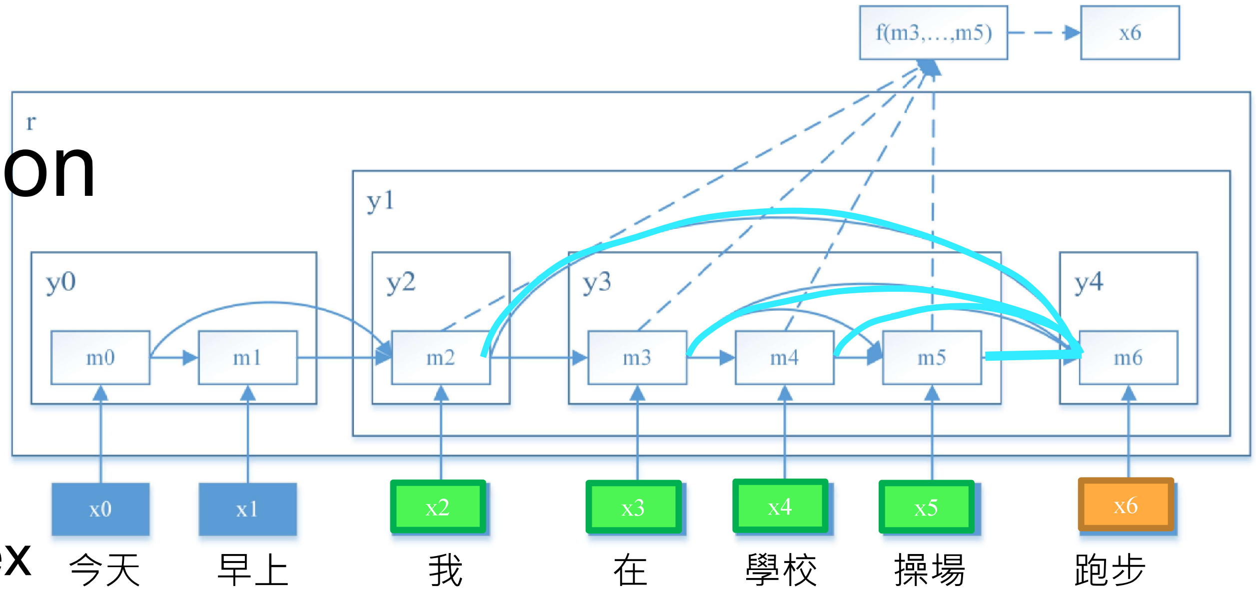
- l_t: 所依賴的word中最遠的index
- $$g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 < i < l_t \end{cases}$$

Parsing network

- $m_t = h(x_t, m_0, \ldots, m_{t-1}, g_0^t, \ldots, g_{t-1}^t)$
- $p(x_{t+1} | x_0, \ldots, x_t) \approx p(x_{t+1}; f(m_0, \ldots, m_t, g_0^{t+1}, \ldots, g_t^{t+1}))$

上面兩件事是分開做的，當你要預測x6的時候，你還沒有m6，只有m5

Discrete skip-connection



用一個 gate function 做 skip-connection，達到上一頁所說的效果

$l_6 = 2$

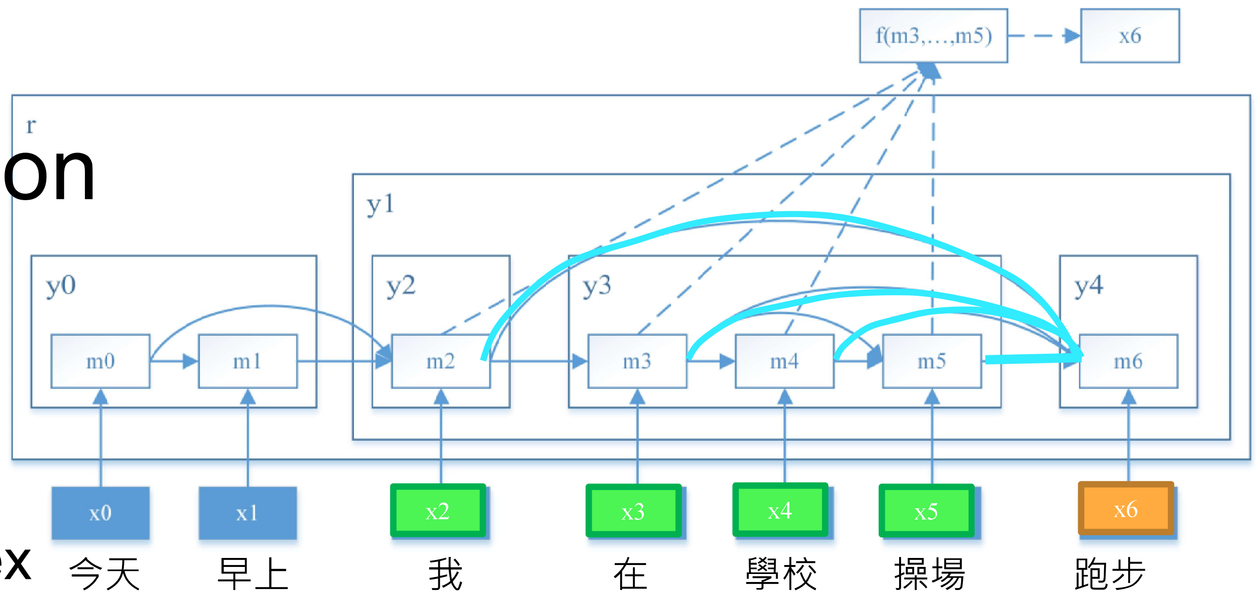
$g_0^6 = 0 \quad g_1^6 = 0 \quad g_2^6 = 1 \quad g_3^6 = 1 \quad g_4^6 = 1 \quad g_5^6 = 1$

- l_t: 所依賴的word中最遠的index
- $$g_i^t = \begin{cases} 1, & l_t \le i < t \\ 0, & 0 < i < l_t \end{cases}$$ **Parsing network**
- $$m_t = h(x_t, m_0, \dots, m_{t-1}, g_0^t, \dots, g_{t-1}^t)$$ **Reading network**
- $$p(x_{t+1} | x_0, \dots, x_t) \approx p(x_{t+1}; f(m_0, \dots, m_t, g_0^{t+1}, \dots, g_t^{t+1}))$$

上面兩件事是分開做的，當你要預測x6的時候，你還沒有m6，只有m5

Discrete skip-connection

用一個 gate function 做 skip-connection，達到上一頁所說的效果



- l_t: 所依賴的word中最遠的index

$l_6 = 2$

$g_0^6 = 0 \quad g_1^6 = 0 \quad g_2^6 = 1 \quad g_3^6 = 1 \quad g_4^6 = 1 \quad g_5^6 = 1$

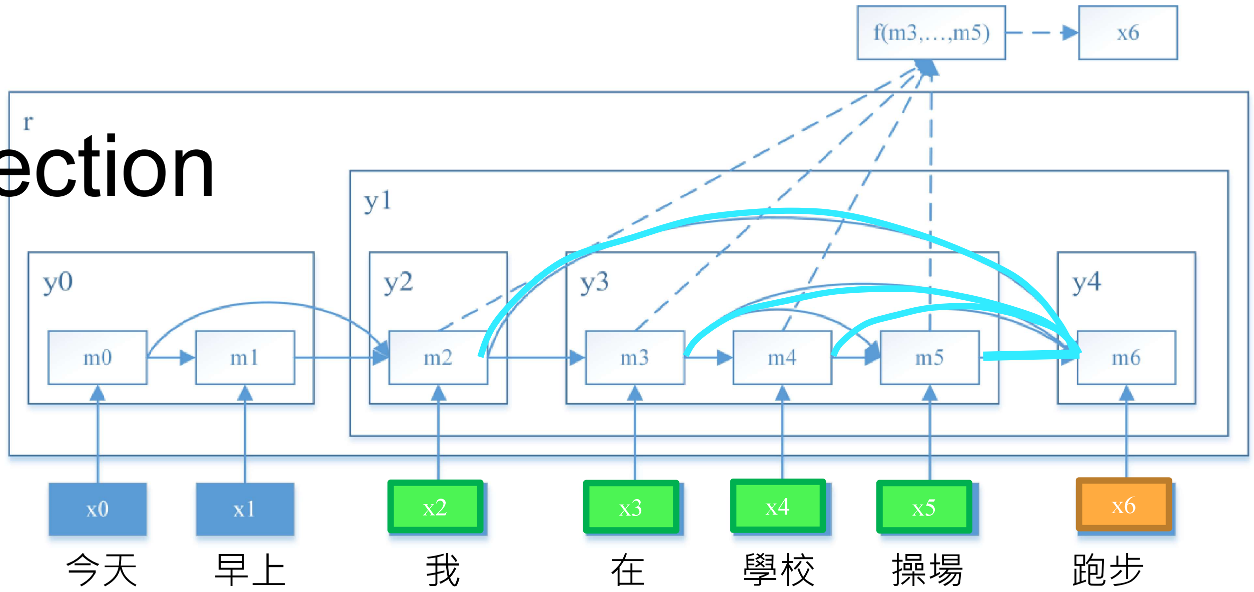
- $$g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 < i < l_t \end{cases}$$ **Parsing network**
- $m_t = h(x_t, m_0, \ldots, m_{t-1}, g_0^t, \ldots, g_{t-1}^t)$ **Reading network**
- $p(x_{t+1} | x_0, \ldots, x_t) \approx p(x_{t+1}; f(m_0, \ldots, m_t, g_0^{t+1}, \ldots, g_t^{t+1}))$ **Predict network**

上面兩件事是分開做的，當你要預測x6的時候，你還沒有m6，只有m5

Continuous skip-connection

l_t變成是機率分布

g_t^i變成l_t的CDF



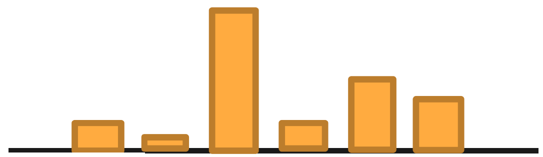
$$p(l_t = i | x_0, ..., x_t) = (1 - \alpha_i^t) \prod_{j=i+1}^{t-1} \alpha_j^t$$

CDF

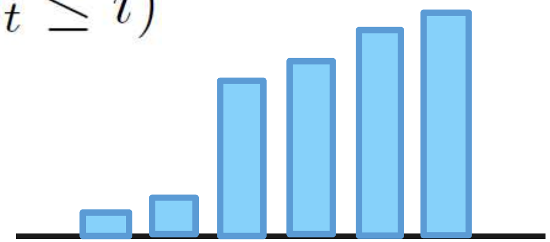
$$g_i^t = \mathbf{P}(l_t \leq i) = \prod_{j=i+1}^{t-1} \alpha_j^t$$

From Dirichlet Process

$$p(l_t = i)$$



$$g_i^t = \mathbf{P}(l_t \leq i)$$



What is α_j^t ?

今天	早上	我	在	學校	操場	跑步
x_0	x_1	x_2	x_3	x_4	x_5	x_6

What is α_j^t ?

今天	早上	我	在	學校	操場	跑步
x_0	x_1	x_2	x_3	x_4	x_5	x_6
	α_0^1	α_0^2	α_0^3	α_0^4		
		α_1^2	α_1^3	α_1^4	$\cdots$	
			α_2^3	α_2^4		
				α_3^4		

What is α_j^t ?

今天	早上	我	在	學校	操場	跑步
x_0	x_1	x_2	x_3	x_4	x_5	x_6
	α_0^1	α_0^2	α_0^3	α_0^4		
		α_1^2	α_1^3	α_1^4	$\cdots$	
			α_2^3	α_2^4		
				α_3^4		

$$p(l_t = i | x_0, ..., x_t) = (1 - \alpha_i^t) \prod_{j=i+1}^{t-1} \alpha_j^t$$

What is α_j^t ?

今天	早上	我	在	學校	操場	跑步
x_0	x_1	x_2	x_3	x_4	x_5	x_6
	α_0^1	α_0^2	α_0^3	α_0^4		
		α_1^2	α_1^3	α_1^4	$\cdots$	
			α_2^3	α_2^4		
				α_3^4		

$$p(l_t = i | x_0, \ldots, x_t) = (1 - \alpha_i^t) \prod_{j=i+1}^{t-1} \alpha_j^t$$

Ex:

$$p(l_4 = 1) =$$

What is α_j^t ?

今天	早上	我	在	學校	操場	跑步
x_0	x_1	x_2	x_3	x_4	x_5	x_6
	α_0^1	α_0^2	α_0^3	α_0^4		
		α_1^2	α_1^3	$(1-\alpha_1^4)$	$\cdots$	
				$\times$		
			α_2^3	α_2^4		
				$\times$		
				α_3^4		

$$p(l_t = i | x_0, \ldots, x_t) = (1 - \alpha_i^t) \prod_{j=i+1}^{t-1} \alpha_j^t$$

Ex:

$$p(l_4 = 1) =$$

What is α_j^t ?

今天	早上	我	在	學校	操場	跑步
x_0	x_1	x_2	x_3	x_4	x_5	x_6
	α_0^1	α_0^2	α_0^3	α_0^4		
		α_1^2	α_1^3	$(1-\alpha_1^4)$	$\cdots$	
				$\times$		
			α_2^3	α_2^4		
				$\times$		
				α_3^4		

$$p(l_t = i | x_0, ..., x_t) = (1 - \alpha_i^t) \prod_{j=i+1}^{t-1} \alpha_j^t$$

Ex:

$$p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$$

What is α_j^t ?

今天	早上	我	在	學校	操場	跑步
x_0	x_1	x_2	x_3	x_4	x_5	x_6
	α_0^1	α_0^2	α_0^3	α_0^4		
		α_1^2	α_1^3	$(1-\alpha_1^4)$ $\times$ α_2^4 $\times$ α_3^4	…	
			α_2^3			

$$p(l_t = i | x_0, ..., x_t) = (1 - \alpha_i^t) \prod_{j=i+1}^{t-1} \alpha_j^t$$

$$g_i^t = \mathbf{P}(l_t \leq i) = \prod_{j=i+1}^{t-1} \alpha_j^t$$

Ex:

$$p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

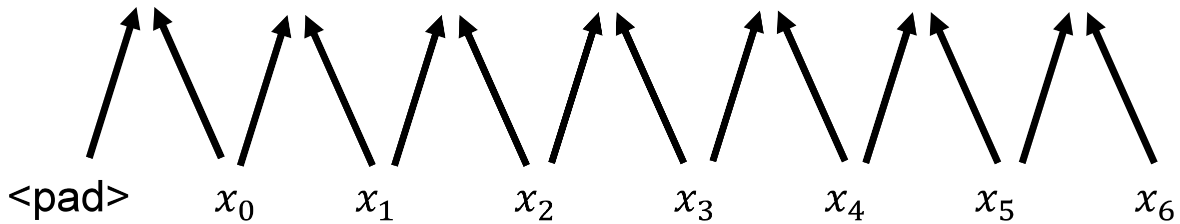
$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

<pad> x_0 x_1 x_2 x_3 x_4 x_5 x_6

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)



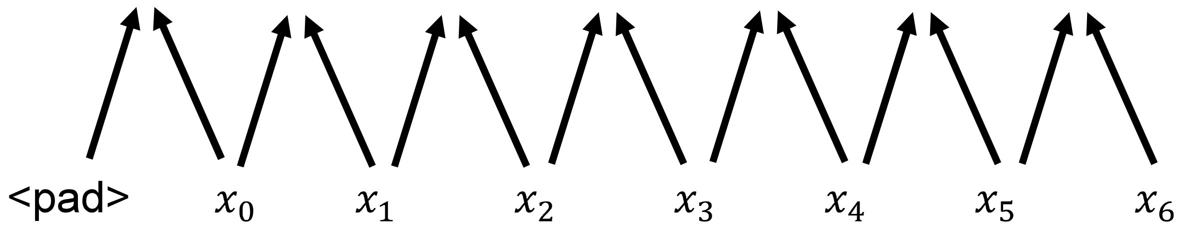
$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \dots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

Syntax distance:

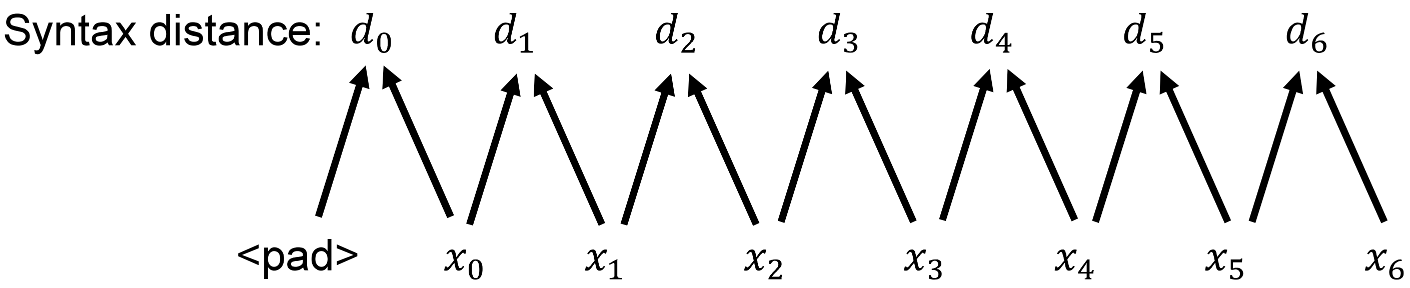


$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)



$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

Syntax distance: d_0 d_1 d_2 d_3 d_4 d_5 d_6

<pad> x_0 x_1 x_2 x_3 x_4 x_5 x_6

$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

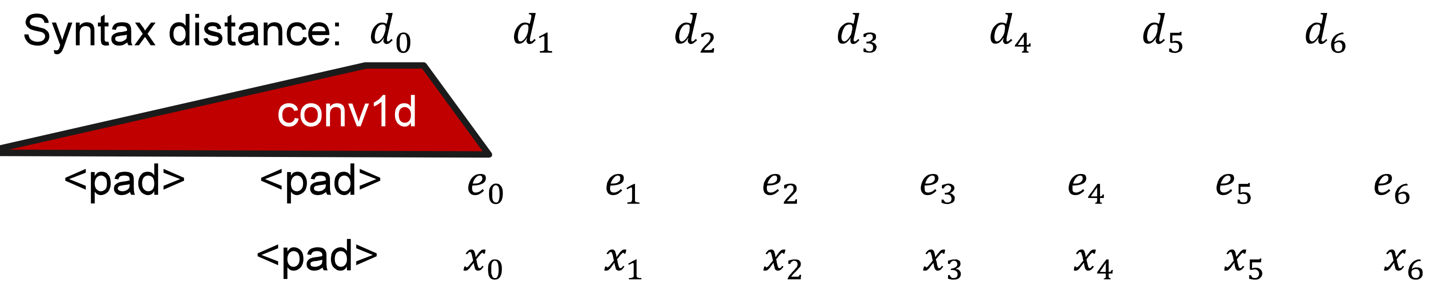
Syntax distance:		d_0	d_1	d_2	d_3	d_4	d_5	d_6
<pad>	<pad>	e_0	e_1	e_2	e_3	e_4	e_5	e_6
	<pad>	x_0	x_1	x_2	x_3	x_4	x_5	x_6

$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

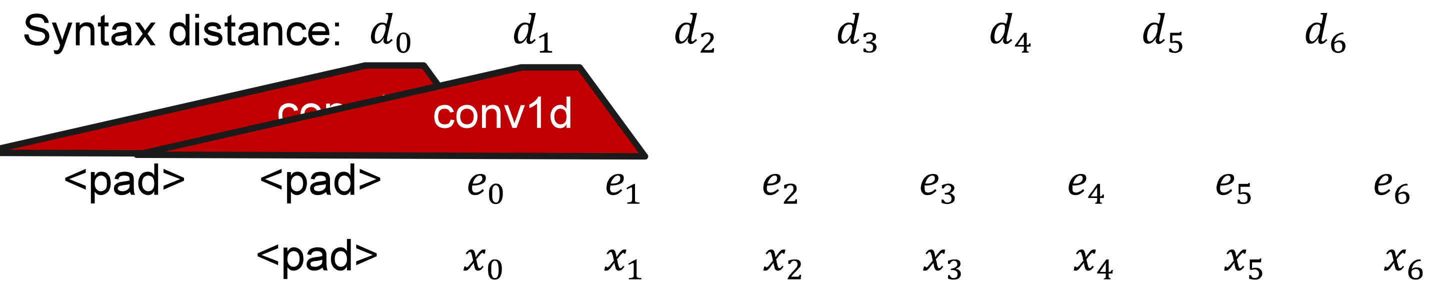


$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \dots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

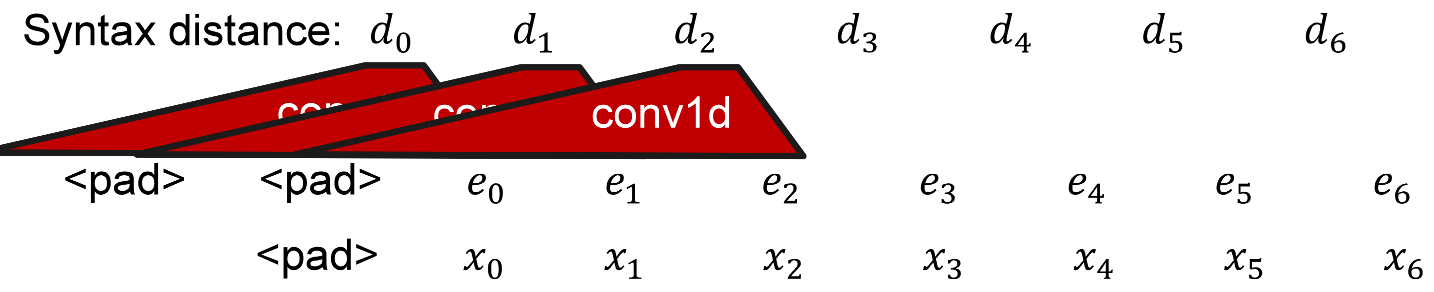


$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

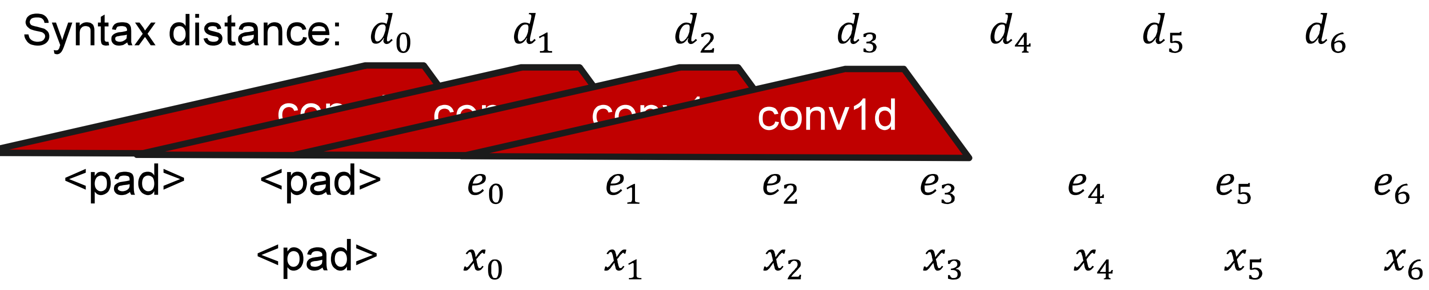


$$h_i = \mathrm{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

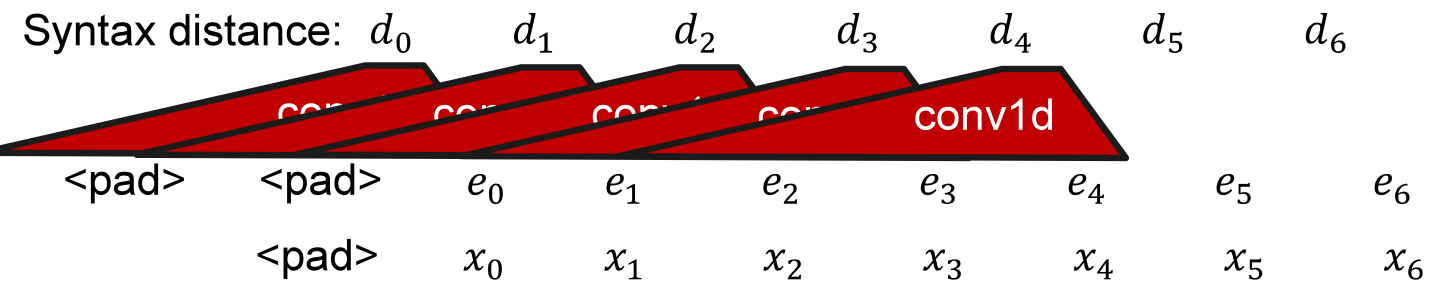


$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

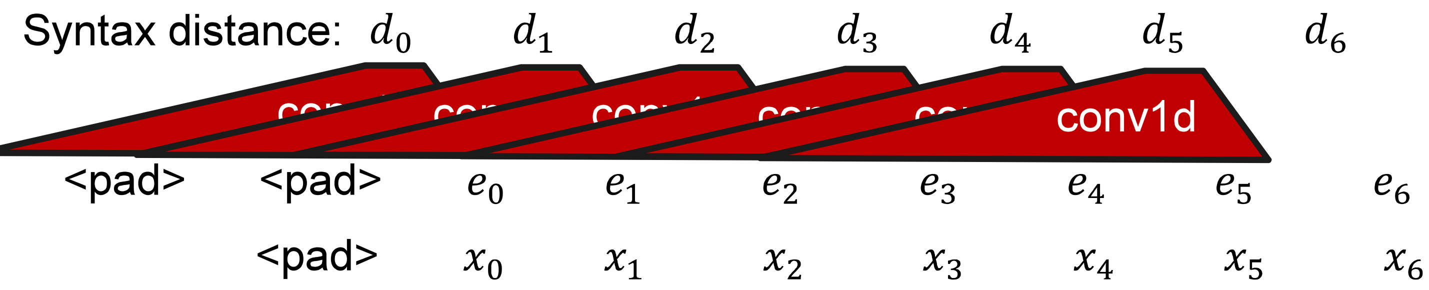


$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

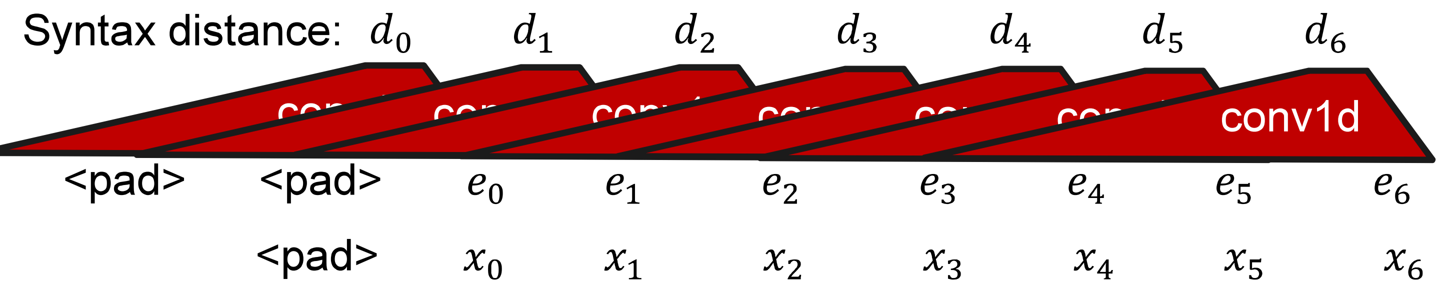


$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)



$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

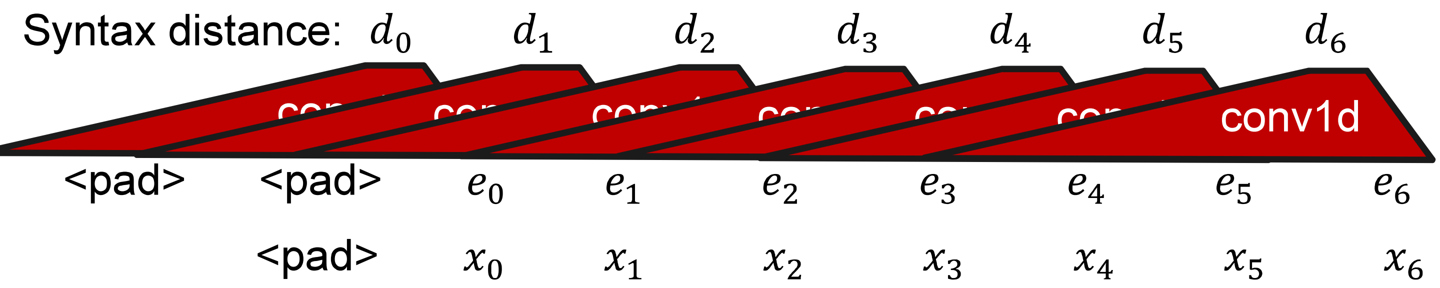
$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\operatorname{hardtanh}\left((d_t - d_j) \cdot \tau\right) + 1}{2}$$

(0~1之間)



$$h_i = \operatorname{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \operatorname{ReLU}\left(W_d h_i + b_d\right)$$

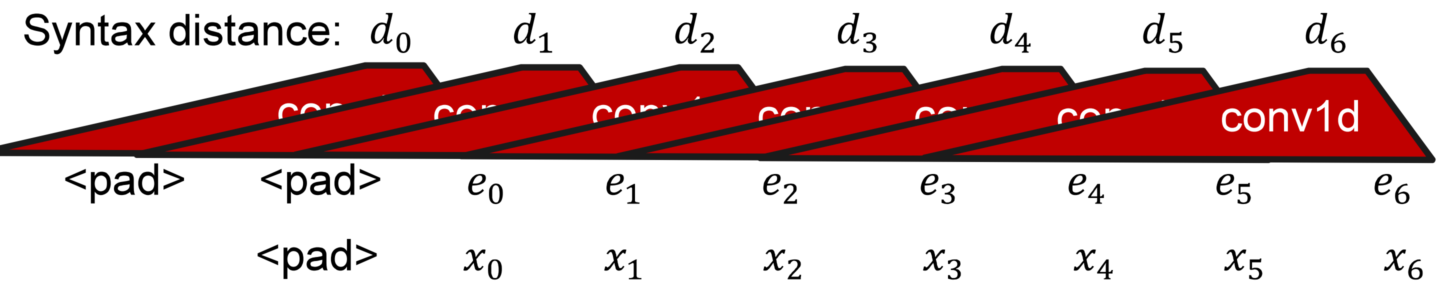
How we get α_j^t ?

Parsing network

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\mathrm{hardtanh}\left((d_t - d_j)\cdot\tau\right) + 1}{2}$$

(0~1之間)



$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

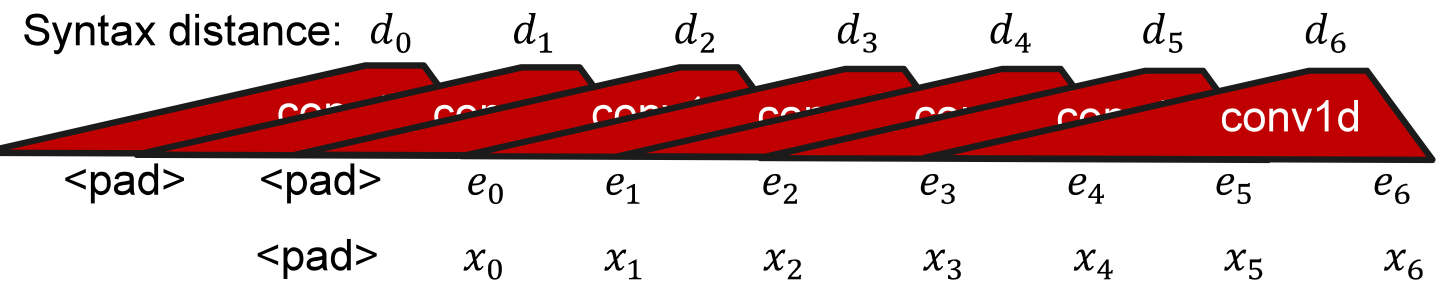
Parsing network

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j)\cdot\tau\right) + 1}{2}$$

(0~1之間)

Ex:



$$h_i = \text{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

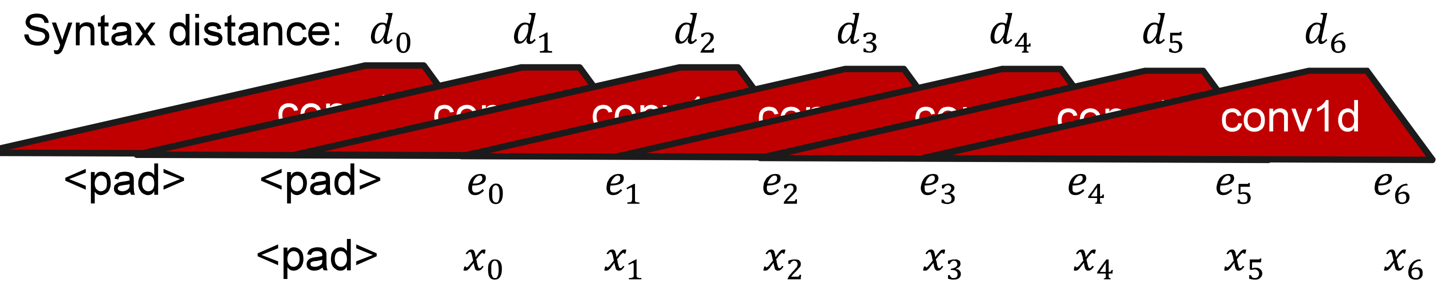
Parsing network

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j) \cdot \tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$



$$h_i = \text{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

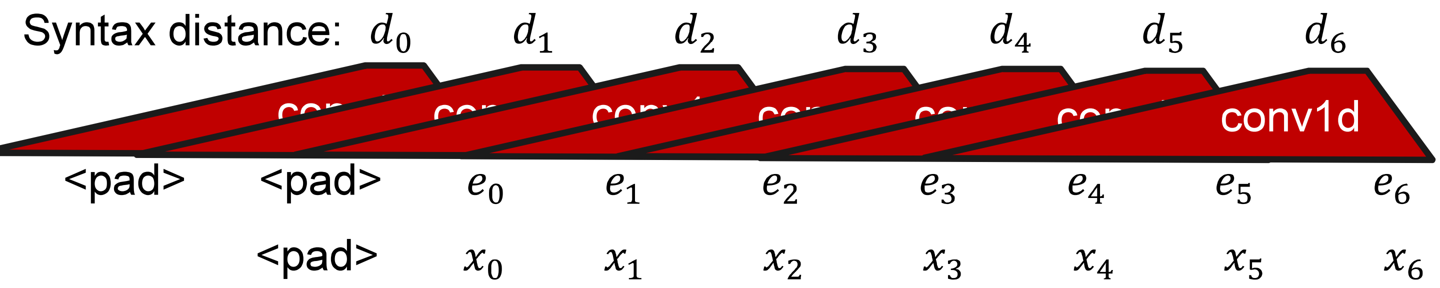
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j) \cdot \tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$

$(1 - \alpha_1^4)\uparrow$



$$h_i = \text{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

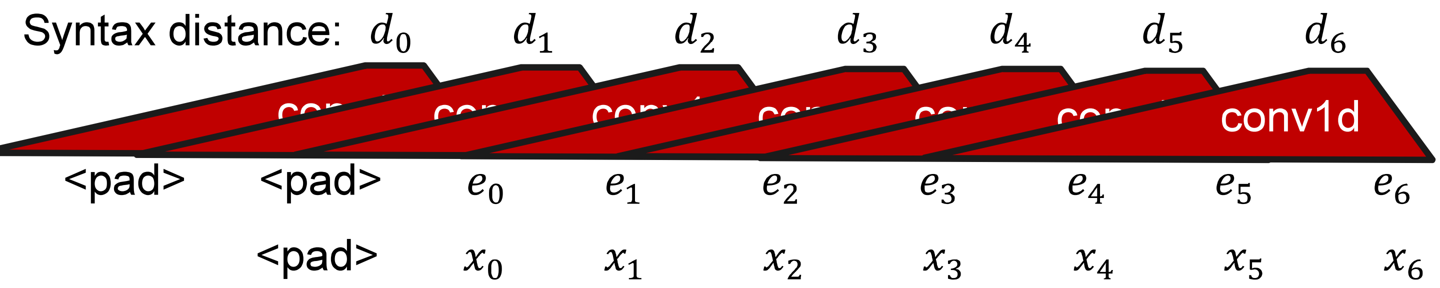
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j) \cdot \tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$

$(1 - \alpha_1^4)\uparrow$



$$h_i = \text{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

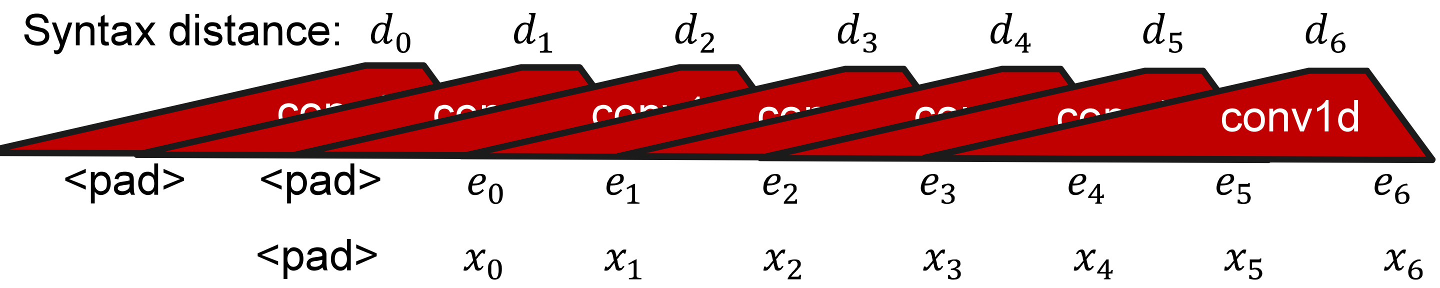
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\mathrm{hardtanh}\left((d_t - d_j) \cdot \tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$

$(1 - \alpha_1^4)\uparrow \Rightarrow \alpha_1^4\downarrow$



$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

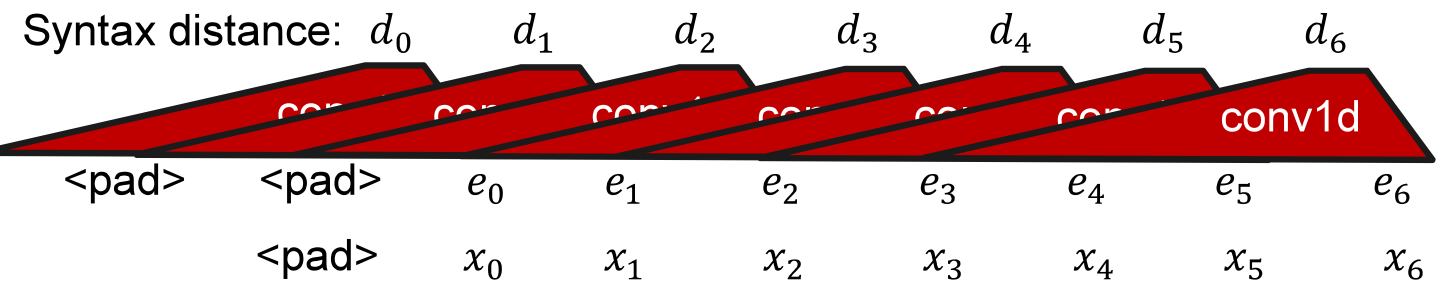
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j) \cdot \tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$

$(1 - \alpha_1^4)\uparrow$ ⇨ $\alpha_1^4\downarrow$ ⇨



$$h_i = \text{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

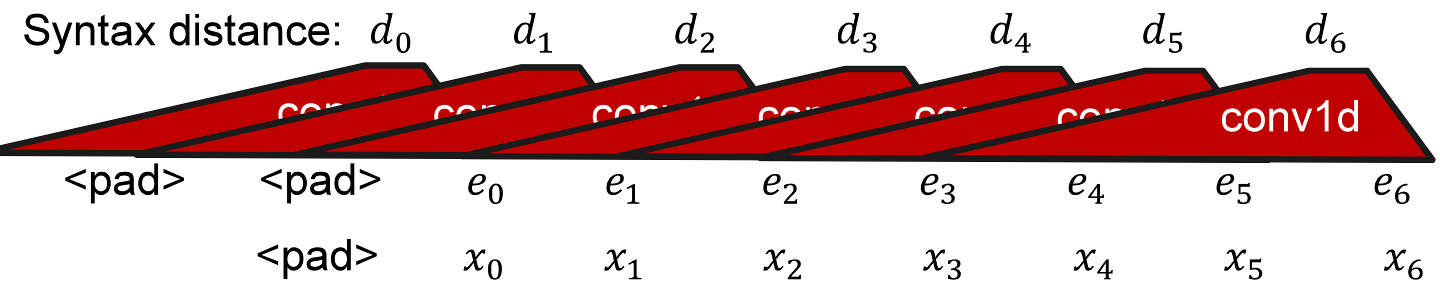
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j)\cdot\tau\right)+1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1-\alpha_1^4)\times\alpha_2^4\times\alpha_3^4$

$(1-\alpha_1^4)\uparrow \Rightarrow \alpha_1^4\downarrow \Rightarrow (d_4 - d_1)\downarrow$



$$h_i = \text{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

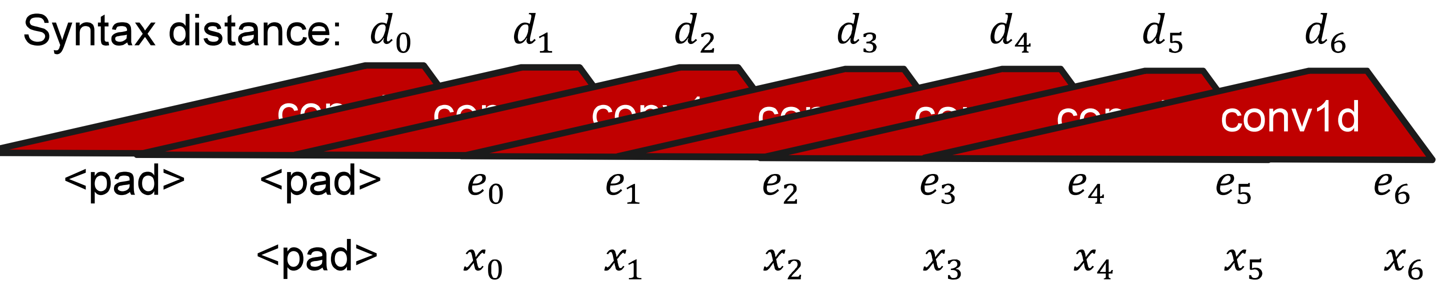
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\mathrm{hardtanh}\left((d_t - d_j)\cdot\tau\right)+1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1-\alpha_1^4)\times\alpha_2^4\times\alpha_3^4$

$(1-\alpha_1^4)\uparrow$ ⇨ $\alpha_1^4\downarrow$ ⇨ $(d_4 - d_1)\downarrow$ ⇨



$$h_i = \mathrm{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \mathrm{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

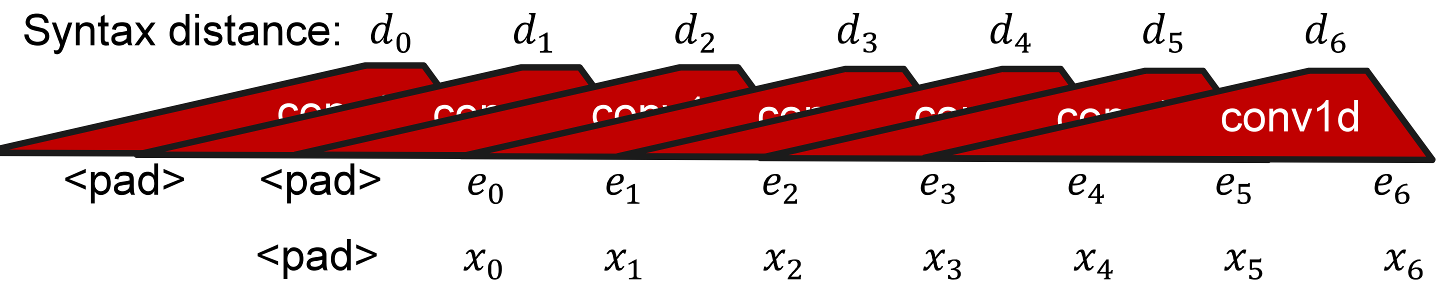
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j)\cdot\tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1-\alpha_1^4)\times\alpha_2^4\times\alpha_3^4$

$(1-\alpha_1^4)\uparrow \Rightarrow \alpha_1^4\downarrow \Rightarrow (d_4 - d_1)\downarrow \Rightarrow d_4\downarrow$ >



$$h_i = \text{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

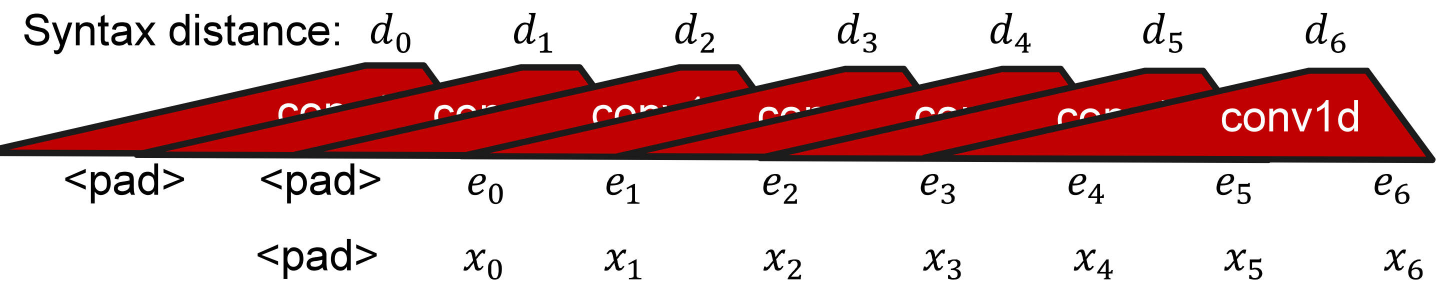
- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j) \cdot \tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1 - \alpha_1^4) \times \alpha_2^4 \times \alpha_3^4$

$(1 - \alpha_1^4)\uparrow \Rightarrow \alpha_1^4\downarrow \Rightarrow (d_4 - d_1)\downarrow \Rightarrow d_4\downarrow > d_1\uparrow$



$$h_i = \text{ReLU}(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

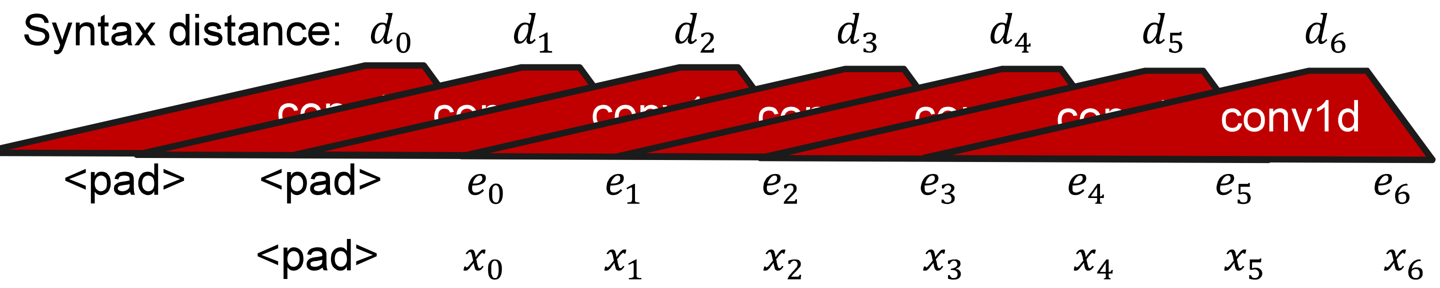
$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j)\cdot\tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1-\alpha_1^4)\times\alpha_2^4\times\alpha_3^4$

$(1-\alpha_1^4)\uparrow \Rightarrow \alpha_1^4\downarrow \Rightarrow (d_4-d_1)\downarrow \Rightarrow d_4\downarrow > d_1\uparrow$

x_3 與 x_4 關聯大



$$h_i = \text{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

How we get α_j^t ?

Parsing network

- Compute a scalar: syntax distance d_i with 1D-Conv (width = L)

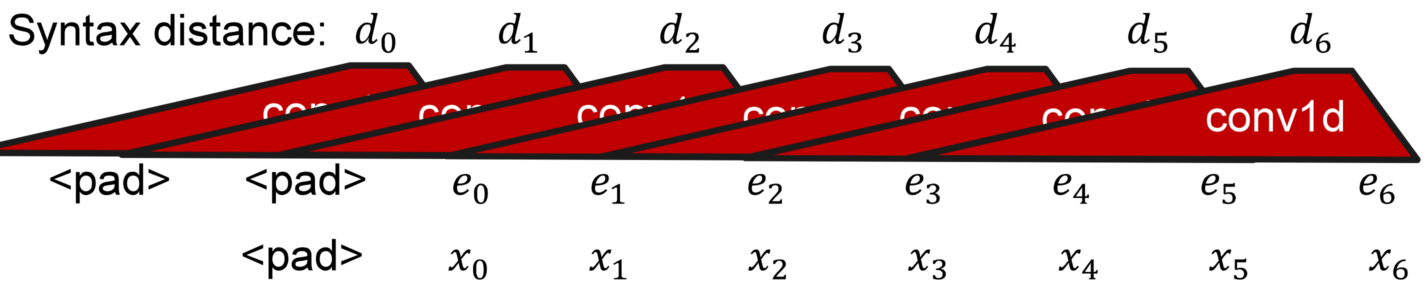
$$\alpha_j^t = \frac{\text{hardtanh}\left((d_t - d_j)\cdot\tau\right) + 1}{2}$$

(0~1之間)

Ex: $p(l_4 = 1) = (1-\alpha_1^4)\times\alpha_2^4\times\alpha_3^4$

$(1-\alpha_1^4)\uparrow \Rightarrow \alpha_1^4\downarrow \Rightarrow (d_4 - d_1)\downarrow \Rightarrow d_4\downarrow > d_1\uparrow$

x_3 與 x_4 關聯大 $\quad$ x_0 與 x_1 關聯小



$$h_i = \text{ReLU}\left(W_c \begin{bmatrix} e_{i-L} \\ e_{i-L+1} \\ \cdots \\ e_i \end{bmatrix} + b_c\right)$$

$$d_i = \text{ReLU}\left(W_d h_i + b_d\right)$$

What does d_i means?

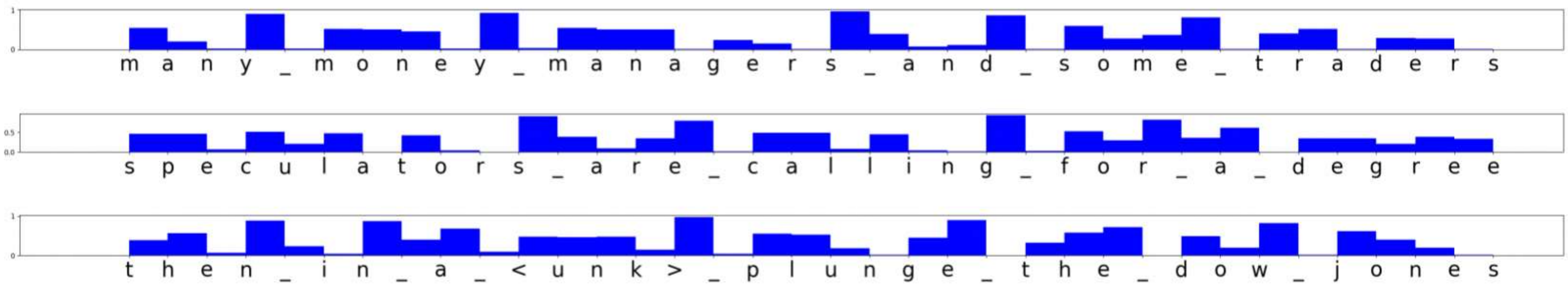


Figure 4: Syntactic distance estimated by Parsing Network. The model is trained on PTB dataset at the character level. Each blue bar is positioned between two characters, and represents the syntactic distance between them. From these distances we can infer a tree structure according to Section 4.2.

What does d_i means?

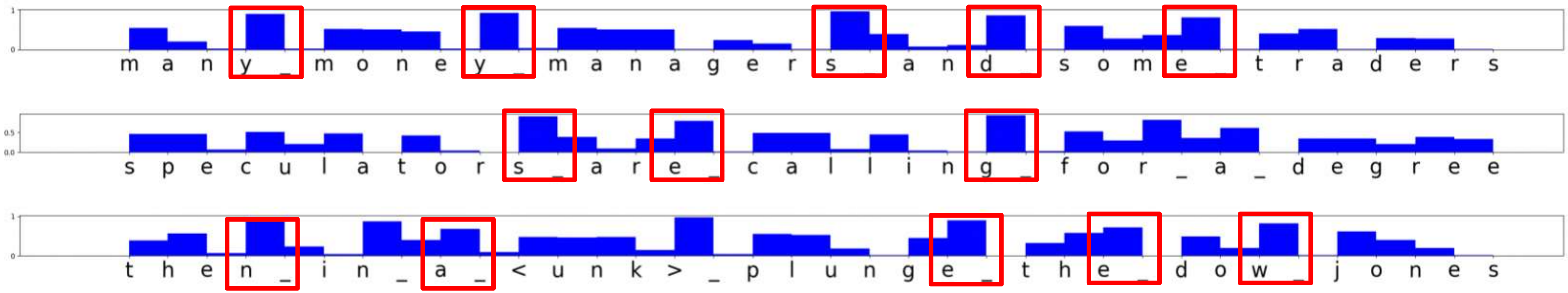


Figure 4: Syntactic distance estimated by Parsing Network. The model is trained on PTB dataset at the character level. Each blue bar is positioned between two characters, and represents the syntactic distance between them. From these distances we can infer a tree structure according to Section 4.2.

What does d_i means?

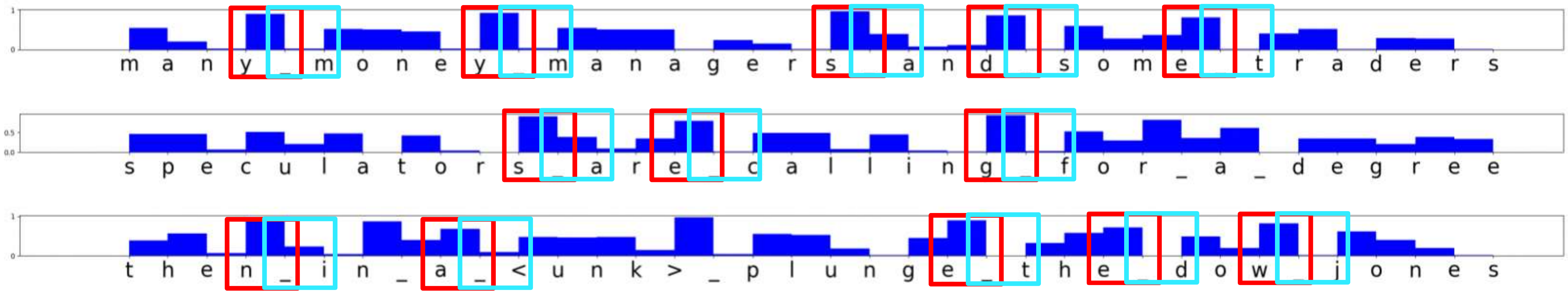


Figure 4: Syntactic distance estimated by Parsing Network. The model is trained on PTB dataset at the character level. Each blue bar is positioned between two characters, and represents the syntactic distance between them. From these distances we can infer a tree structure according to Section 4.2.

Reading network (Structure Attention + Recurrent)

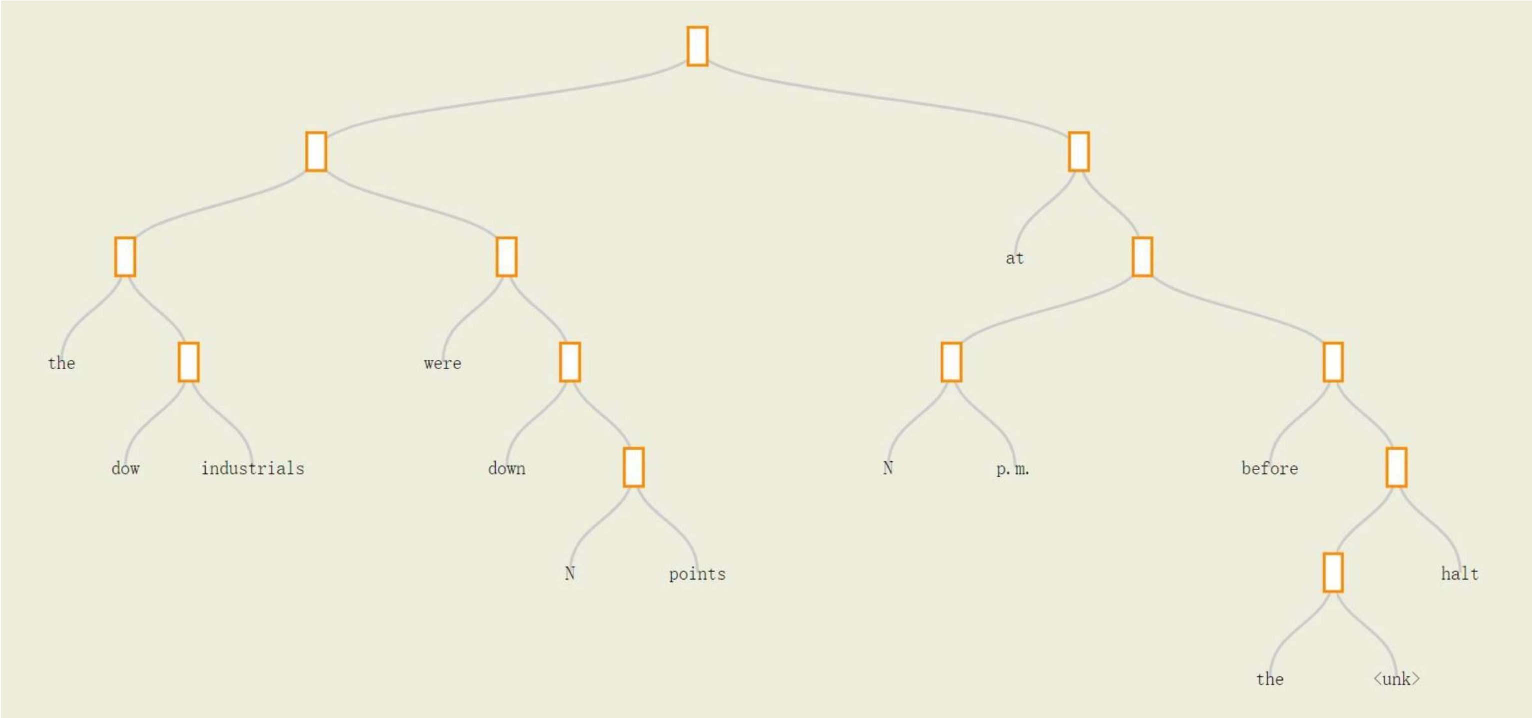
- 跟LSTM 一樣有 h_t, c_t
- 但吃的input不是上一個time step的hidden，而是經過算attention再加權
- 算key: $k_t = W_h h_{t-1} + W_x x_t$ (只取決於比較有local資訊的 x_t, h_{t-1})
- 算 attention weight: $\tilde{s}_i^t = \mathrm{softmax}(\frac{h_i k_t^{\mathrm{T}}}{\sqrt{\delta_k}})$ δ_k is the dimension of the hidden state.
- 把 attention weight 拿來做 gated: $$s_i^t = \frac{g_i^t \tilde{s}_i^t}{\sum_i g_i^t}$$
- 算出要現在要吃的input: $$\begin{bmatrix} \tilde{h}_t \\ \tilde{c}_t \end{bmatrix} = \sum_{i=1}^{t-1} s_i^t \cdot m_i = \sum_{i=1}^{t-1} s_i^t \cdot \begin{bmatrix} h_i \\ c_i \end{bmatrix}$$
- 有了 x_t, $\tilde{c}_t$ and $\tilde{h}_t$ ，後面就跟LSTM一樣了

Predict Network

- 現在要預測 x_{t+1}，input 是 $m_0, \dots, m_t$ 和 $g_0^{t+1}, \dots, g_t^{t+1}$
- But ，要算 $g_0^{t+1}, \dots, g_t^{t+1}$，需要$d_{t+1}$，算$d_{t+1}$需要$x_{t+1}$，我們還沒有$x_{t+1}$
- 先用自己 predict 的: $$d'_{t+1} = \mathrm{ReLU}(W'_d h_t + b'_d)$$
- 就可以算出所有 $\{\alpha^{t+1}\}$ and $\{g_i^{t+1}\}$ (parsing network)
- 再針對$h_{l_{t+1}}, \dots, h_{t-1}$跑一次剛剛的structure attention($g_0^{t+1}, \dots, g_{t-1}^{t+1}$)，得到$h_{l:t-1}$
- Predict: $$f(m_0, \dots, m_t, g_0^{t+1}, \dots, g_t^{t+1}) = \hat{f}([h_{l:t-1}, h_t])$$
- $\hat{f}(\cdot)$ could be a simple feed-forward MLP, or more complex architecture, like ResNet

How to get parsing results?

- 依照 d_i 順序，由大到小split句子，直到不能再切
- 三個children的狀況可能就無法與正確答案一致，吃大虧



Experiments

- character-level language model
- Penn Treebank

Model	BPC
Norm-stabilized RNN (Krueger & Memisevic, 2015)	1.48
CW-RNN (Koutnik et al., 2014)	1.46
HF-MRNN (Mikolov et al., 2012)	1.41
MI-RNN (Wu et al., 2016)	1.39
ME n-gram (Mikolov et al., 2012)	1.37
BatchNorm LSTM (Cooijmans et al., 2016)	1.32
Zoneout RNN (Krueger et al., 2016)	1.27
HyperNetworks (Ha et al., 2016)	1.27
LayerNorm HM-LSTM (Chung et al., 2016)	1.24
LayerNorm HyperNetworks (Ha et al., 2016)	1.23
PRPN	**1.202**

Table 1: BPC on the Penn Treebank test set

Experiments

- word-level language model
- Penn Treebank

Model	PPL
RNN-LDA + KN-5 + cache (Mikolov & Zweig, 2012)	92.0
LSTM (Zaremba et al., 2014)	78.4
Variational LSTM (Kim et al., 2016)	78.9
CharCNN (Kim et al., 2016)	78.9
Pointer Sentinel-LSTM (Merity et al., 2016)	70.9
LSTM + continuous cache pointer (Grave et al., 2016)	72.1
Variational LSTM (tied) + augmented loss (Inan et al., 2016)	68.5
Variational RHN (tied) (Zilly et al., 2016)	65.4
NAS Cell (tied) (Zoph & Le, 2016)	62.4
4-layer skip connection LSTM (tied) (Melis et al., 2017)	**58.3**
PRPN	61.98

Table 2: PPL on the Penn Treebank test set

Experiments

- ablation test

Model	PPL
PRPN	61.98
- Parsing Net	64.42
- Reading Net Attention	64.63
- Predict Net Attention	63.65
Our 2-layer LSTM	65.81

Table 3: Ablation test on the Penn Treebank.

→ 原始的
→ 把Parsing net 改成一般的Attention
→ Reading net 不用SA，改成LSTM
→ Predict net 不用SA，改成LSTM
→ 完全不用Parsing net 的 SA，退化成 LSTM

Experiments

- word-level language model
- Text8

Model	PPL
LSTM-500 (Mikolov et al., 2014)	156
SCRNN (Mikolov et al., 2014)	161
MemNN (Sukhbaatar et al., 2015)	147
LSTM-1024 (Grave et al., 2016)	121
LSTM + continuous cache pointer (Grave et al., 2016)	99.9
PRPN	**81.64**

Table 4: PPL on the Text8 valid set

Experiments

- unsupervised constituency parsing
- WSJ10
- Unlabeled F1

Model	UF_1	
LBRANCH	28.7	→ 猜 pure left tree
RANDOM	34.7	→ 猜 random binary tree
DEP-PCFG (Carroll & Charniak, 1992)	48.2	→ unsupervised dependency
RBRANCH	61.7	→ 猜 pure right tree
CCM (Klein & Manning, 2002)	71.9	→ constituent-context model
DMV+CCM (Klein & Manning, 2005)	77.6	→ unsupervised dependency
UML-DOP (Bod, 2006)	**82.9**	→ cons. + dep. joint model
PRPN	70.02	
UPPER BOUND	88.1	→ binary tree 能近似的極限

Table 5: Parsing Performance on the WSJ10 dataset

Interesting story…(1)

- Kyunghyun Cho 真的很嚴格…

not fully unsupervised parsing

Kyunghyun Cho

05 Apr 2018 ICLR 2018 Conference Paper679 Public Comment Readers: Everyone

Comment: thanks to the authors for promptly releasing the code public!

one thing i noticed from the released code is that early stopping for the experiments with sentence-level language modelling on penn treebank is done based on the F-1 score using the gold standard annotations:

https://github.com/yikangshen/PRPN/blob/master/main_UP.py#L228-L237

i believe this makes it less of unsupervised learning, and i couldn't find this setup mentioned in the paper. i kindly ask the authors to reflect this in the text to avoid misleading any reader.

Early stopping is not necessary

ICLR 2018 Conference Paper679 Authors

11 Apr 2018 ICLR 2018 Conference Paper679 Official Comment Readers: Everyone

Comment: Thanks for pointing this out!
We reran the experiment and found out that the early stopping by monitoring F1 is not necessary for unsupervised parsing task. As training loss decrease, the F1 almost always increase. Thus, we updated the code to use training loss as learning rate schedule and early stopping criteria:
https://github.com/yikangshen/PRPN/blob/a1a8431499918a99f4689dca9428018ca2e256d9/main_UP.py#L229-L242

Interesting story…(2)

- Yoon Kim 也 fork

yoonkim forked yoonkim/PRPN from yikangshen/PRPN on 20 Sep

yikangshen/PRPN

Parsing Reading Predict Network

Python ★ 35 Updated Oct 16

★ Unstar

Interesting story…(3)

- Yikang Shen, Zhouhan Lin, Chin-wei Huang, Aaron Courville
- 轉領域強者校友

Université de Montréal
Doctor of Philosophy - PhD, Computer Science
2017 年

Université de Montréal - École Polytechnique de Montréal
Doctor of Philosophy (Ph.D.), Mechanical Engineering
2016 年 – 2017 年

National Taiwan University
Bachelor' s Degree, Chemical Engineering
2011 年 – 2015 年
活動和社團： NTU Guitar Club 40th-term President

Interesting story…(4)

- 宗男又被cite了 Orz

被引用 451 次

Apart from the approach of using recursive networks to capture structures, there is another line of research which try to learn recurrent features at multiple scales, which can be dated back to 1990s (e.g. El Hihi & Bengio (1996); Schmidhuber (1991); Lin et al. (1998)). The NARX RNN (Lin et al., 1998) is another example which used a feed forward net taking different inputs with predefined time delays to model long-term dependencies. More recently, Koutnik et al. (2014) also used multiple layers of recurrent networks with different pre-defined updating frequencies. Instead, our model tries to learn the structure from data, rather than predefining it. In that respect, Chung et al. (2016) relates to our model since it proposes a hierarchical multi-scale structure with binary gates controlling intra-layer connections, and the gating mechanism is learned from data too. The difference is that their gating mechanism controls the updates of higher layers directly, while ours control it softly through an attention mechanism.