Recurrent Neural Networks

Notation

x:	Harry	Potter	and	Herminone	Granger	invented	a	new	spell
	$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	$x^{<5>}$	$x^{<6>}$	$x^{<7>}$	$x^{<8>}$	$x^{<9>}$

$T_x = 9$: x의 길이

y:									
	$y^{<1>}$	$y^{<2>}$	$y^{<3>}$	$y^{<4>}$	$y^{<5>}$	$y^{<6>}$	$y^{<7>}$	$y^{<8>}$	$y^{<9>}$

$T_y = 9$: y의 길이

$x^{(i)<t>}$	i번째 데이터의 t번째 단어	$T_x^{(i)}$	i번째 x의 길이
$y^{(i)<t>}$	i번째 데이터의 t번째 단어에 대한 prediction	$T_y^{(i)}$	i번째 y의 길이

Vocabulary: 알고 있는 단어 사전

단어를 입력받고 one-hot encoding으로 vocabulary의 단어와 연결

〈UNK〉: Unkown. 모르는 단어

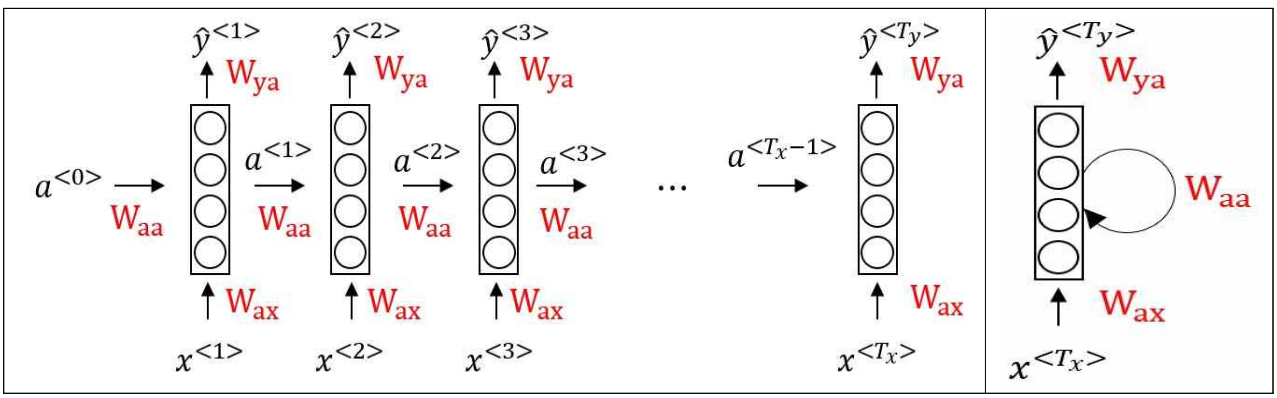
〈EOS〉: End of Sentence. 문장 끝

Why not a standard network?

Problems ⇒

1. Inputs, outputs can be different lengths in different examples.
2. Doesn't share features learned across different positions of text.

Recurrent Neural Networks(RNN)



$a^{<0>}$: vector of zeros

$\hat{y}^{<t>}$를 만들 때 $x^{<t>}$ 뿐만 아니라 이전 데이터의 정보도 얻는다. 그러나 앞서 나온 정보만

이용한다는 단점이 있다.

ex) He said, "Teddy Roosevelt was a great President."

He said, "Teddy bears are on sale!"

Teddy라는 단어를 볼 때 앞서 나온 단어만 알고 있으므로 Teddy가 의미하는 것이 Teddy Roosevelt인지 Teddy bear인지 알 수 없다.

Forward Propagation

$$a^{<0>} = \vec{0}, \quad a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$y^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

$$\Downarrow$$

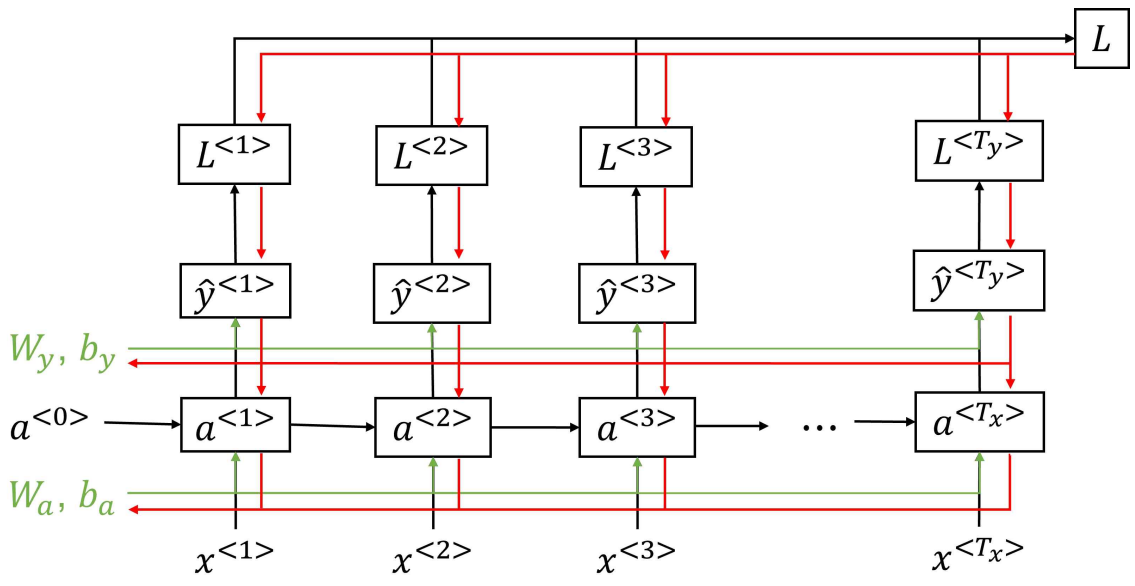
$$[W_{aa} | W_{ax}] = W_a, \quad [a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$$

$$\Rightarrow \quad [W_{aa} | W_{ax}] \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_{aa}a^{<t-1>} + W_{ax}x^{<t>}$$

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

$$y^{<t>} = g(W_y a^{<t>} + b_y)$$

Forward Propagation and Backpropagation

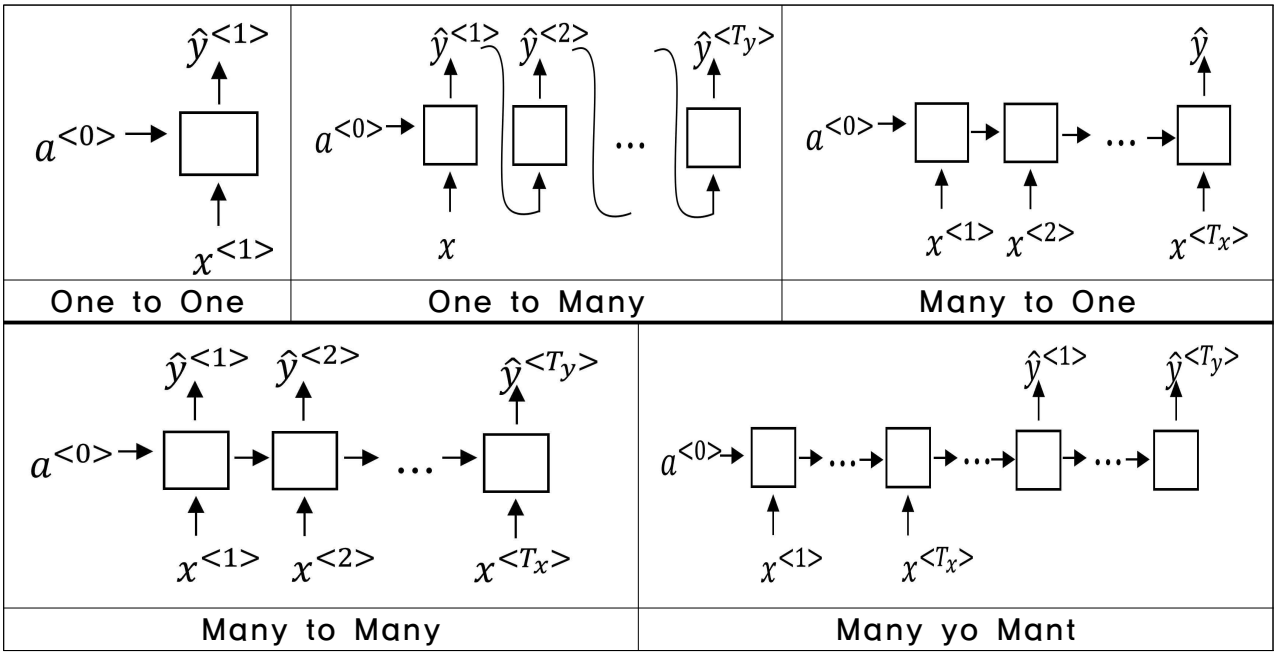


$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>}\log\hat{y}^{<t>} - (1-y^{<t>})\log(1-\hat{y}^{<t>})$$

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Backpropagation through time

Types of RNN



Language model

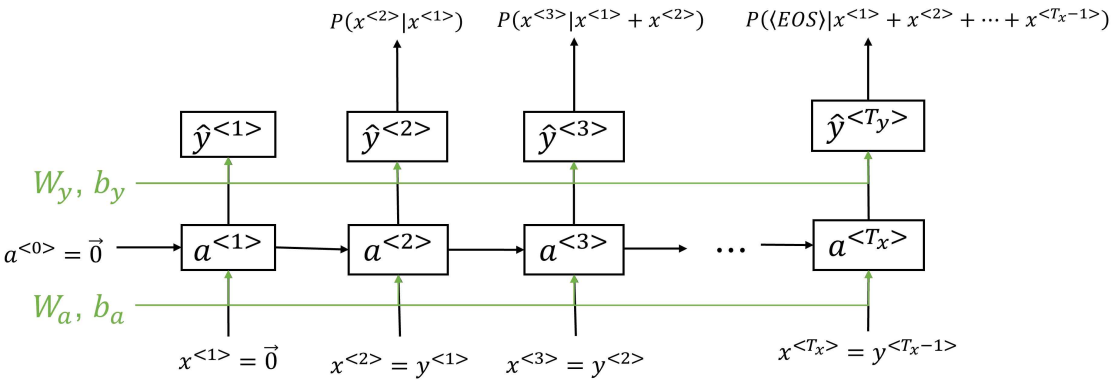
Speech recognition

1. "The apple and pair salad."
   or "The apple and pear salad."
2. P(The apple and pair salad)
   and P(The apple and pear salad)
3. P(sentence) = ?

Language modeling with an RNN

1. Tokenize
2. one-hot encoding
3. ⟨EOS⟩: End of Sentence

※ 만약 Vocabulary에 없는(알지 못하는) 단어가 있으면 ⟨UNK⟩로 처리

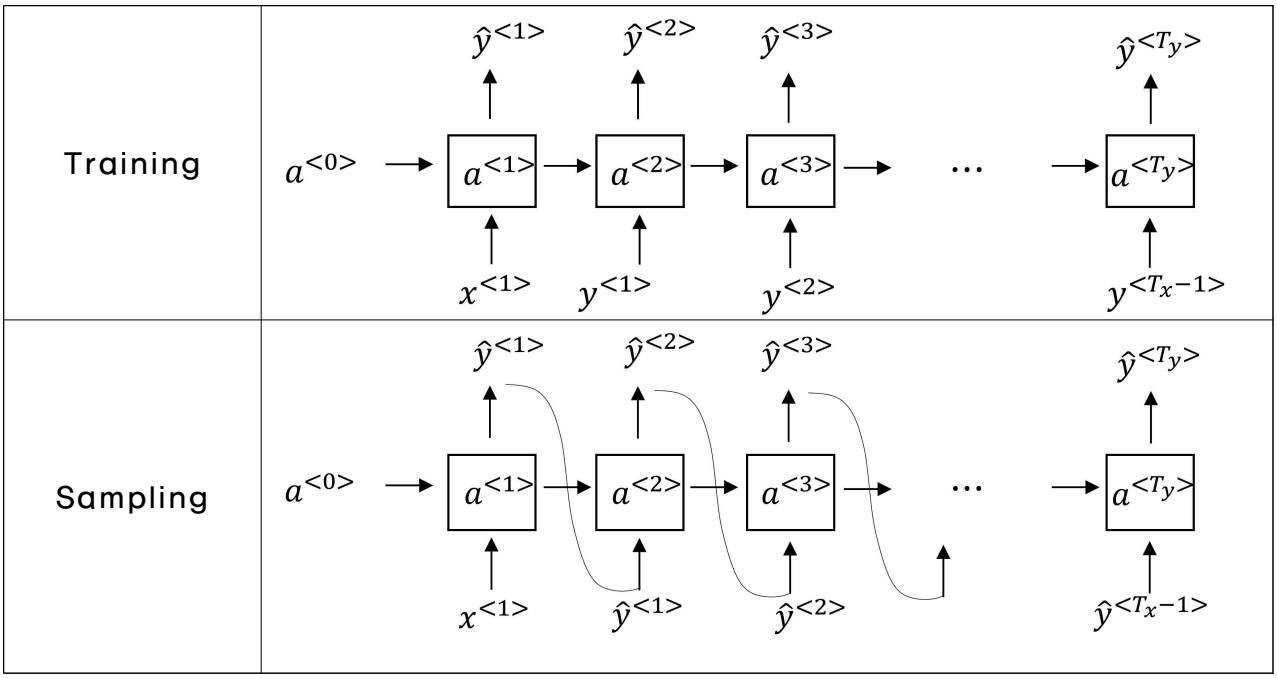


$$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>})P(y^{<2>}|y^{<1>})P(y^{<3>}|y^{<1>}, y^{<2>})$$

Sampling novel sequences

Sampling a sequence from a trained RNN

sequence model은 특정 단어의 시퀀스를 모델링한다.



〈UNK〉을 얻지 않으려면 모르는 단어는 무시하고 아는 단어가 나올 때까지 나머지 단어의 리샘플링한다. 〈UNK〉이 상관없으면 모르는 단어를 아웃풋에 그냥 놔둬도 된다.

Word-level language model(단어 수준)
vs. Character-level language model(문자 수준)

Character-level language model의

장점

1. 모르는 단어에 대해 걱정할 필요가 없다. 그저 알파벳의 시퀀스로 처리한다.

단점

1. 더 긴 시퀀스로 끝난다.
2. 더 많은 계산 비용이 든다.

Vanishing gradient with RNNs

RNN은 너무 멀리 떨어져 있는 두 단어에 대해 처리를 잘 못 한다. Vanishing gradient 때문이다. 그래서 근처에 있는 단어들끼리 유사한 특성을 보인다.

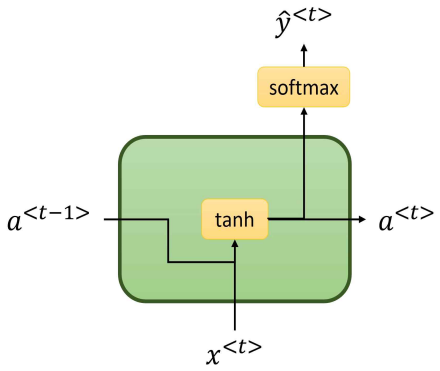
Exploding gradient가 발생하면 numerical overflow가 발생해서 Nan과 같은 에러가 뚜렷하게 보이고, Exploding gradient는 gradient clipping으로 해결할 수 있다.

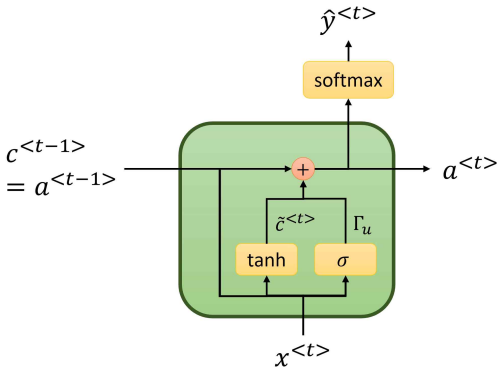
따라서 RNN의 주요 문제는 Vanishing gradient이다.

Gated Recurrent Unit(GRU)

much better capture long range connection

and help a lot with the vanishing gradient problems

RNN unit	
	$a^{<t>} = g\left(W_a\left[a^{<t-1>}, x^{<t>}\right] + b_a\right)$ $= \tanh\left(W_a\left[a^{<t-1>}, x^{<t>}\right] + b_a\right)$

GRU unit	
	c: memory cell GRU에서는 $a^{<t>} = c^{<t>}$ $\tilde{c}^{<t>} = \tanh\left(W_c\left[c^{<t-1>}, x^{<t>}\right] + b_c\right)$ $\Gamma_u = \sigma\left(W_u\left[c^{<t-1>}, x^{<t>}\right] + b_u\right)$ $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$

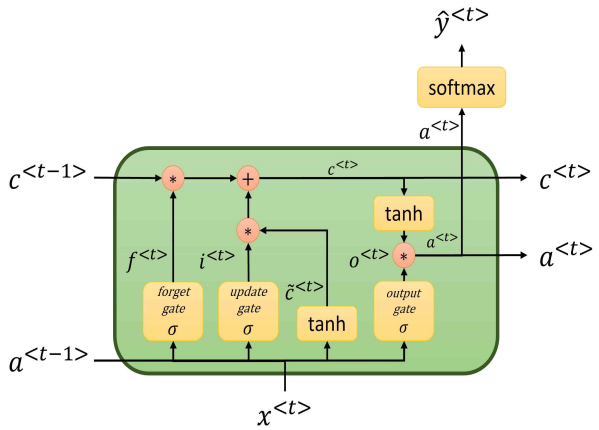
[Cho et al., 2014.
On the properties of neural machine translation: Encoder–decoder approaches]
[Chung et al., 2014.
Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling]

식 $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$으로 gradient vanishing 문제를 해결했다. Γ가 0에 가까울수록 $c^{<t>} = c^{<t-1>}$이 상당히 오랜 시간 유지된다.

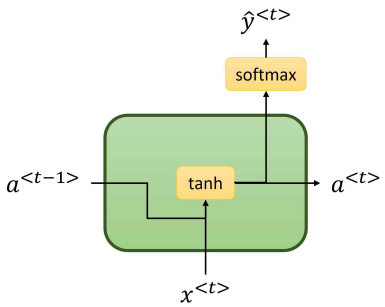
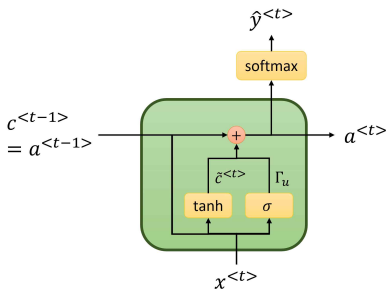
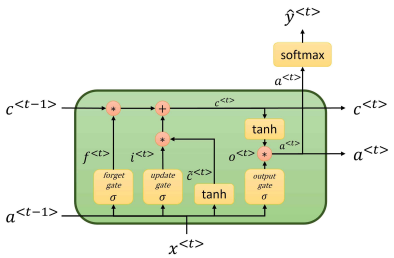
Full GRU	
$\tilde{h}$	$\tilde{c}^{<t>} = \tanh\left(W_c\left[c^{<t-1>}, x^{<t>}\right] + b_c\right)$ or $\tilde{c}^{<t>} = \tanh\left(W_c\left[\Gamma_r * c^{<t-1>}, x^{<t>}\right] + b_c\right)$
u	$\Gamma_u = \sigma\left(W_u\left[c^{<t-1>}, x^{<t>}\right] + b_u\right)$
r	$\Gamma_r = \sigma\left(W_r\left[c^{<t-1>}, x^{<t>}\right] + b_r\right)$
h	$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$

LSTM(long short term memory) unit

GRU	LSTM
$\tilde{c}^{<t>} = \tanh\left(W_c\left[\Gamma_r * c^{<t-1>}, x^{<t>} \right] + b_c \right)$ $\Gamma_u = \sigma\left(W_u\left[c^{<t-1>}, x^{<t>} \right] + b_u \right)$ $\Gamma_r = \sigma\left(W_r\left[c^{<t-1>}, x^{<t>} \right] + b_r \right)$ $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$ $a^{<t>} = c^{<t>}$	$\tilde{c}^{<t>} = \tanh\left(W_c\left[\Gamma_r * c^{<t-1>}, x^{<t>} \right] + b_c \right)$ $\Gamma_u = \sigma\left(W_u\left[c^{<t-1>}, x^{<t>} \right] + b_u \right)$ $\Gamma_f = \sigma\left(W_f\left[c^{<t-1>}, x^{<t>} \right] + b_f \right)$ $\Gamma_o = \sigma\left(W_o\left[c^{<t-1>}, x^{<t>} \right] + b_o \right)$ $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$ $a^{<t>} = \Gamma_o * \tanh c^{<t>}$

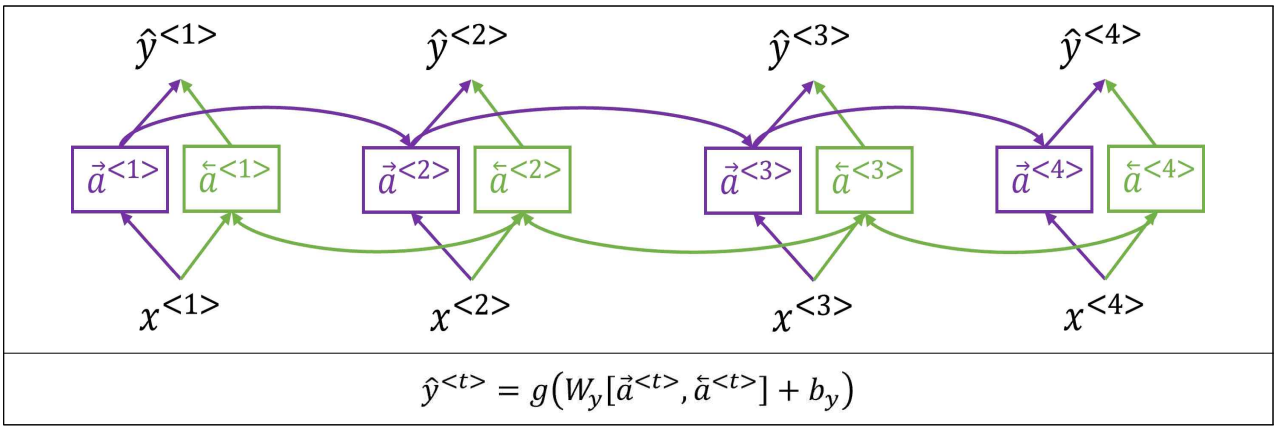
LSTM unit	
	$\tilde{c}^{<t>} = \tanh\left(W_c\left[\Gamma_r * c^{<t-1>}, x^{<t>} \right] + b_c \right)$ $\Gamma_u = \sigma\left(W_u\left[c^{<t-1>}, x^{<t>} \right] + b_u \right)$ $\Gamma_f = \sigma\left(W_f\left[c^{<t-1>}, x^{<t>} \right] + b_f \right)$ $\Gamma_o = \sigma\left(W_o\left[c^{<t-1>}, x^{<t>} \right] + b_o \right)$ $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$ $a^{<t>} = \Gamma_o * \tanh c^{<t>}$

peephole connection

RNN unit	GRU unit	LSTM unit
		

Bidirectional RNN

Getting informatinofrom the future



$$\hat{y}^{<t>} = g\left(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y\right)$$

Deep RNNs