

3. Recurrent Neural Networks

Minsung Kim

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

Intro

- 이번주는 NLP에 대해 배워봅시다
- 말로만 듣던 ‘attention’, ‘Transformer’가 뭐지... ‘attention’만 있으면 된다던데
- 딥러닝의 대표적인 분야 중 하나지만 처음 들으면 많이 어려울 수 있어요..!
 - D2D 목적에 맞게 어려운 내용은 최대한 배제하고 NLP가 어떤 느낌인지 알려주기 위해 열심히 준비했지만 그래도 질문이 있으면 언제든지 연락주세요!

Contents

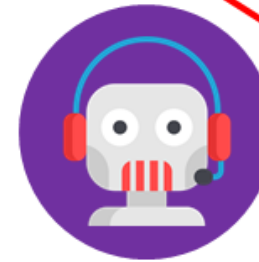
- Word Vector
- Language Modeling
- RNNs
 - Basic Concept
 - Various tasks using RNNs
 - LSTMs, Bidirectional RNNs, Multi-layer RNNs

Word Vector

Can Computers Understand Language?



국어 = 80점
영어 = 85점
수학 = 78점
평균(국어, 영어, 수학)



사전	
평균	000101010101101 010101001010101 010101010110100 101010101010100
...	...
...	...
...	...

이미지 출처: <https://m.blog.naver.com/stupidon9876/221485632342>

How do we represent the meaning of a word?

- In Traditional NLP, **one-hot vectors**
 - 개 = $[1, 0, 0, 0]$
 - 고양이 = $[0, 1, 0, 0]$
 - 선생님 = $[0, 0, 1, 0]$
 - 학생 = $[0, 0, 0, 1]$
- 이렇게 표현했을 때의 문제점은 무엇일까?

How do we represent the meaning of a word?

- 하지만 이 세상에 존재하는 단어의 개수는?
 - 벡터의 차원 = 단어의 개수
- 뜻이 비슷한 단어는?
 - 개 = $[1, 0, 0, 0, 0]$
 - 강아지 = $[0, 0, 0, 0, 1]$
 - 두 개의 벡터는 수직(orthogonal) \rightarrow Cosine Similarity = 0

\Rightarrow 두 벡터는 관계성이 없다!

Word Vectors (word embeddings)

- Distributional Semantics: 단어의 뜻은 주변에 자주 나타나는 단어들로 표현
 - 영국의 언어학자 J.R. Firth가 제시한 개념
 - “You should know a word by the company it keeps” (J.R. Firth 1957:11)
 - 단어의 맥락은 주변에 나타나는 다른 단어들에 의해 정해진다는 뜻



J.R. Firth

출처 :

<http://linguisticstheoryii.blogspot.com/2011/09/applied-linguistics-and-linguistics.html>

...government debt problems turning into **banking** crises as happened in 2009...
...saying that Europe needs unified **banking** regulation to replace the hodgepodge...
...India has just given its **banking** system a shot in the arm...

These **context words** will represent **banking**

출처 : CS224n (2021) Lecture 1

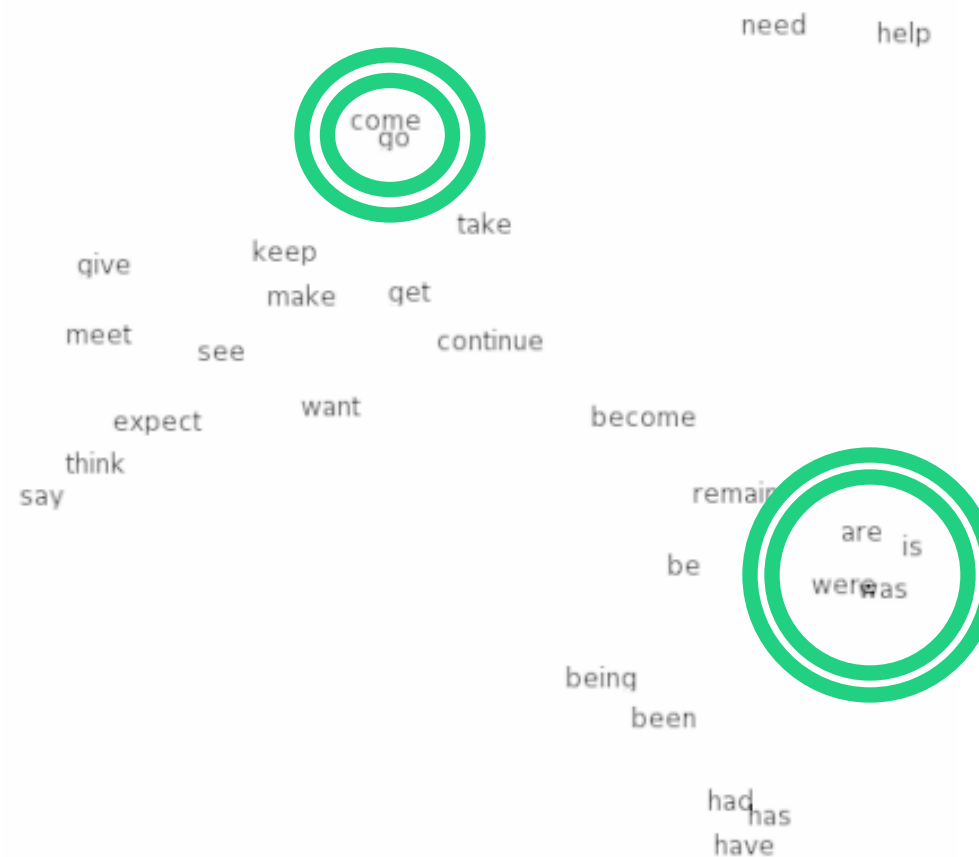
Word Vectors (word embeddings)

- Distributional Semantics: 단어의 뜻은 주변에 자주 나타나는 단어들로 표현

expect =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

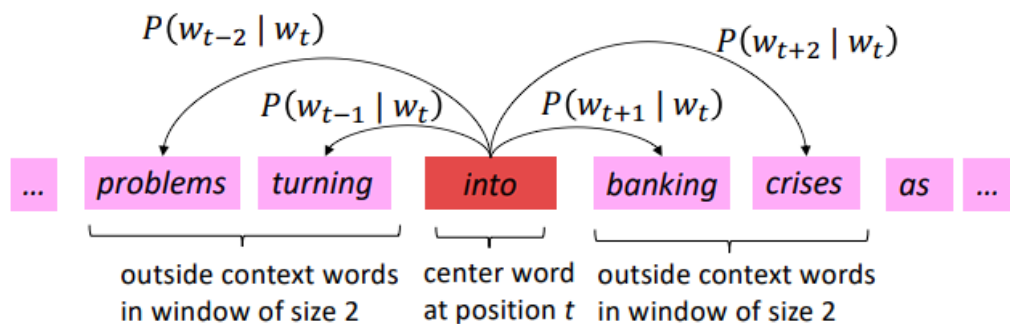
출처 : CS224n (2021) Lecture 1



출처 : CS224n (2021) Lecture 1

How to train word vectors?

- Word2Vec



출처 : CS224n (2021) Lecture 1

Problem

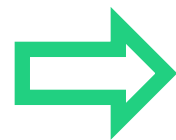
1. 큰 의미가 없는 단어
 - I like deep learning.
 - I like NLP.
2. Window 밖에 있는 중요한 단어
 - I went to the bank money ...

- 중심 단어를 기반으로 주변 단어를 예측하기
- 주변 단어를 기반으로 중심 단어를 예측하기

How to train word vectors?

- GloVe

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0



Problem

1. 행렬의 크기

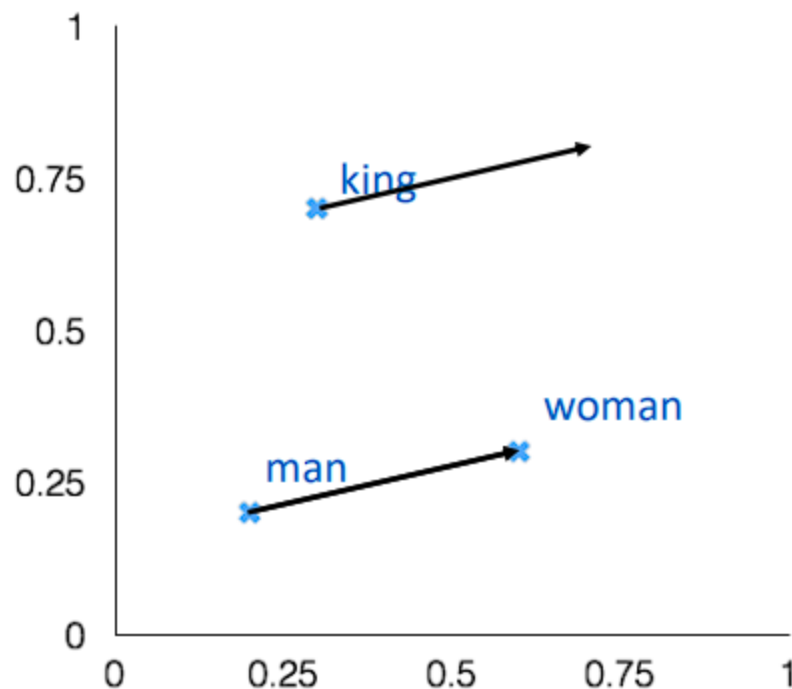
- 문장 속에 등장하는 단어의 수가 많으면?

출처 : CS224n (2021) Lecture 2

- 전체적인 통계 정보를 반영
- 두 단어가 동시에 등장할 확률을 계산

Word Vectors

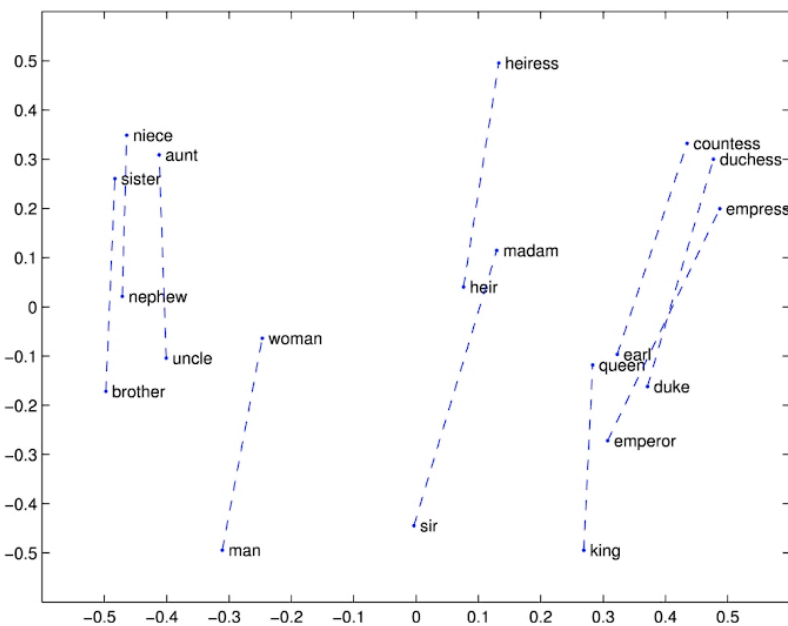
- Word Vector Analogies – 단어들 간의 관계를 잘 유추
 - Man : Woman = King : ??? ('King' - 'Man' + 'Woman' = ???)



출처 : CS224n (2021) Lecture 2

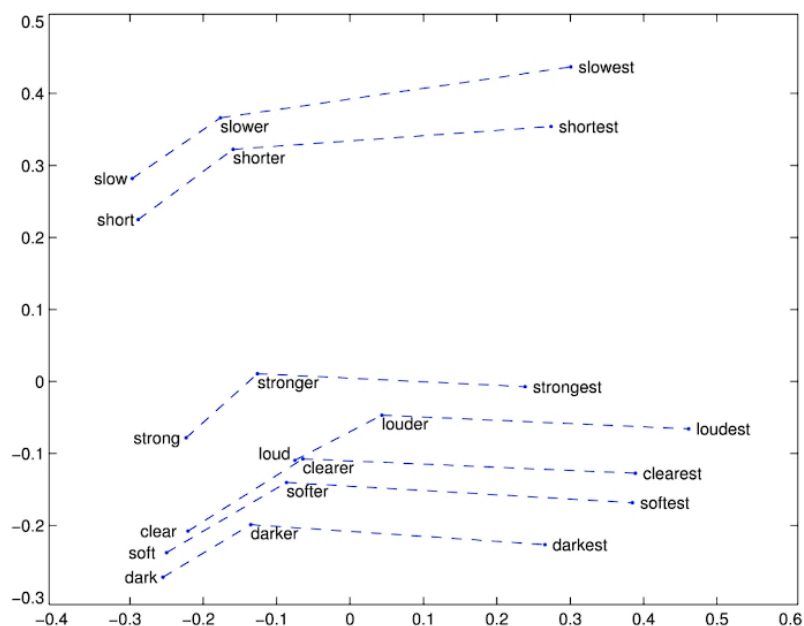
Word Vectors

Man - Woman



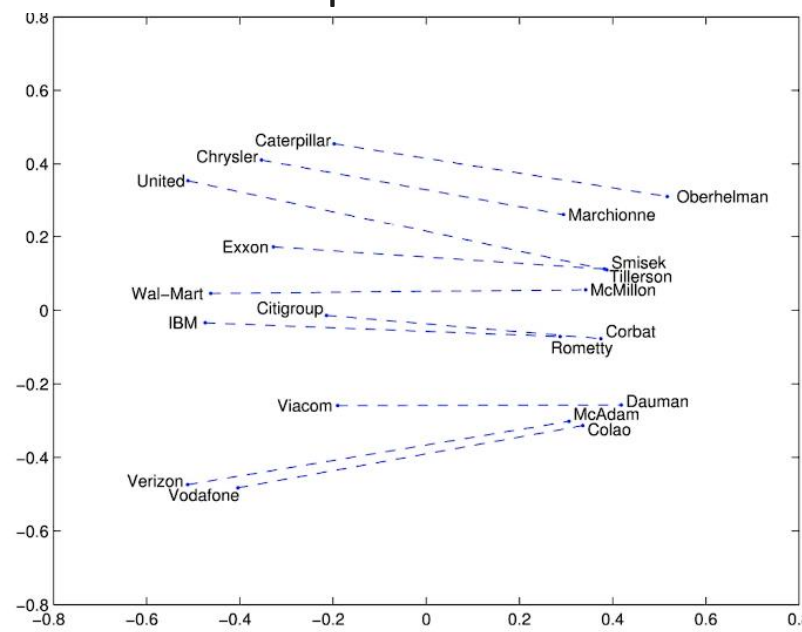
출처 : CS224n (2021) Lecture 2

Company - CEO



출처 : CS224n (2021) Lecture 2

Comparatives & Superlatives



출처 : CS224n (2021) Lecture 2

Language Modeling

What makes NLP different from CV?

- 우리가 다루고 싶은 것은 단어가 아니라 문장, 글, ...



출처 : <http://inmyownterms.com/finding-the-right-context-for-a-term/>

What makes NLP different from CV?

- 빈 칸에 들어갈 말은?
 - 오늘 _____
 - 오늘 수업이 _____
 - 오늘 수업이 1교시인데 _____
 - 오늘 수업이 1교시인데 늦잠을 _____
 - 오늘 수업이 1교시인데 늦잠을 자서 _____
 - 오늘 수업이 1교시인데 늦잠을 자서 수업에 _____
- 문장을 읽을 때는 **앞 내용이 중요하다!**

Language Modeling

- Language Modeling: 다음에 올 단어를 예측하는 것

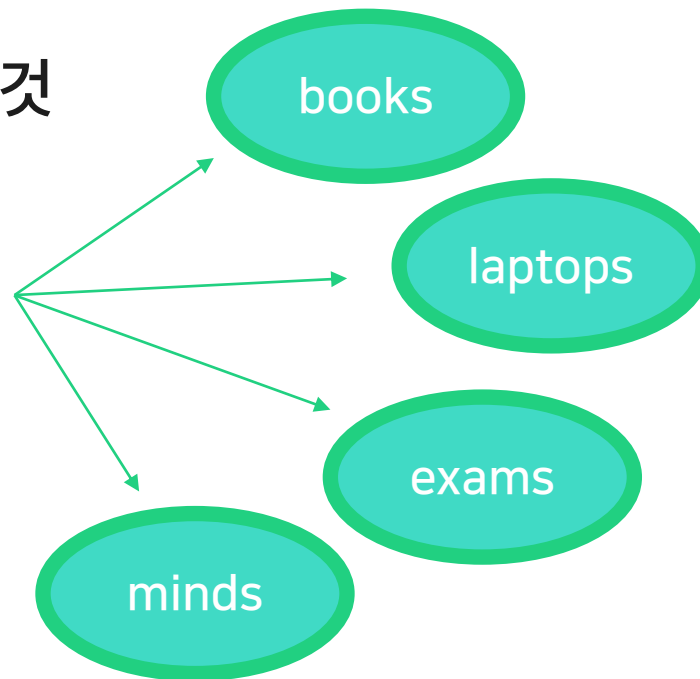
The students opened their _____

$x^{(1)}$

$x^{(2)}$

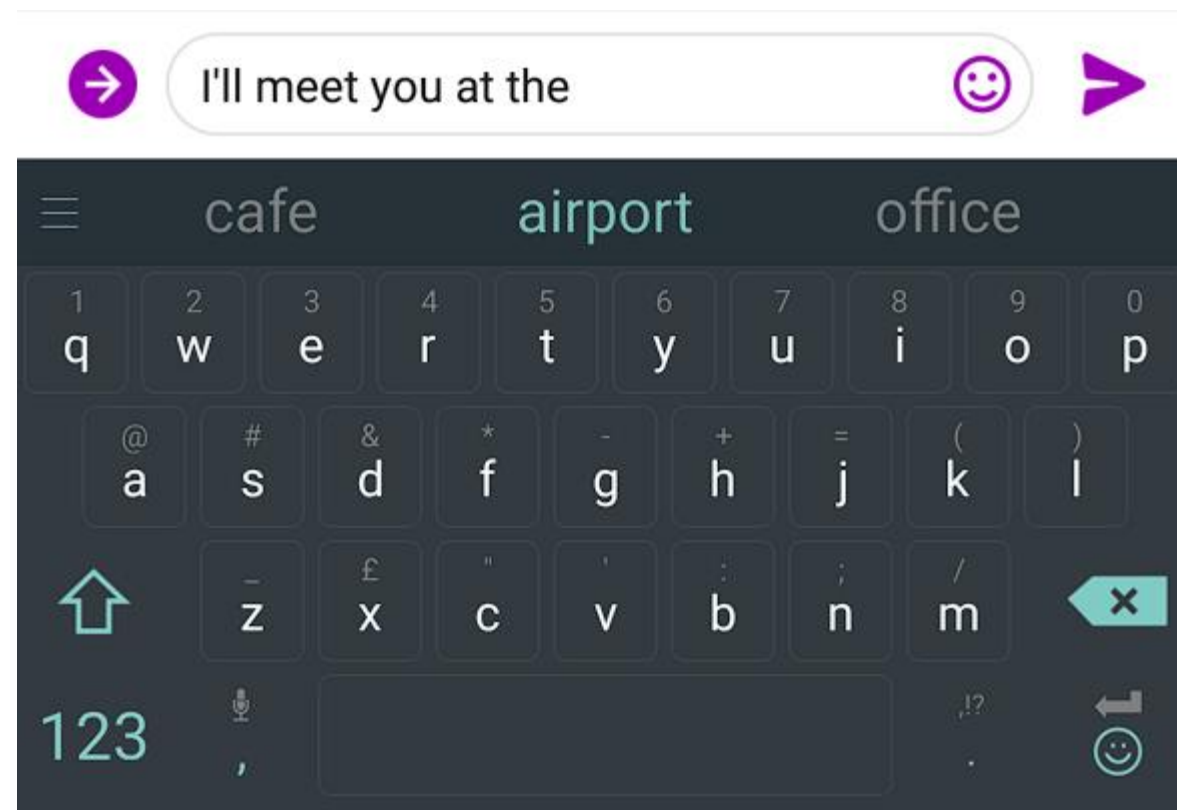
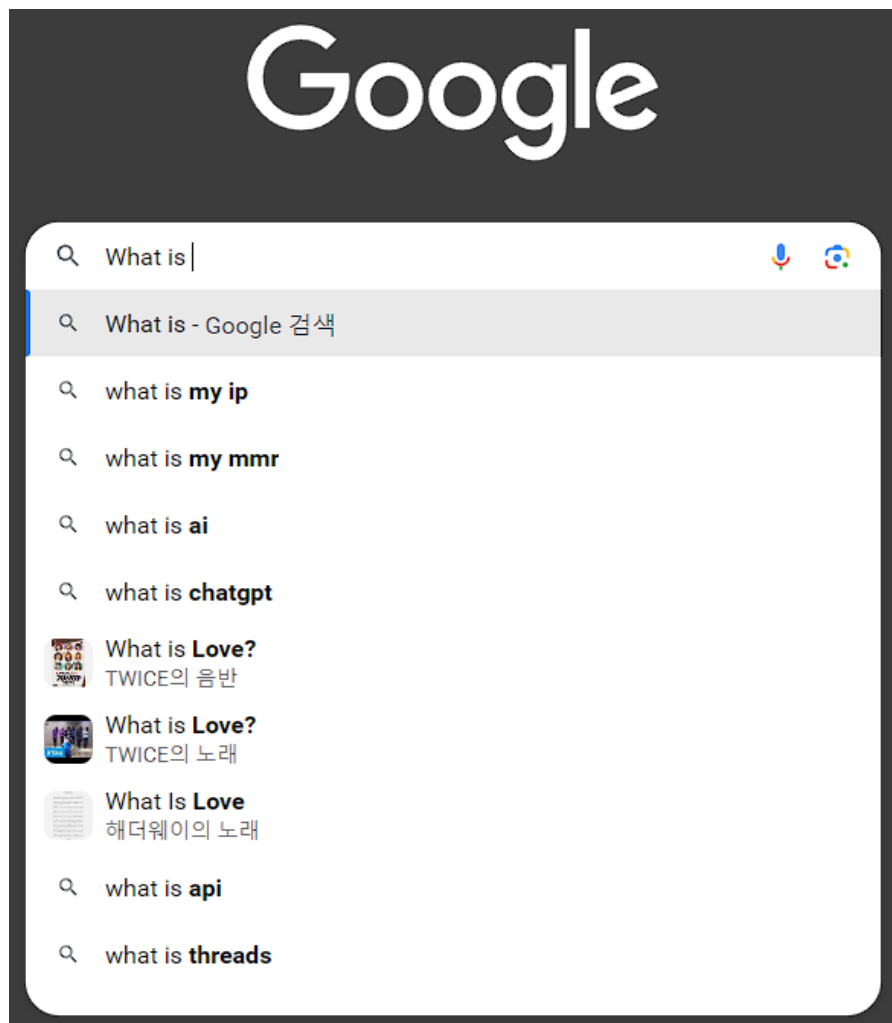
$x^{(3)}$

$x^{(4)}$



- $P(x^{(t+1)} | x^{(t)}, \dots, x^{(1)})$
 - $x^{(1)}, x^{(2)}, \dots, x^{(t)}$ 라는 t 개의 단어로부터 그 다음 단어로 올 확률이 가장 높은 단어인 $x^{(t+1)}$ 를 예측

Language Modeling



출처 : CS224n (2021) Lecture 5

n-gram Language Models

- How to learn a Language Model? n-gram Language Model을 학습하자
- Ex) the students opened their _____
- n-gram: n개의 연속되는 단어들
 - Unigrams: “the”, “students”, “opened”, “their”
 - Bigrams: “the students”, “students opened”, “opened their”
 - Trigrams: “the students opened”, “students opened their”
 - 4-grams: “the students opened their”

n-gram Language Models

- $x^{(t+1)}$ 번째 단어는 앞에 $n-1$ 개의 단어에 의해 결정

- Assumption:

- $P(x^{(t+1)} | x^{(t)}, \dots, x^{(1)}) = P(x^{(t+1)} | x^{(t)}, \dots, x^{(t-n+2)})$

$$= \frac{P(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)})}{P(x^{(t)}, \dots, x^{(t-n+2)})} \approx \frac{\text{count}(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)})}{\text{count}(x^{(t)}, \dots, x^{(t-n+2)})}$$

- Example: 4-gram Language Model

- ~~As the proctor started the clock, the~~ students opened their _____

- $P(w | \text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})}$

n-gram Language Models

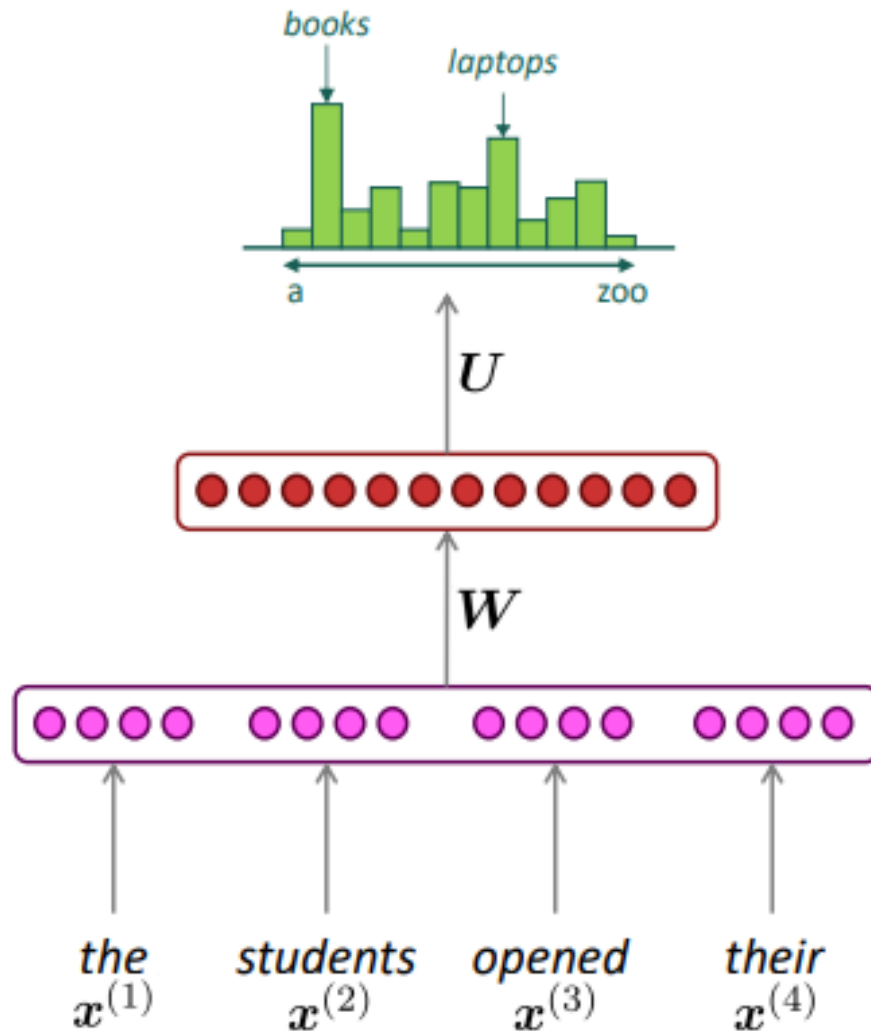
- Example: 4-gram Language Model

- $P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$
 - “Students opened their”: 1000개
 - “Students opened their **books**”: 400개 $\rightarrow P(books|students\ opened\ their) = 0.4$
 - “Students opened their **exams**”: 100개 $\rightarrow P(exams|students\ opened\ their) = 0.1$

- Problem

- Sparsity Problem
- Storage Problem

A fixed-window neural Language Model



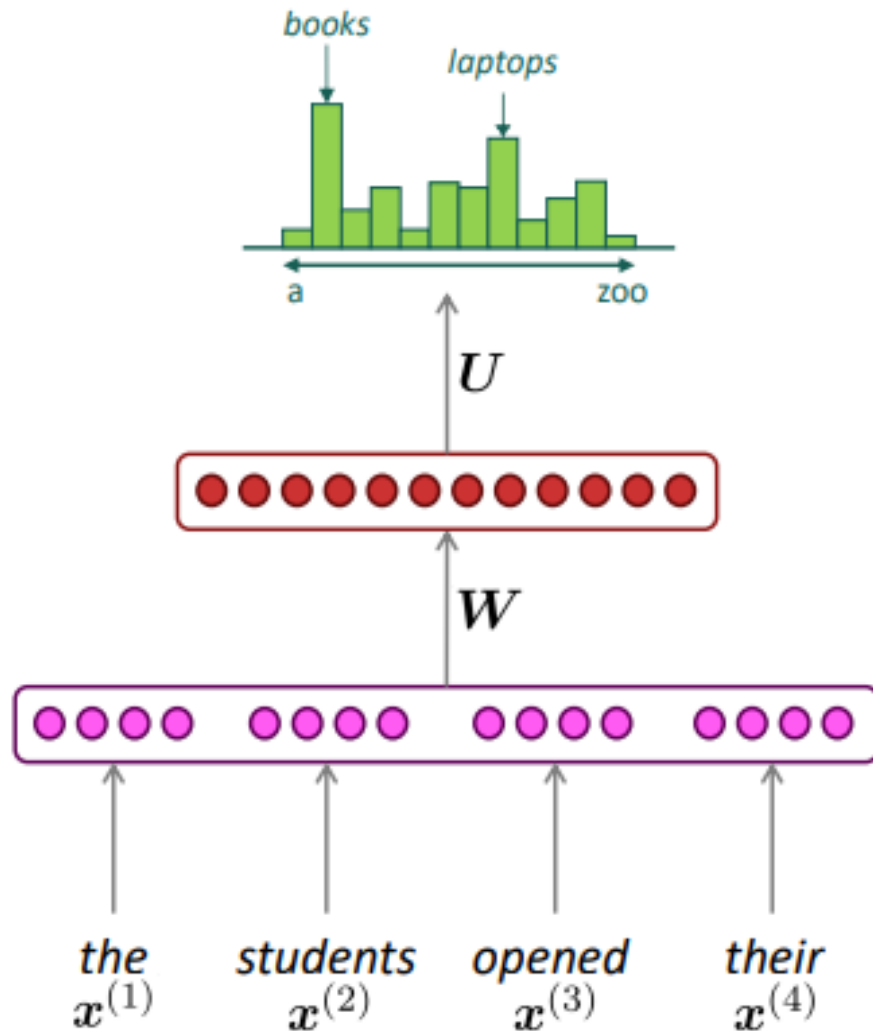
Output distribution
 $\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$

Hidden layer
 $h = f(We + b_1)$

Concatenated word embeddings
 $e = [e^{(1)}; e^{(2)}; e^{(3)}; e^{(4)}]$

words / one-hot vectors
 $x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$

A fixed-window neural Language Model



Improvements

- Sparsity / Storage Problem을 해결

Problems

- Fixed window의 크기가 작다
- Window의 크기를 늘리면 W 의 크기가 커진다
- Input 단어들을 처리하는데 대칭성이 없다
- 고정된 길이의 input만 처리할 수 있다

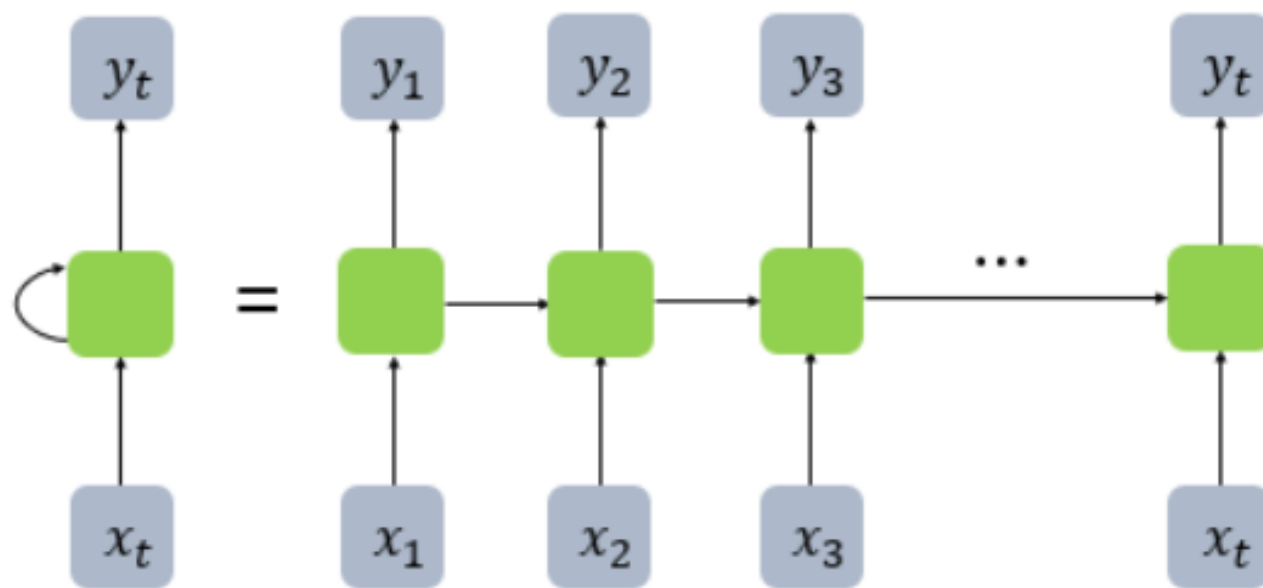
하지만 우리는 input의 길이에 상관없이 작동하는 neural architecture가 필요하다

RNNs

Recurrent Neural Networks (RNN)

- Why Recurrent?

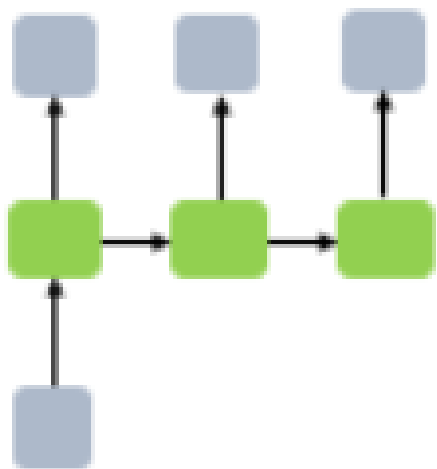
- 기존에는 hidden layer를 통과한 hidden state는 오직 출력층(Output layer)로만 향한다.
- RNN에서는 hidden state를 출력층으로 보내면서 다시 다음 계산의 입력으로 보낸다.
- 해당 역할을 하는 노드를 셀(cell)이라고 부르는데 이전의 값을 기억하는 메모리의 역할을 수행하므로 메모리 셀 또는 RNN 셀이라고 표현한다.



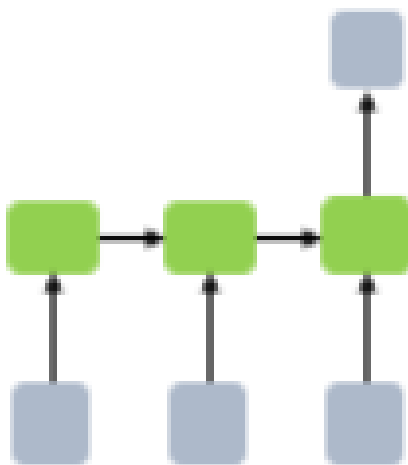
출처 : <https://wikidocs.net/22886>

Recurrent Neural Networks (RNN)

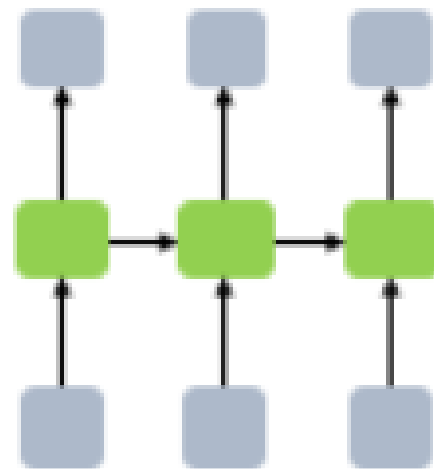
- RNN의 다양한 형태
 - Input과 Output의 길이를 **유동적**으로 설계할 수 있다.



일 대 다(one-to-many)



다 대 일(many-to-one)

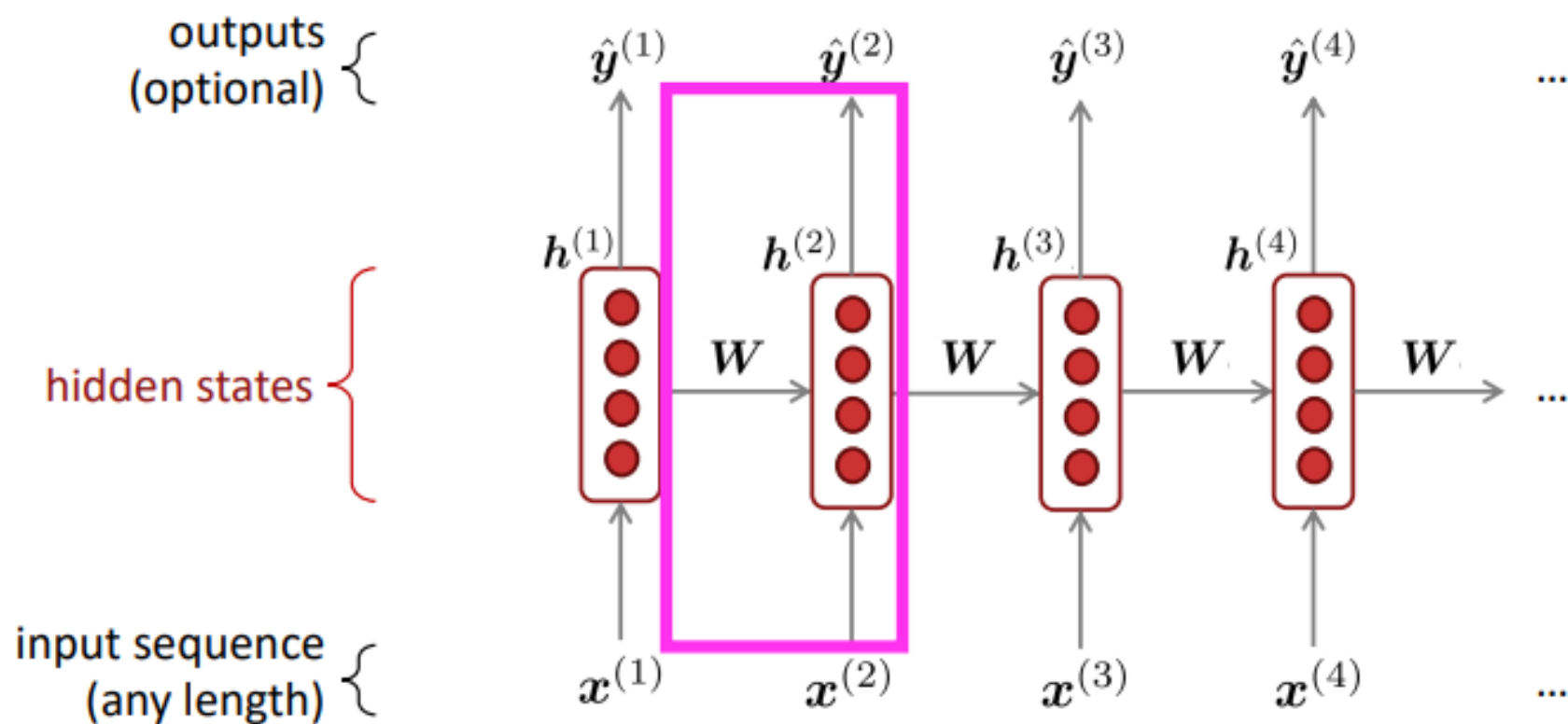


다 대 다(many-to-many)

출처 : <https://wikidocs.net/22886>

Recurrent Neural Networks (RNN)

- Input을 처리할 때, **같은 weights W** 를 반복적으로 적용하자



출처 : CS224n (2021) Lecture 5

Recurrent Neural Networks (RNN)

Output distribution

$$\hat{y}^{(t)} = \text{softmax}(Uh^{(t)} + b_2) \in \mathbb{R}^{|V|}$$

Hidden layer

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$

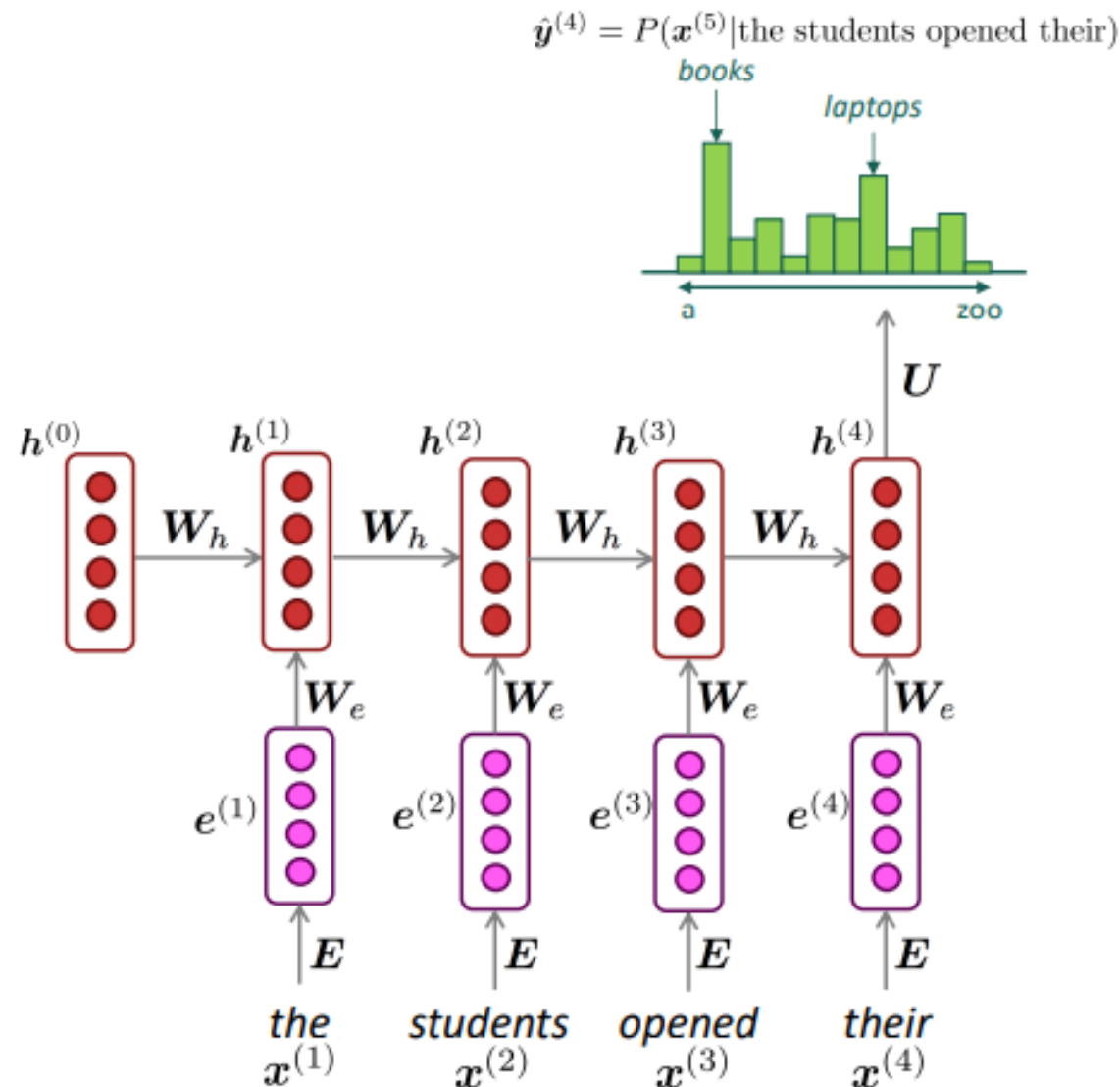
$h^{(0)}$: initial hidden state

Concatenated word embeddings

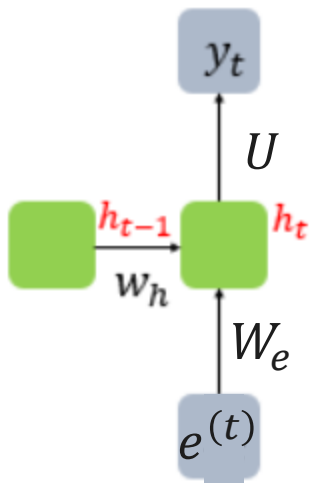
$$e^{(t)} = E x^{(t)}$$

words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$



Recurrent Neural Networks (RNN)



Hidden state:

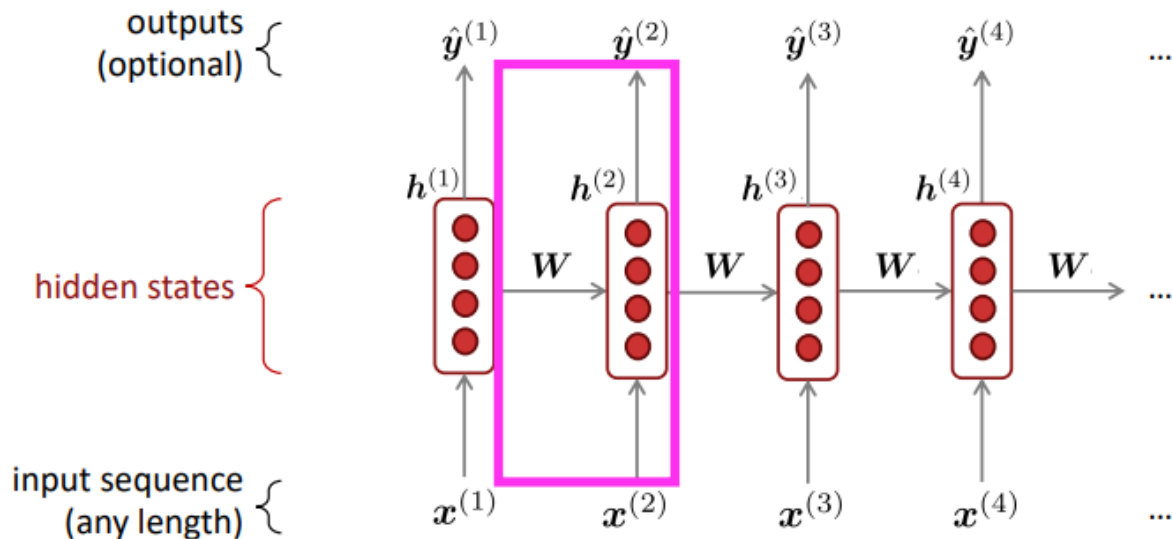
$$h_t = \sigma(W_h h^{(t)} + W_e e^{(t)} + b)$$

Output vector:

$$y_t = f(Uh_t + b)$$

$$\sigma \left(\begin{matrix} W_h \\ D_h \times D_h \end{matrix} \times \begin{matrix} h_{t-1} \\ D_h \times 1 \end{matrix} + \begin{matrix} W_x \\ D_h \times d \end{matrix} \times \begin{matrix} x_t \\ d \times 1 \end{matrix} + \begin{matrix} b \\ D_h \times 1 \end{matrix} \right) = \begin{matrix} h_t \\ D_h \times 1 \end{matrix}$$

Recurrent Neural Networks (RNN)



출처 : CS224n (2021) Lecture 5

Advantages

- Input의 길이에 상관없이 작동한다
- 이전 input에 대한 정보를 반영할 수 있다
- 모델 사이즈가 input에 영향을 받지 않는다
- 모든 input을 동일하게 처리한다

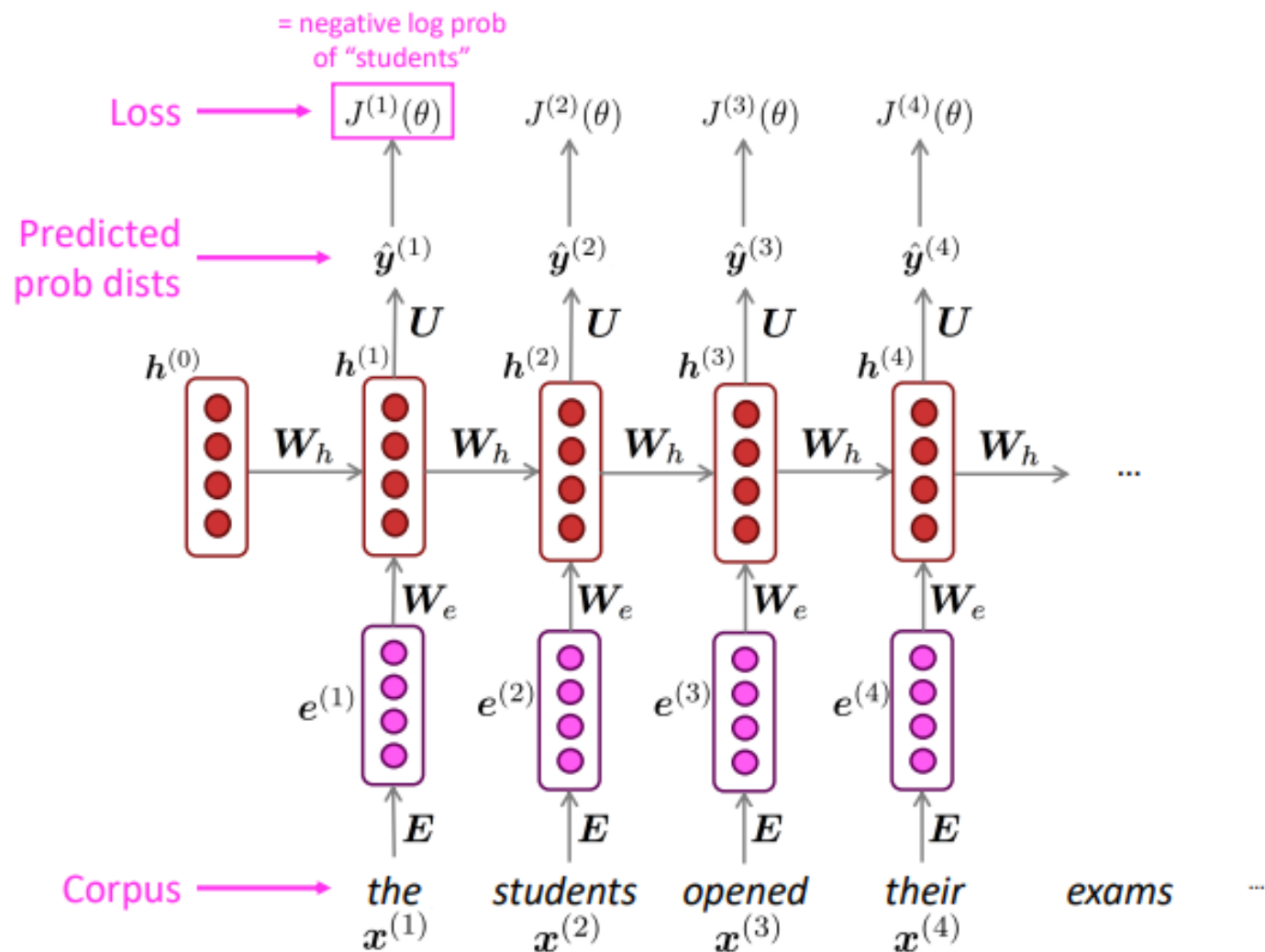
Problems

- 계산 속도가 느리다
- 실제로는 여러 step 전의 정보를 잘 반영하지는 못한다

Training an RNN Language Model

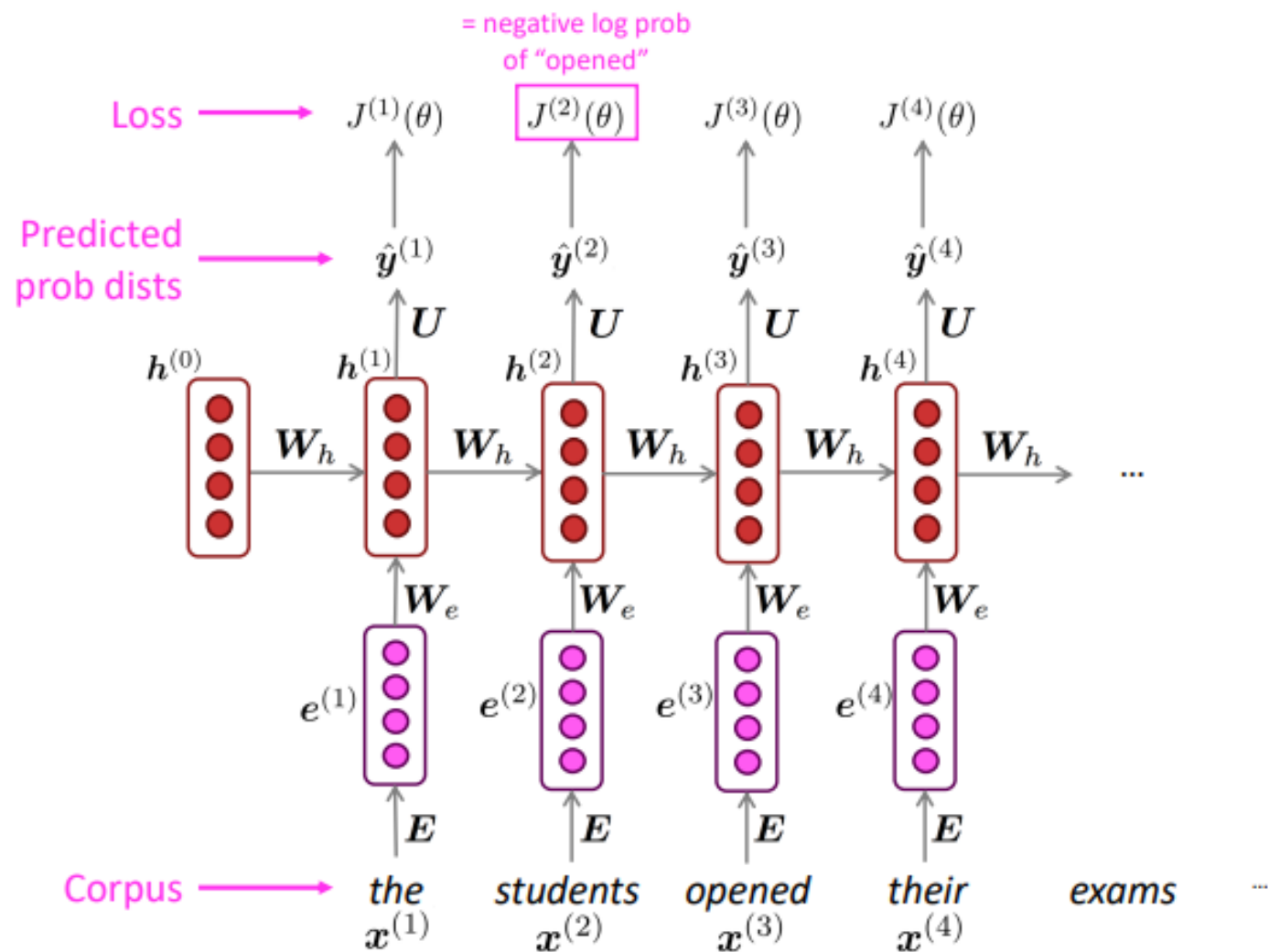
- 각 step t 마다 output distribution $\hat{y}^{(t)}$ 를 계산
 - $t-1$ 개의 단어들을 바탕으로 t 번째 단어의 확률을 예측
- 각 step t 마다 예측한 값과 실제 정답을 비교
 - 이 때, Loss function은 **Cross-Entropy Loss**를 사용
 - $J^{(t)}(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = -\sum_{w \in V} y_w^{(t)} \log \hat{y}_w^{(t)}$
- 전체 학습 데이터에 대한 Loss는 각 Loss들의 평균
 - $J(\theta) = \frac{1}{T} \sum^T J^{(t)}(\theta)$

Training an RNN Language Model



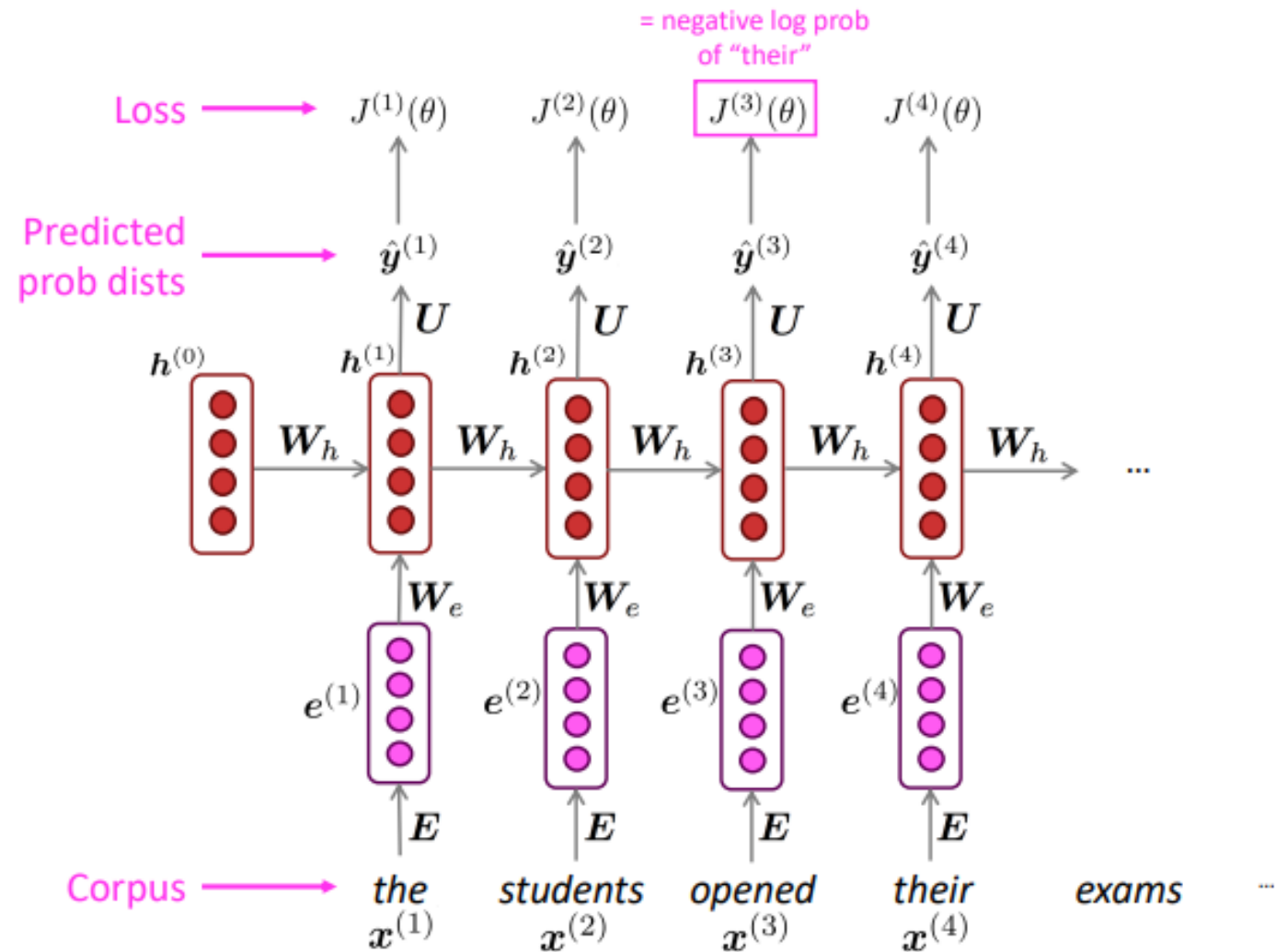
출처 : CS224n (2021) Lecture 5

Training an RNN Language Model



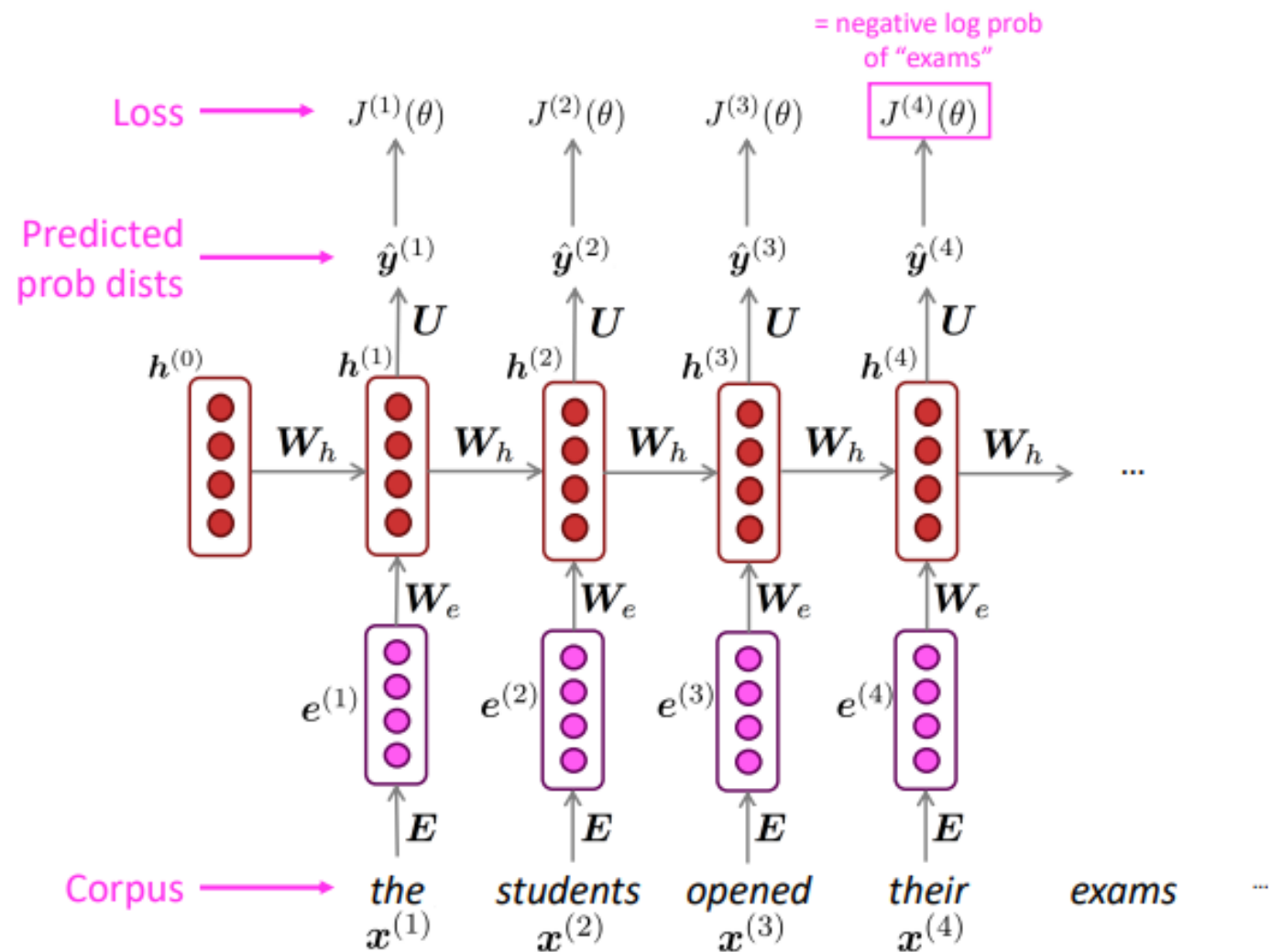
출처 : CS224n (2021) Lecture 5

Training an RNN Language Model



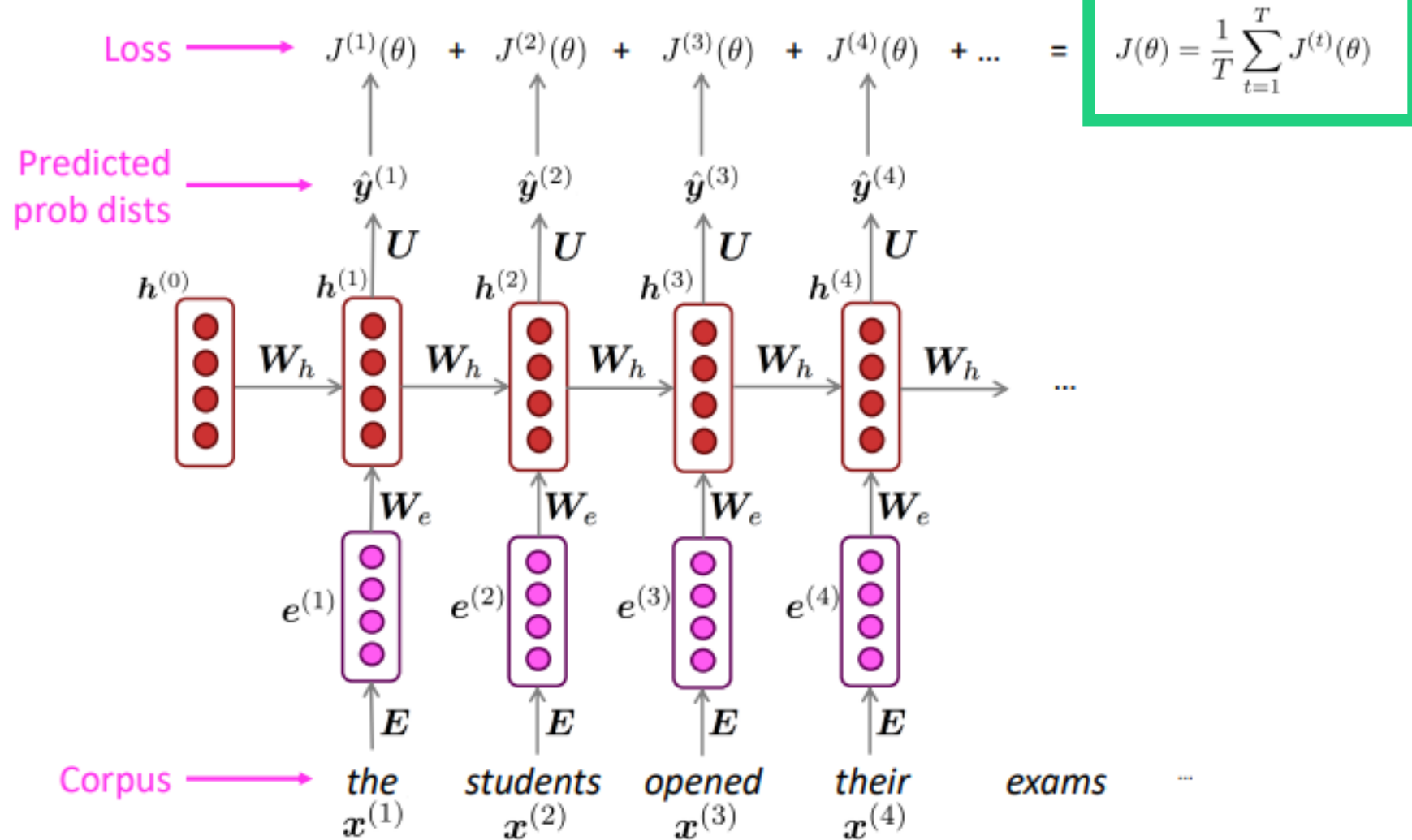
출처 : CS224n (2021) Lecture 5

Training an RNN Language Model



출처 : CS224n (2021) Lecture 5

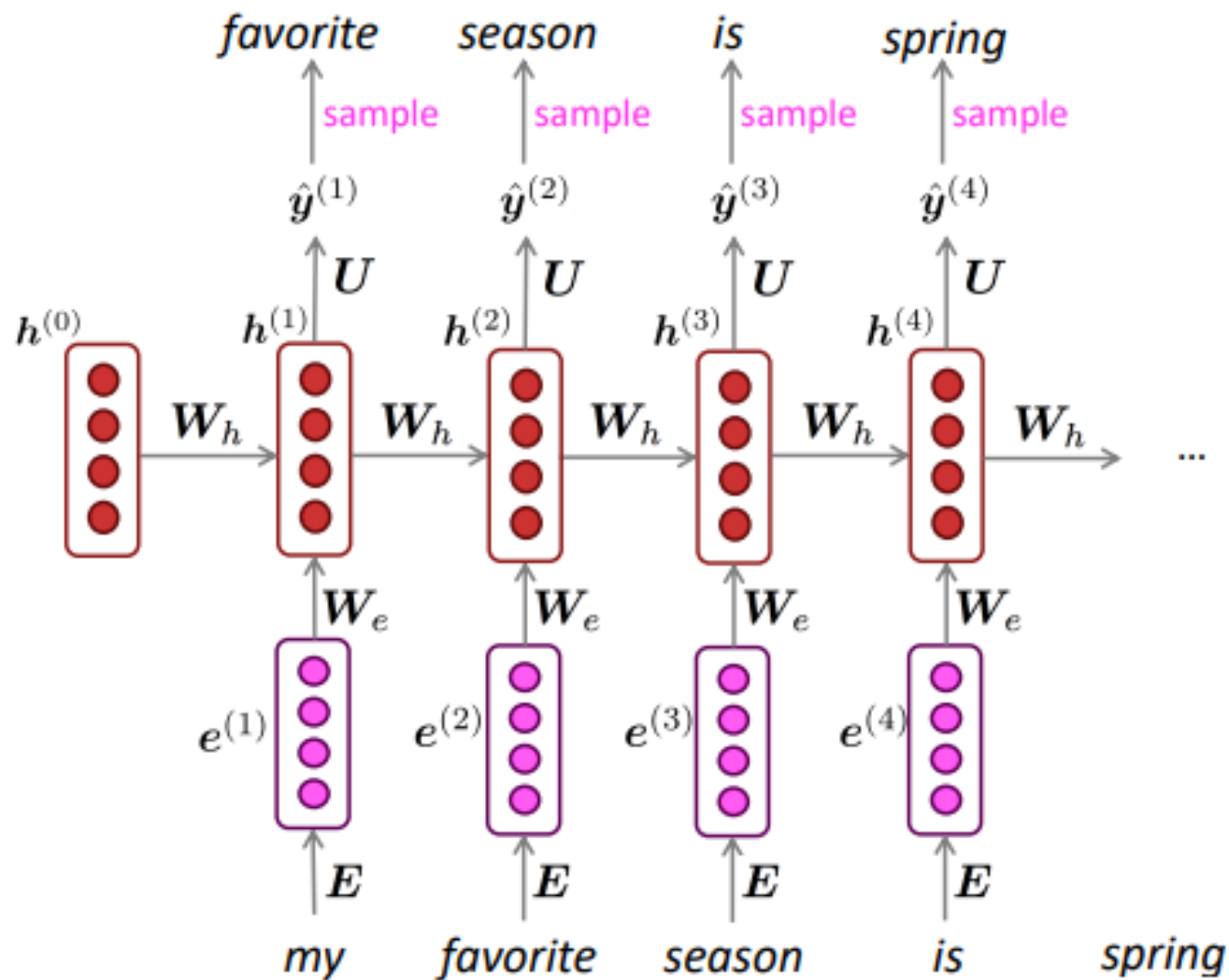
Training an RNN Language Model



Teacher Forcing

- 학습 과정에서 $t-1$ 개의 input으로 다음 단어를 예측하는데 만약 틀린다면?
 - 오늘 _____
 - 오늘 수업이 1교시인데 늦잠을 자서 수업에 _____
- 모델이 잘못된 문장을 학습하지 않도록 예측한 단어가 아닌 **실제 정답**을 다음 input으로 넣어준다. 마치 선생님(Teacher)가 정답을 알려주듯이...
 - ‘오늘’ → ‘오늘 나는’ 이라고 예측한다면?
 - 2번째 input으로 ‘나는’이 아니라 ‘수업이’를 넣어준다

Generating text

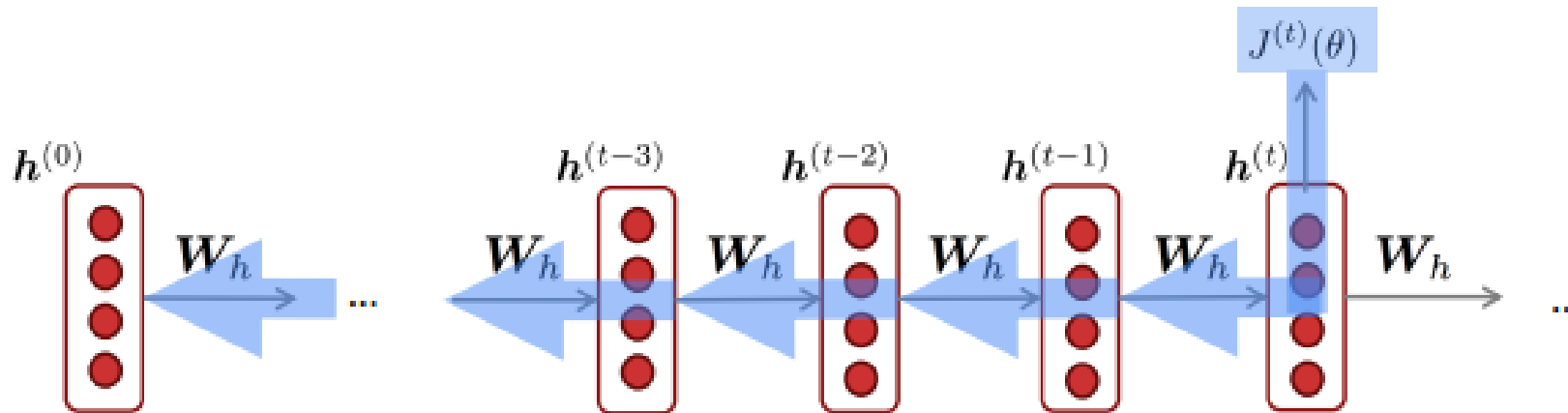


출처 : CS224n (2021) Lecture 5

Backpropagation for RNNs

- Backpropagation through time

- RNN을 학습하는 과정은 다른 딥러닝 모델과 크게 다르지 않다
- 다만, W_h 가 반복해서 곱해지는 만큼 Loss를 W_h 에 대해 미분하고 W_h 를 업데이트 하는 방식에 조금 차이가 있다.

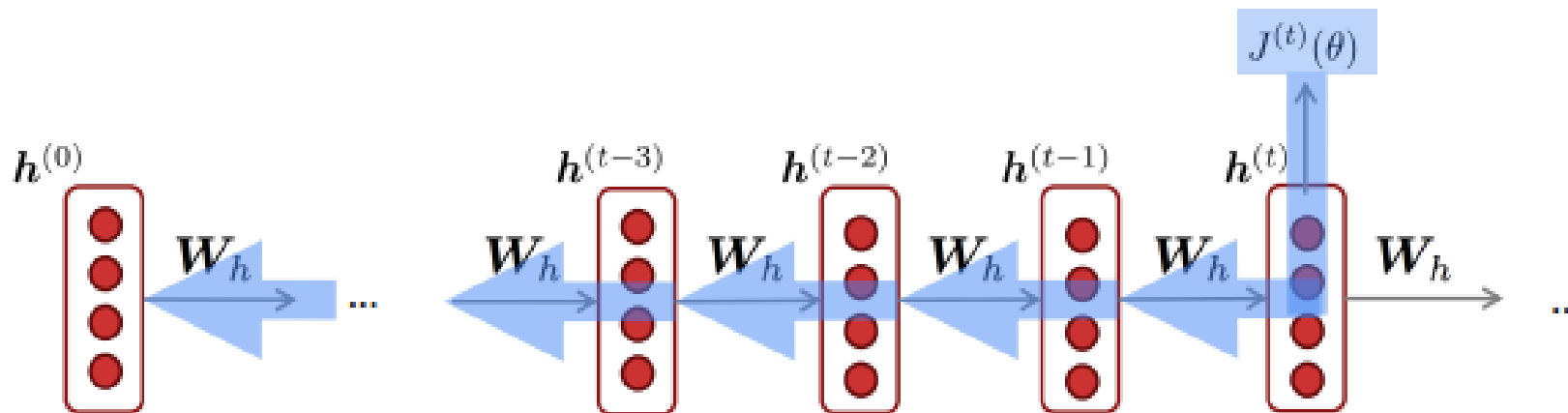


출처 : CS224n (2021) Lecture 5

Backpropagation for RNNs

- Backpropagation through time

- $\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^t \frac{\partial J^{(i)}}{\partial W_h} \Big|_{(i)}$
- W_h 를 각 step마다 바로 업데이트 하지 않고 나중에 한 번에 업데이트 한다.
- 순전파를 계산할 때 W_h 는 일종의 constant였기 때문이다.
- 값을 바로 업데이트 해주면 이후의 역전파 계산 과정에서 W_h 의 값이 달라지게 된다.



출처 : CS224n (2021) Lecture 5

Evaluating Language Models

- Perplexity

- Perplex: '당황하다', '당혹하게 하다 '
- Perplexity란 모델이 헛갈려하는 정도...?

- $$perplexity = \prod_{t=1}^T \left(\frac{1}{P_{LM}(x^{(t+1)} | x^{(t)}, \dots, x^{(1)})} \right)^{\frac{1}{T}}$$

- 모델이 계산한 확률의 역수를 전부 곱한 것
- 확률이 항상 1이라면 곱하면 1, 0에 가깝다면 무수히 커진다
- 따라서 **Perplexity는 낮을수록 좋다!**
 - 낮을수록 다음 단어를 예측하는데 덜 헛갈린다는 뜻..?

Evaluating Language Models

n-gram model →

Increasingly complex RNNs ↓

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Perplexity improves (lower is better) ↓

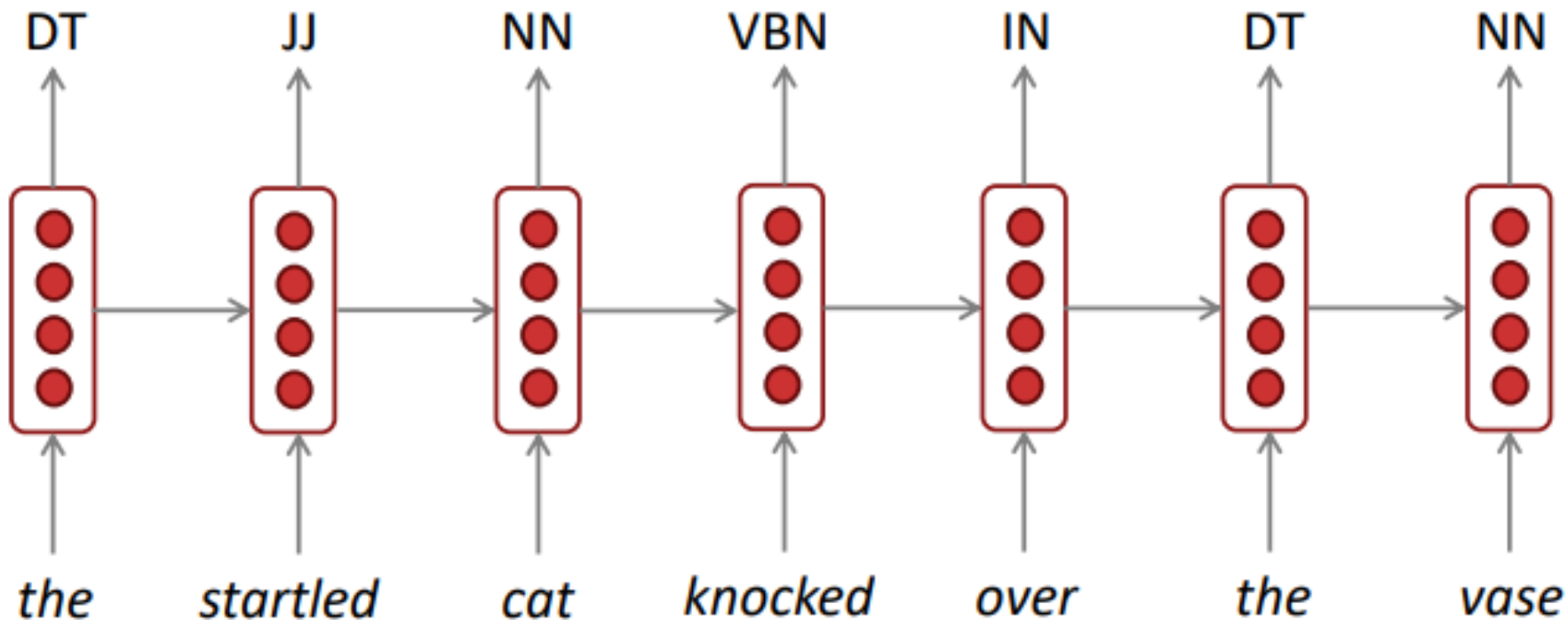
출처 : CS224n (2021) Lecture 5

Various NLP tasks using RNNs

- Language Modeling은 다음에 올 단어를 예측하는 것
- RNN \neq Language Model
- 왜냐하면 RNN을 이용하면 단순히 다음 단어를 예측하는 것보다 더 다양한 task를 처리할 수 있다.

Various NLP tasks using RNNs

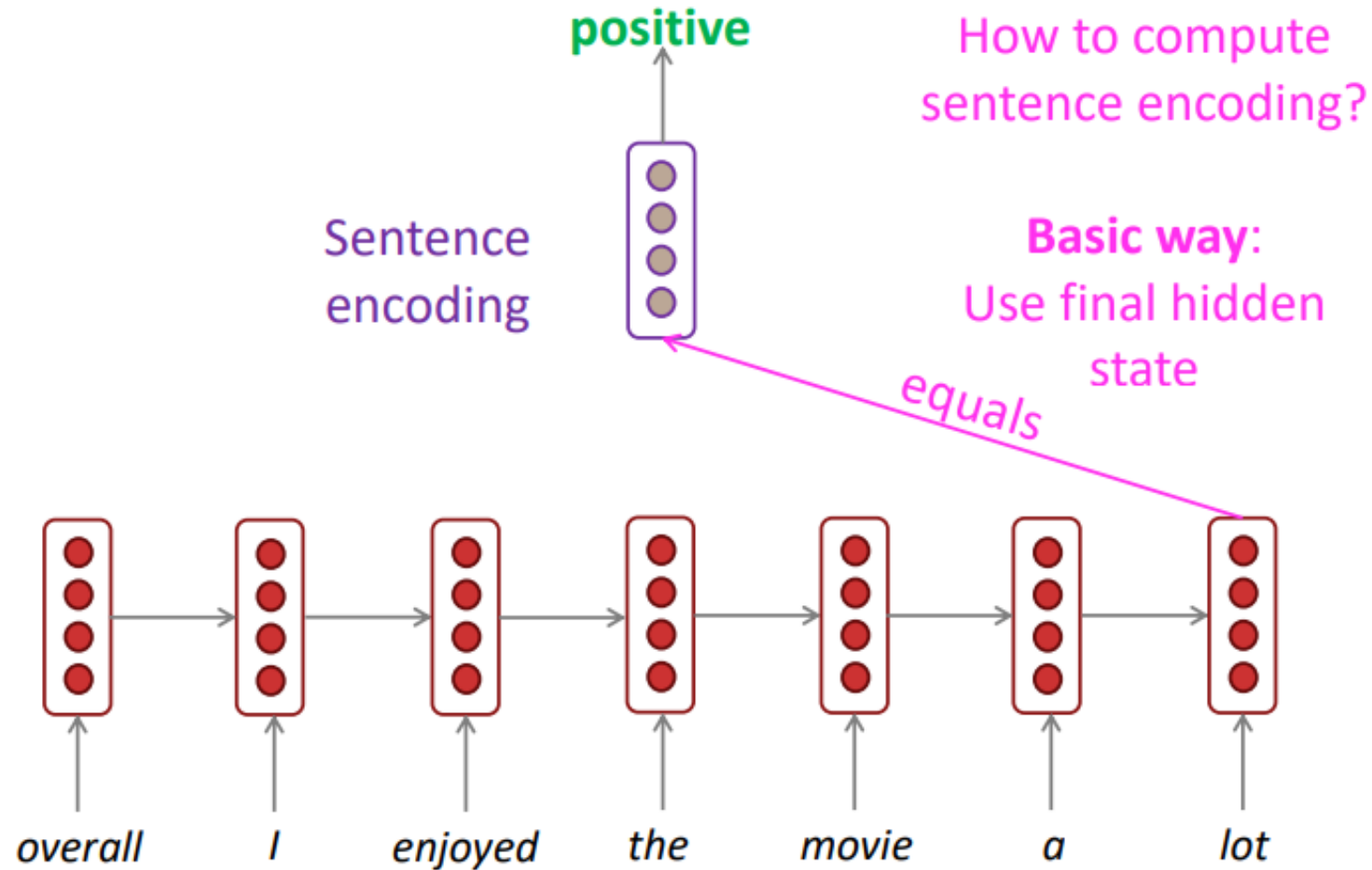
- POS tagging, NER recognition



출처 : CS224n (2021) Lecture 5

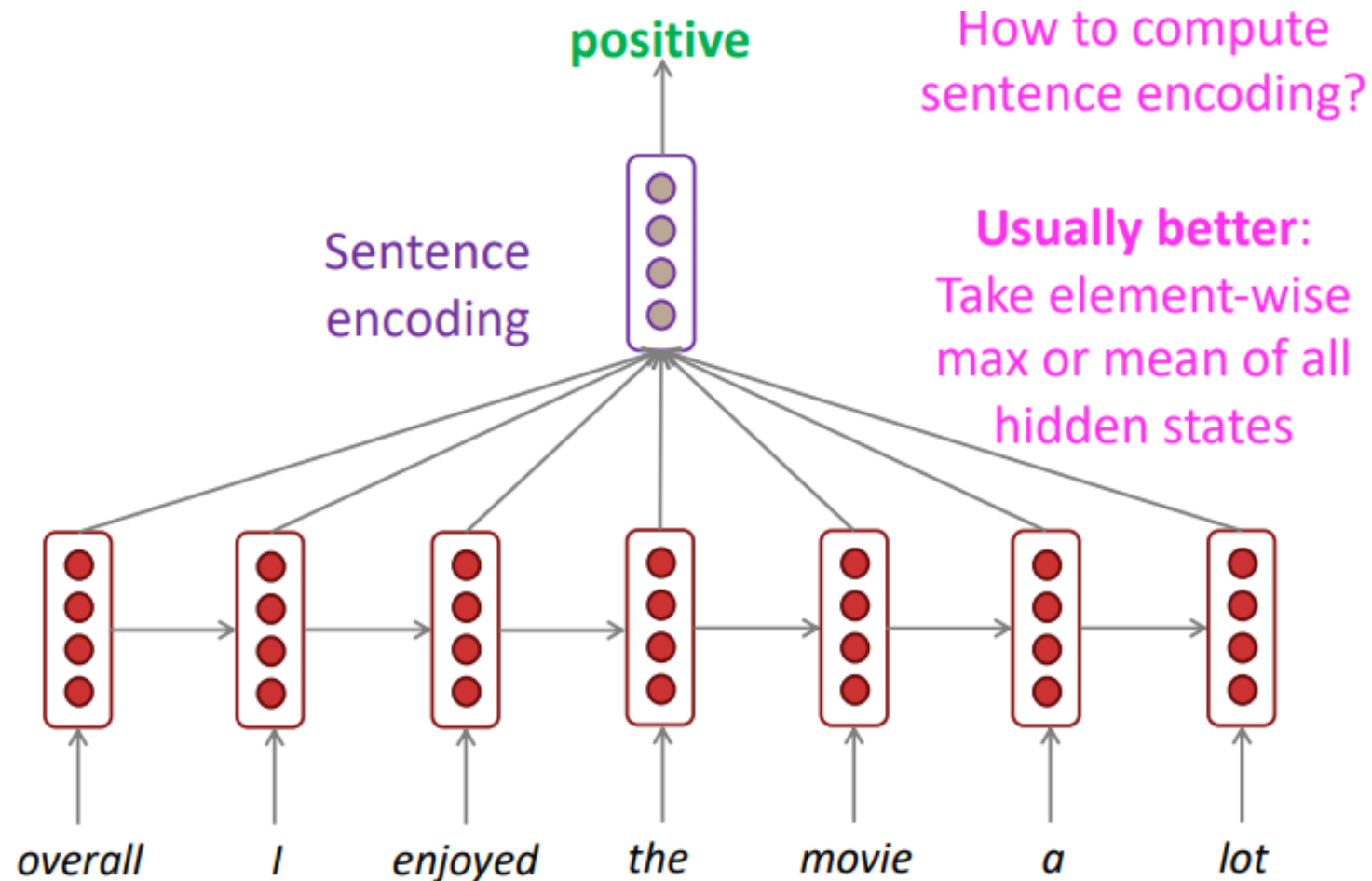
Various NLP tasks using RNNs

- Sentiment classification



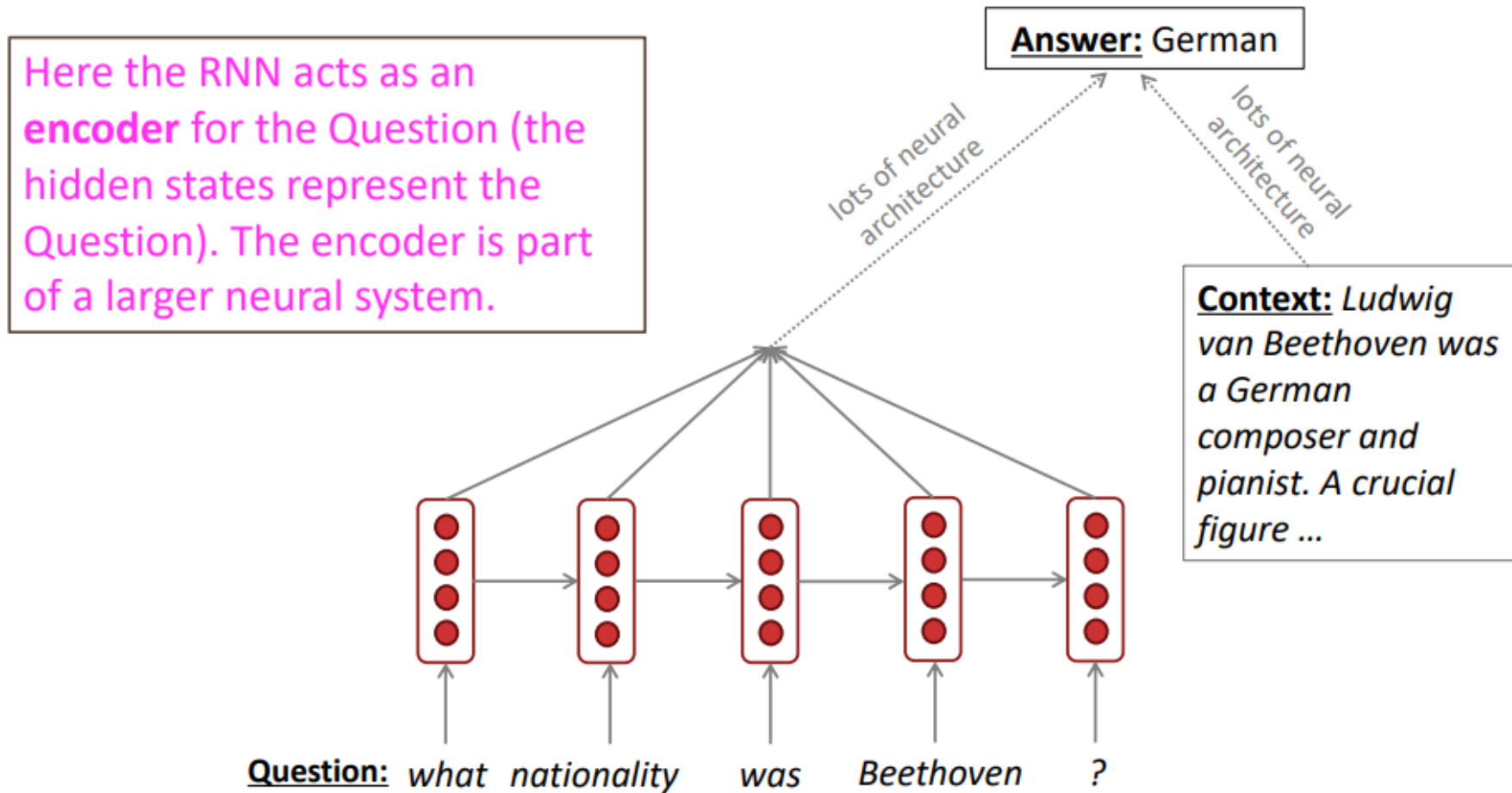
Various NLP tasks using RNNs

- Sentiment classification



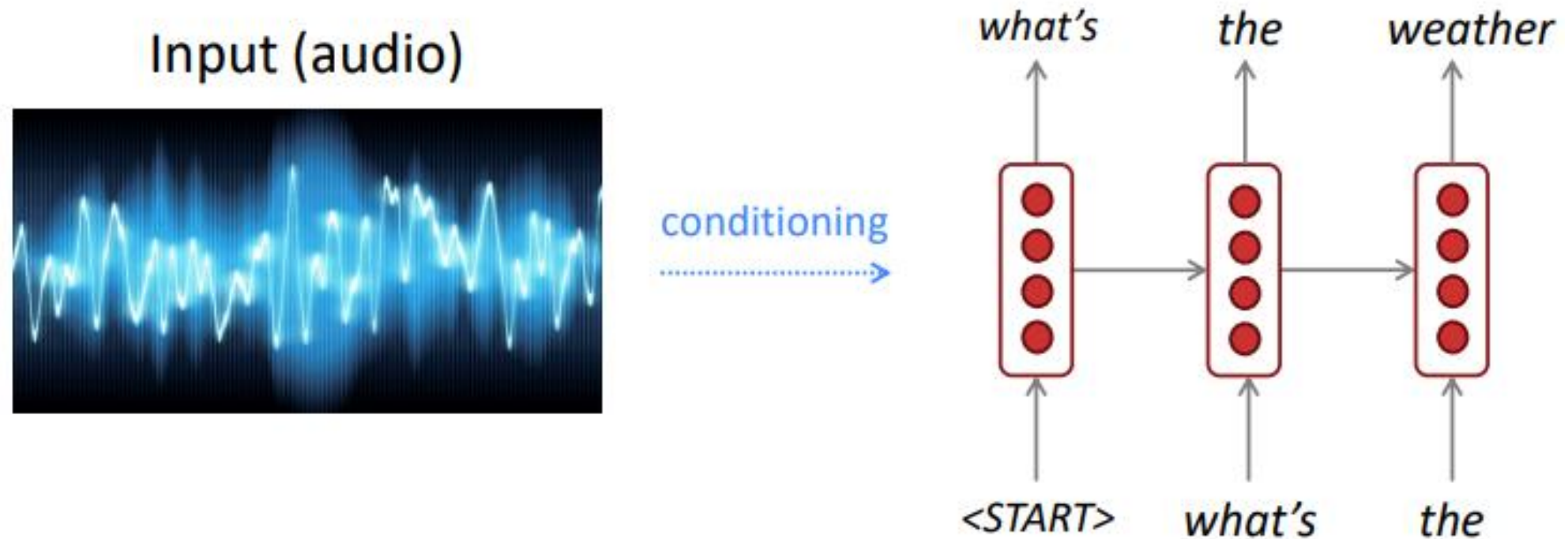
Various NLP tasks using RNNs

- Question answering, Machine translation



Various NLP tasks using RNNs

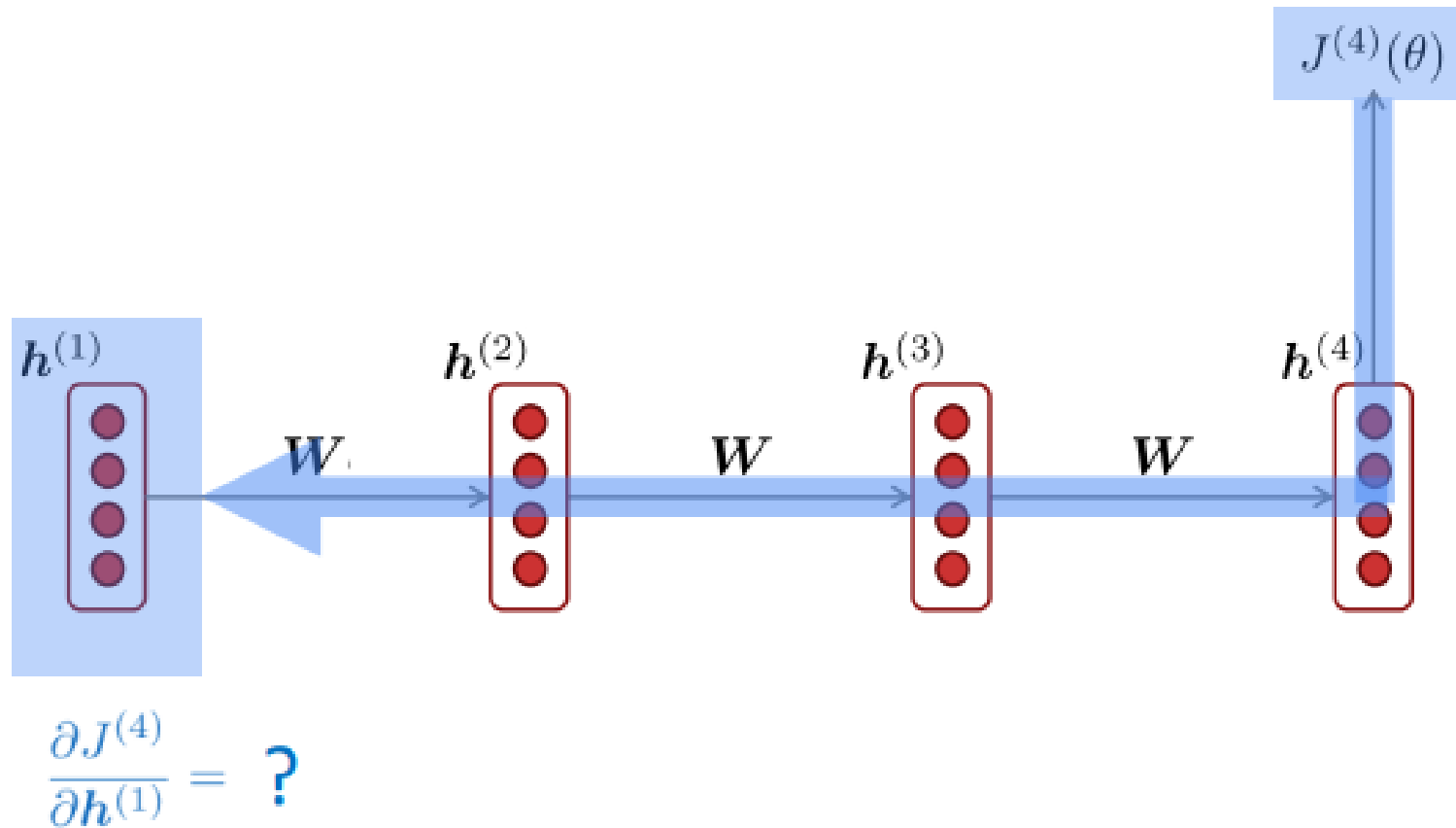
- Speech recognition



출처 : CS224n (2021) Lecture 5

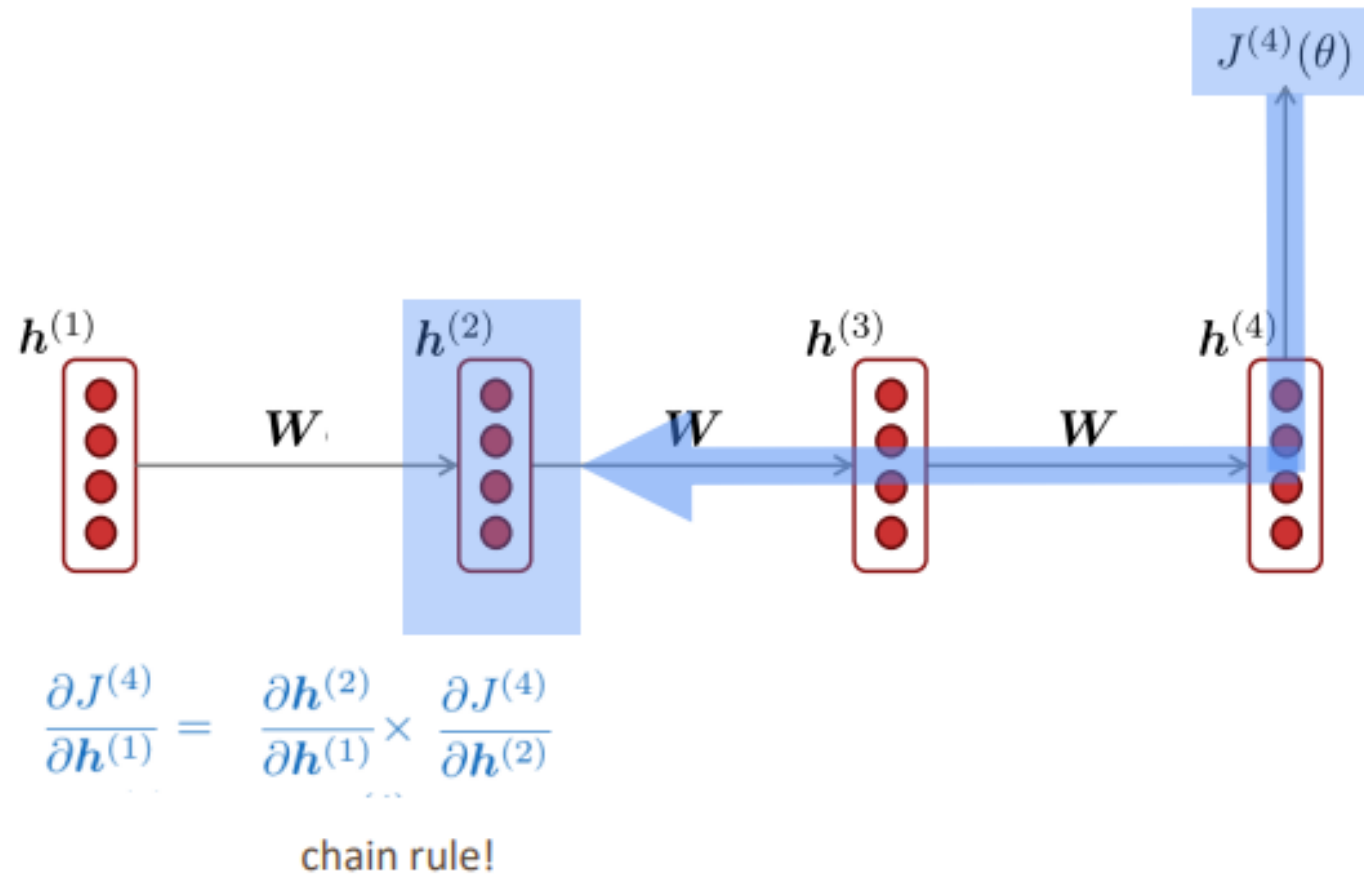
Problems of RNNs

- RNN에서 backpropagation을 계산할 때...

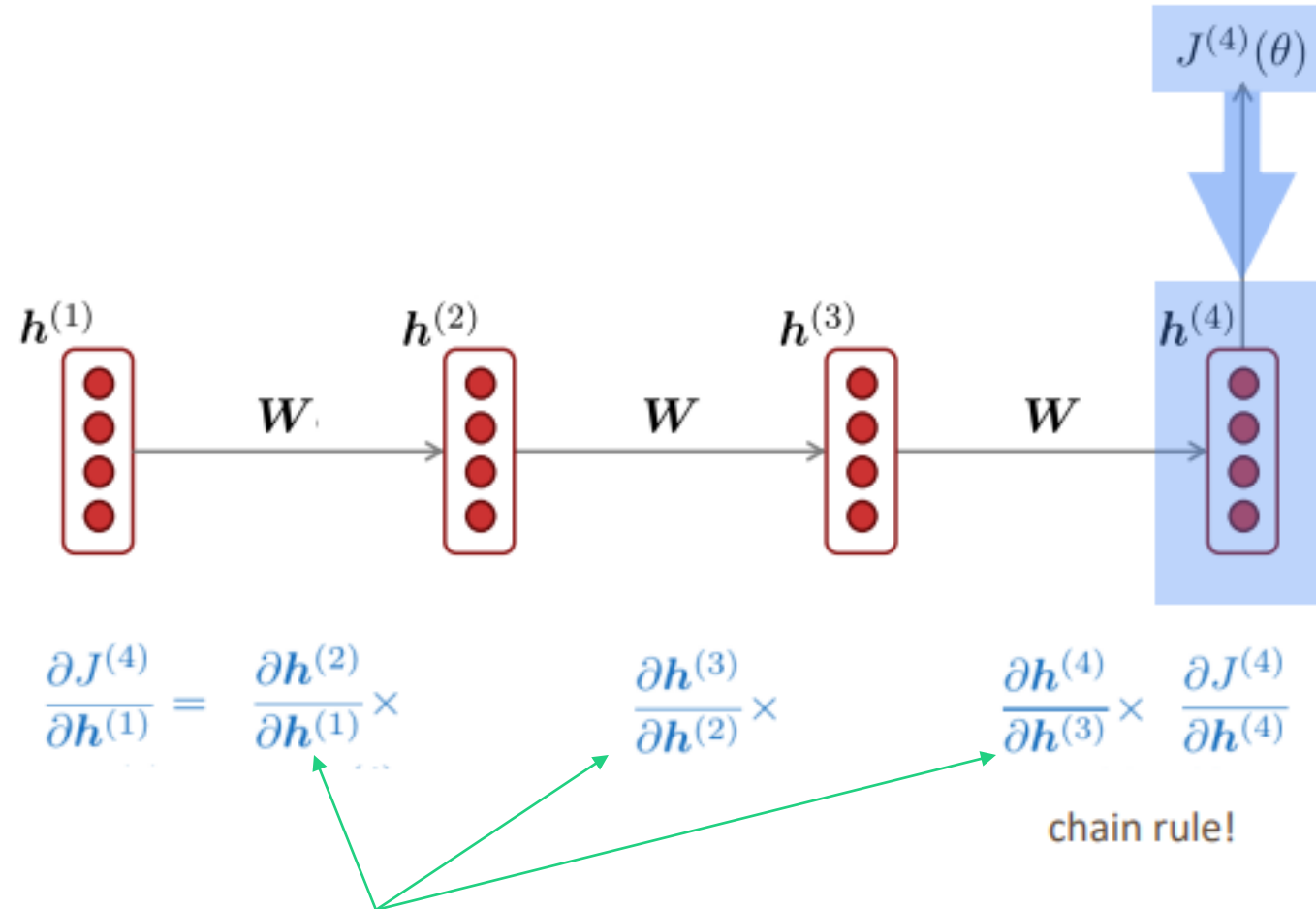


Problems of RNNs

- Chain Rule을 적용해서 구하게 된다



Problems of RNNs



이 값들에 따라 문제가 생길 수 있다

출처 : CS224n (2021) Lecture 6

Problems of RNNs ① Exploding Gradient

- 각각의 값이 너무 크면 gradient의 값이 커지게 되고 SGD update step이 커진다

$$\frac{\partial J^{(4)}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \times \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \times \frac{\partial \mathbf{h}^{(4)}}{\partial \mathbf{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \mathbf{h}^{(4)}}$$

chain rule!

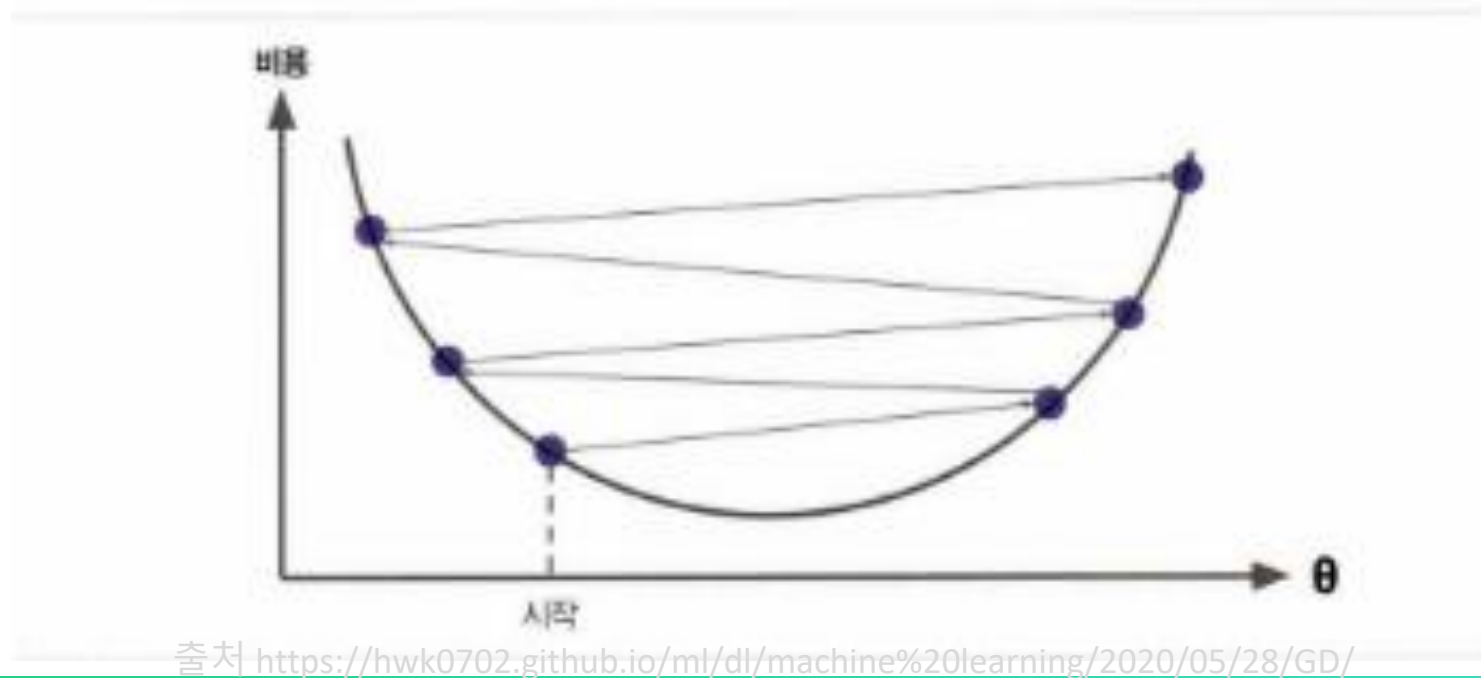
각 값이 너무 커지면 전체 gradient의 값이 커진다

Problems of RNNs ① Exploding Gradient

- SGD update step이 커지면 **bad updates**일 가능성이 높다

gradient

- $\theta^{new} = \theta^{old} - \alpha \overbrace{\nabla_{\theta} J(\theta)}$



출처 <https://hwk0702.github.io/ml/dl/machine%20learning/2020/05/28/GD/>

Problems of RNNs ① Exploding Gradient

- Gradient Exploding은 Gradient clipping으로 해결할 수 있다
- Gradient clipping: gradient의 norm이 threshold보다 크면 SGD update를 진행하기 전에 크기를 줄여준다.
- Update의 **방향은 유지**하되 그 크기를 줄인다.

Algorithm 1 Pseudo-code for norm clipping

```
 $\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$   
if  $\|\hat{\mathbf{g}}\| \geq threshold$  then  
     $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$   
end if
```

출처 : CS224n (2021) Lecture 6

Problems of RNNs ② Vanishing Gradient

- 각각의 값이 너무 작으면 gradient의 값이 0에 가까워지고 SGD update step가 일어나지 않는다.

$$\frac{\partial J^{(4)}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \times \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \times \frac{\partial \mathbf{h}^{(4)}}{\partial \mathbf{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \mathbf{h}^{(4)}}$$

chain rule!

각 값이 너무 작으면 전체 gradient의 값은 0에 가까워진다

Problems of RNNs ② Vanishing Gradient

- 각각의 값이 너무 작으면 gradient의 값이 0에 가까워지고 SGD update step가 일어나지 않는다.

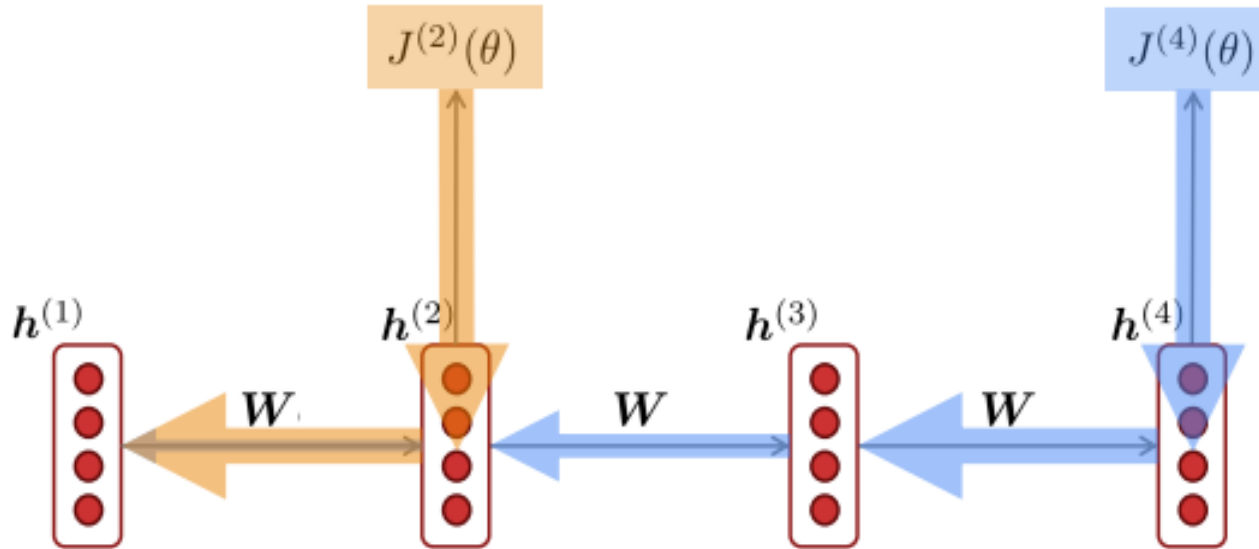
$$\frac{\partial J^{(4)}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \times \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \times \frac{\partial \mathbf{h}^{(4)}}{\partial \mathbf{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \mathbf{h}^{(4)}}$$

chain rule!

각 값이 너무 작으면 전체 gradient의 값은 0에 가까워진다

Problems of RNNs ② Vanishing Gradient

- Vanishing gradient가 왜 문제가 될까?
 - 모델의 weights를 업데이트 할 때 거리가 멀수록 영향이 적어진다.
 - 즉, 업데이트 과정에서 가까이 있는 정보만 반영된다.
 - 학습 과정에서 멀리 떨어진 단어들 사이의 dependency를 학습하지 못하게 되는 문제점이 있다.



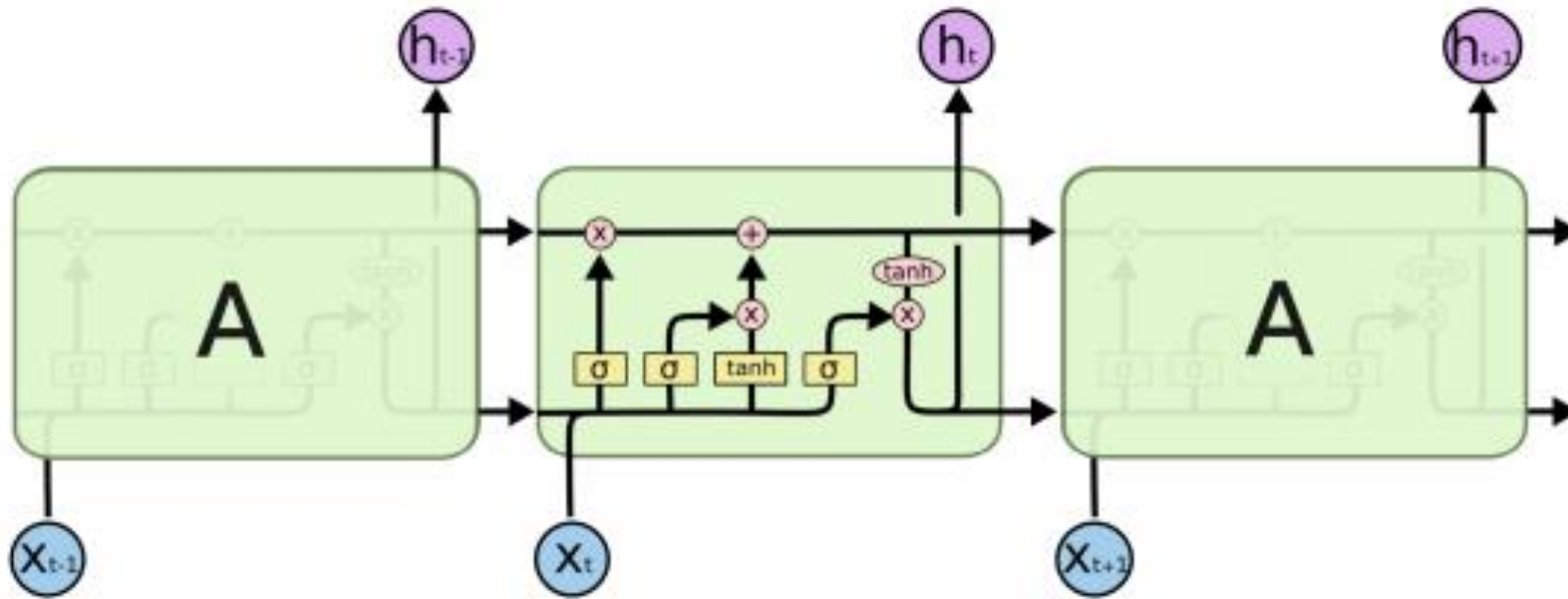
출처 : CS224n (2021) Lecture 6

Long Short-Term Memory RNNs (LSTMs)

- RNN의 Vanishing Gradient Problem을 해결하기 위해서는 어떻게 해야 할까?
- Vanishing Gradient – 특정 step을 지나가면 이전 정보를 잘 전달하지 못한다.
- 그렇다면 **cell state**를 추가해 **메모리의 역할**을 하게 하고, hidden state는 더 이상 이전 정보는 저장하지 않도록 해보자.

Long Short-Term Memory RNNs (LSTMs)

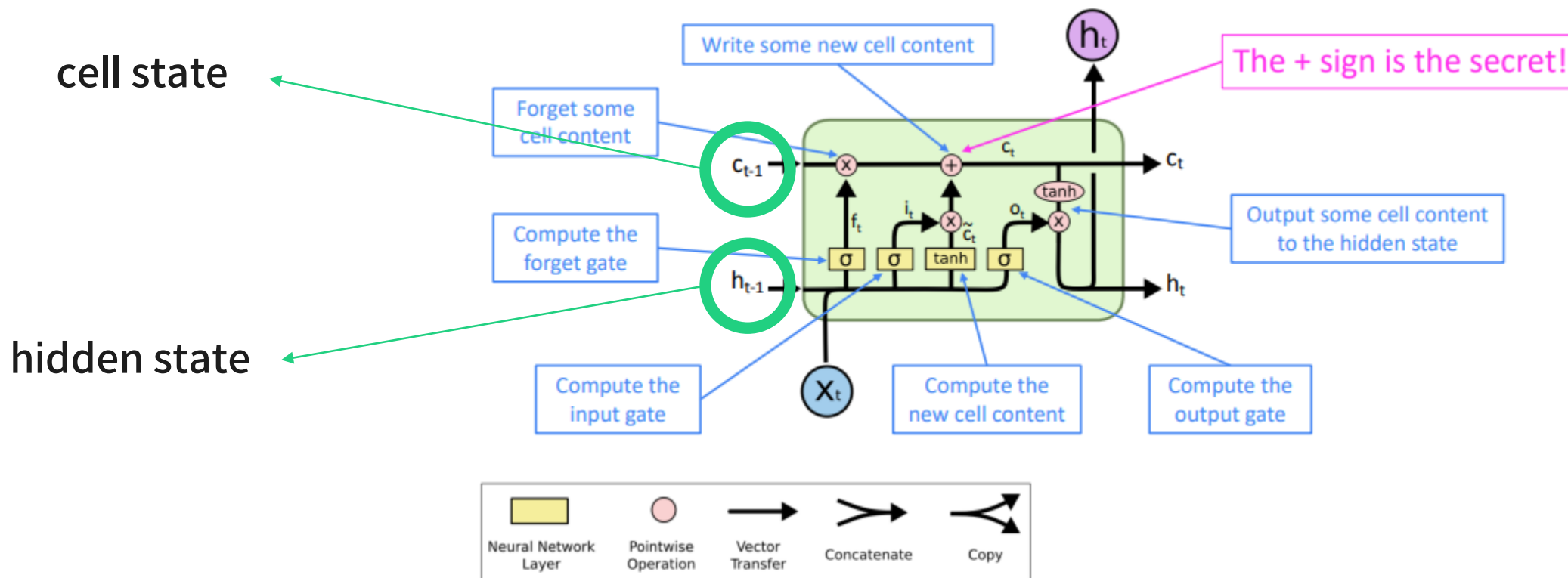
- LSTM의 구조



출처 : CS224n (2021) Lecture 6

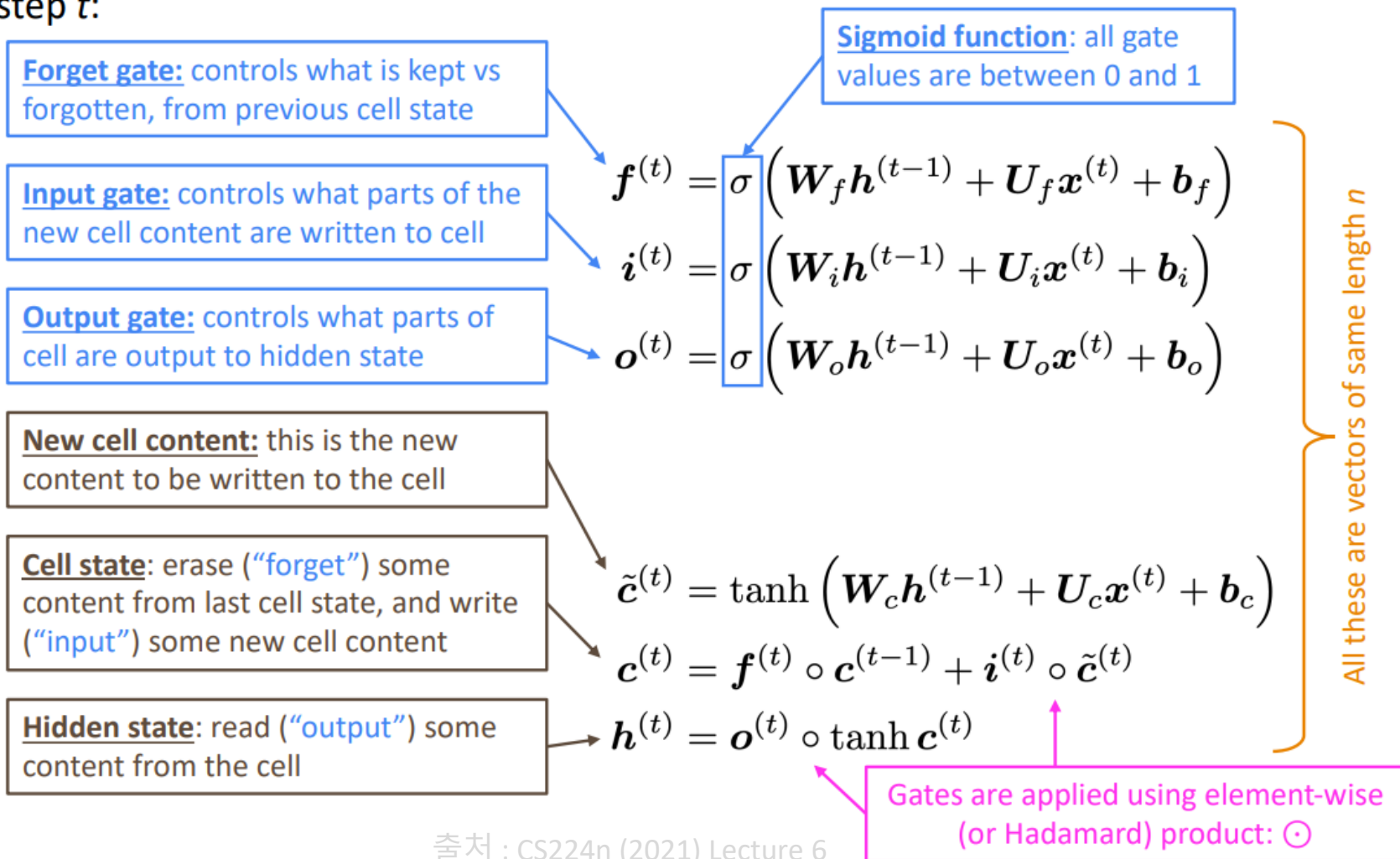
Long Short-Term Memory RNNs (LSTMs)

- LSTM에서 각 step에서의 computation
 - 가장 중요한 것은 RNN의 hidden state에서 이전 정보만 따로 기억하는 cell state가 존재



Long Short-Term Memory RNNs (LSTMs)

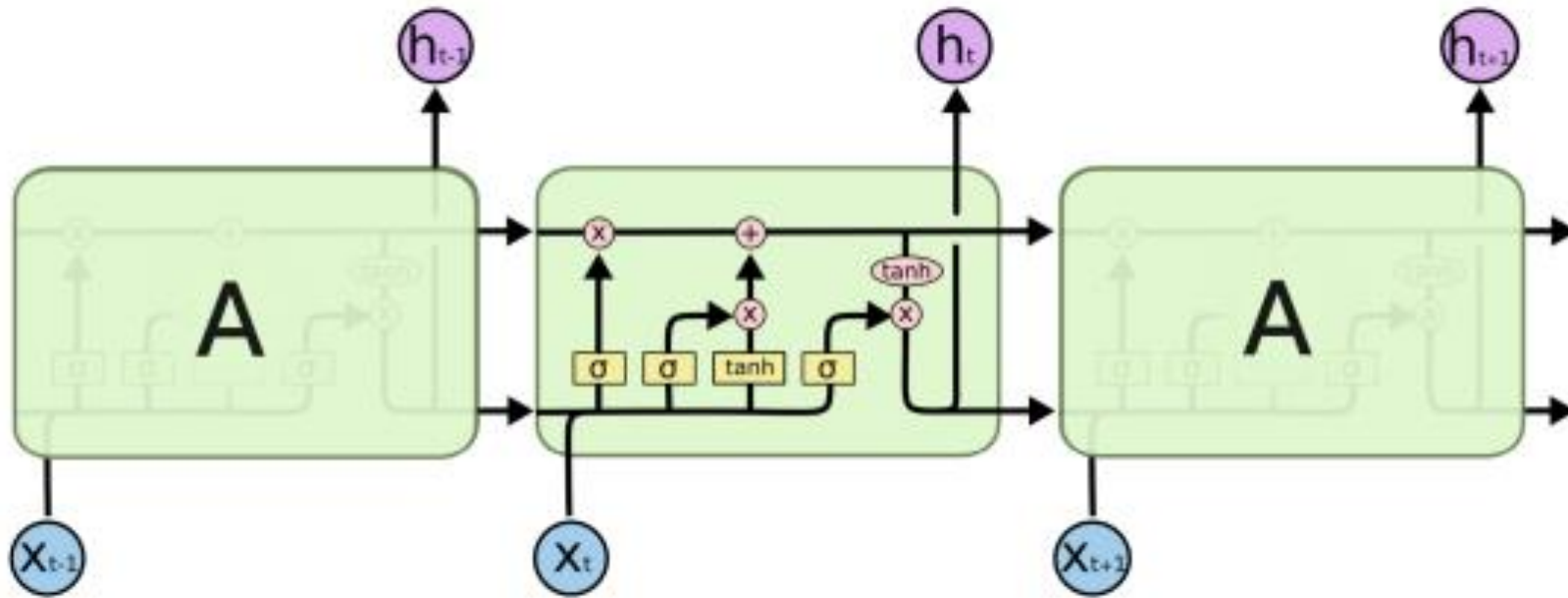
We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t :



출처 : CS224n (2021) Lecture 6

More about LSTM

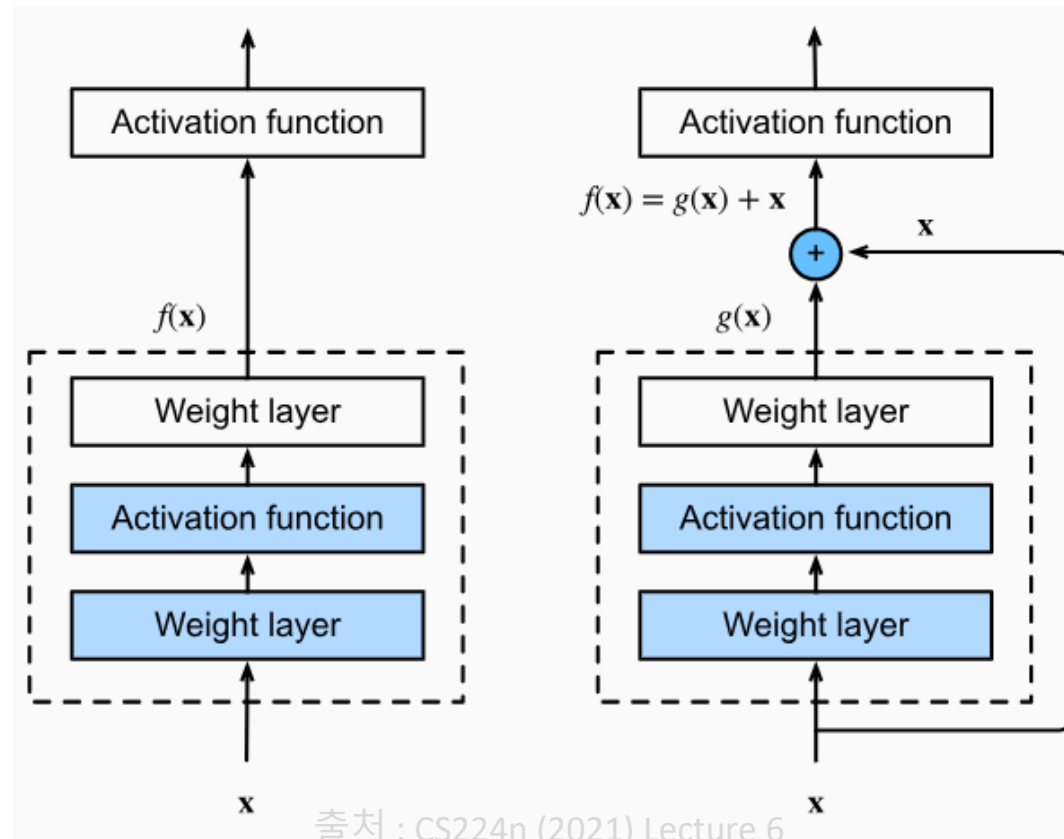
- <https://wikidocs.net/22888>



출처 : CS224n (2021) Lecture 6

Recap - ResNet

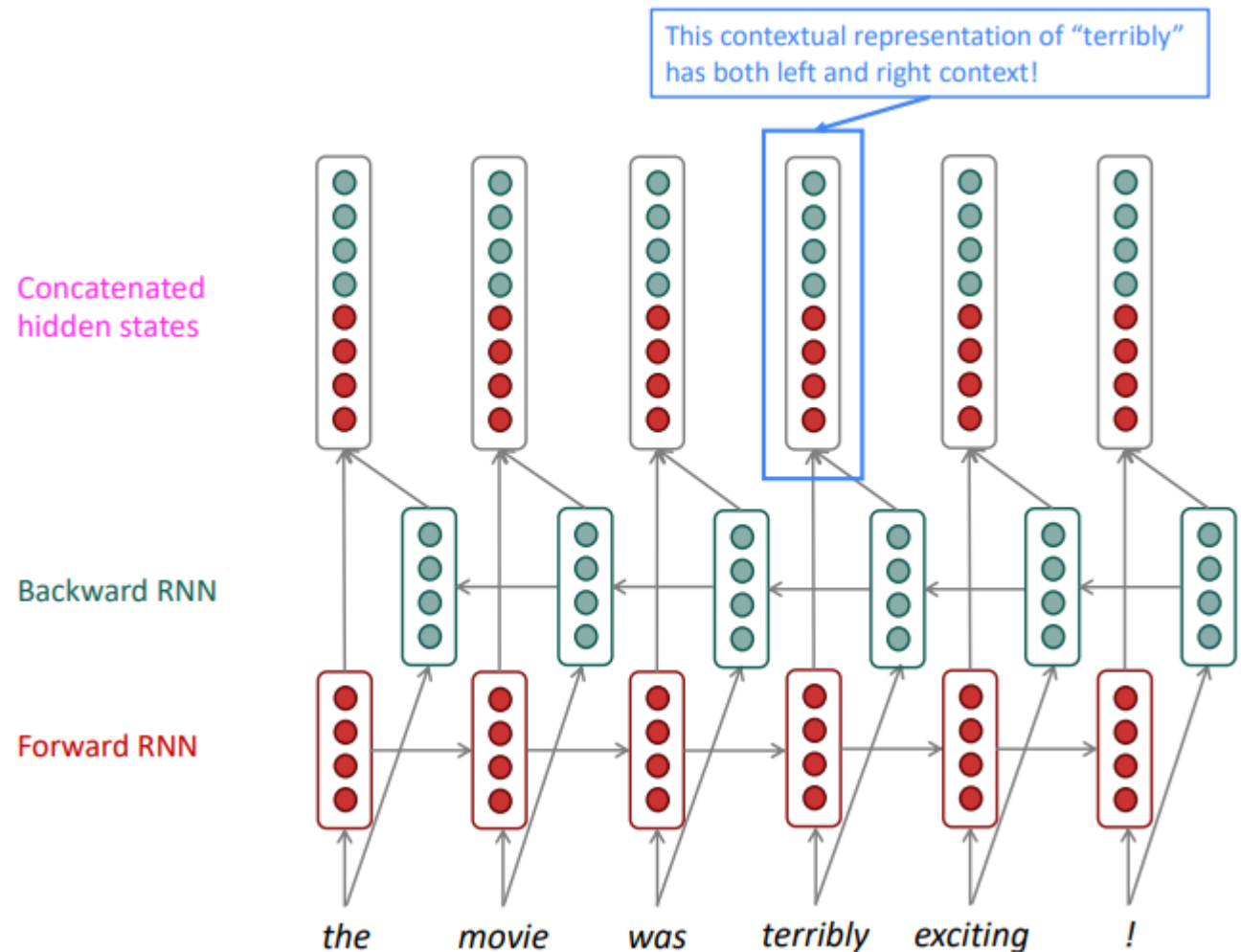
- Vanishing/Exploding Gradient Problem은 단순히 RNN만의 문제는 아니다.
- All neural architectures, especially **very deep** ones.



출처 : CS224n (2021) Lecture 6

Bidirectional RNNs

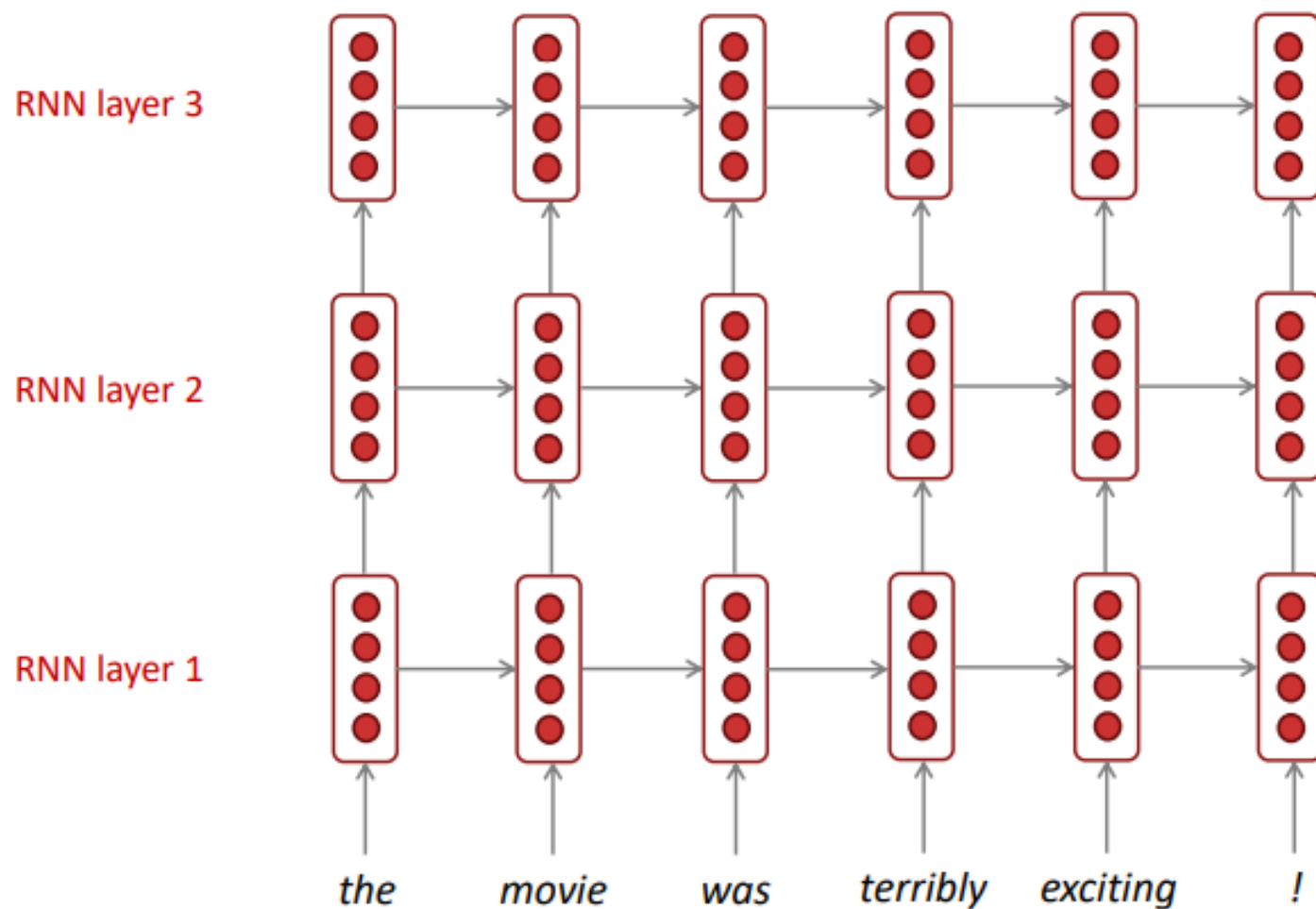
- 특정 task에서는 한 방향으로만 읽는 것이 옳지 않을 수 있다.
- 하나의 input 단어마다 순방향과 역방향으로 RNN을 적용했을 때 각각의 hidden state가 존재
- 2개의 hidden state를 이어 붙여 (Concatenate) 해당 단어의 hidden state로 계산



출처 : CS224n (2021) Lecture 6

Multi-layer RNNs

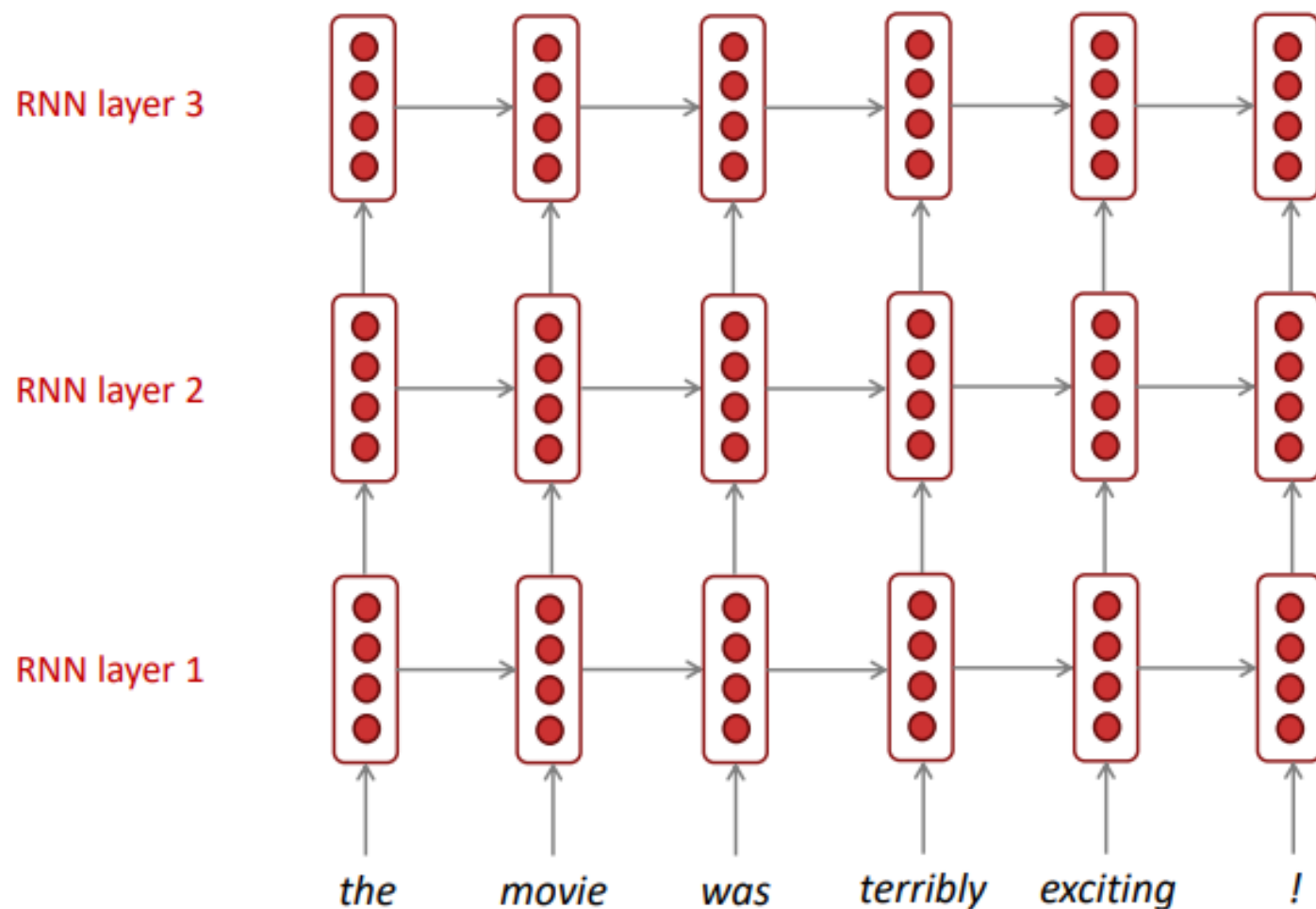
- RNN은 이미 가로로 굉장히 deep neural network (시간 측면)
- RNN layer를 위로 쌓아보자
- Multi-layer RNNs은 stacked RNN이라고도 불린다.



출처 : CS224n (2021) Lecture 6

Multi-layer RNNs

- Lower RNNs은 주로 lower-level feature를, higher RNNs는 주로 high-level feature를 추출
- Neural Machine Translation
 - Encoder: 2~4 layers
 - Decoder: 4 layers
- Transformer-based networks
 - 12 or 24 layers



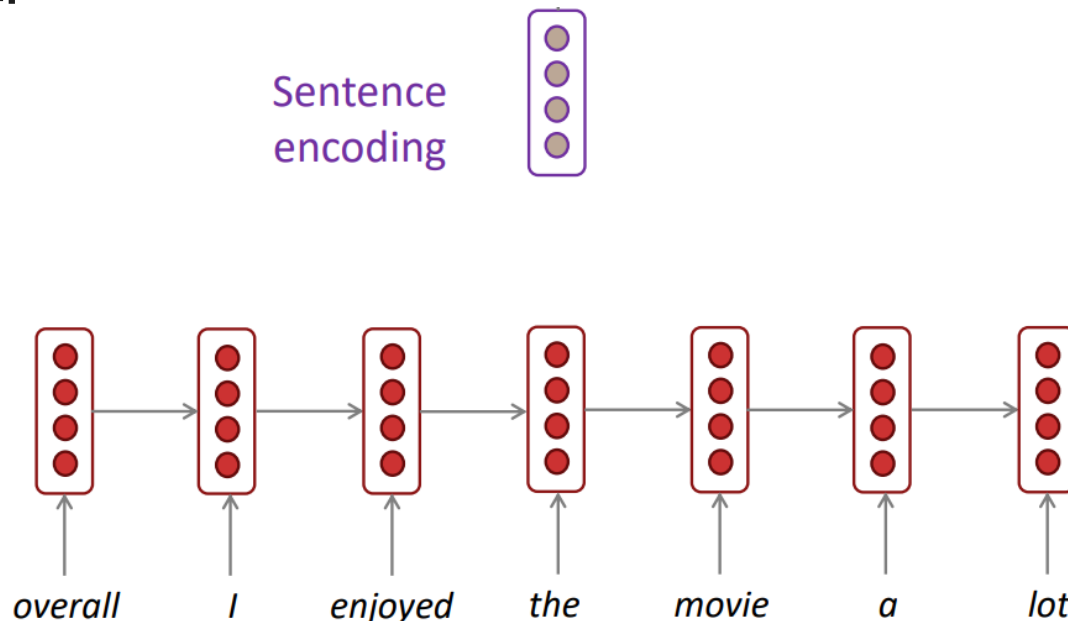
출처 : CS224n (2021) Lecture 6

Discussion questions

- RNN은 아래와 같은 문제에 잘못된 답을 낸다. 그 이유를 Vanishing Gradient Problem의 관점에서 각각 설명해보자.
 - When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _____.
 - Answer: tickets / RNN's prediction: printer
 - The writer of the books _____ ...
 - Answer: is / RNN's prediction: are

Discussion questions

- RNN을 통해 문장을 encoding할 수 있다. 마지막 hidden state를 문장의 encoding으로 보는 방법이 있는데 이 외에 어떤 방법들이 있을까? (하나는 강의 자료에 힌트가 있습니다!)
- 이 방법들 중에 어떤 것이 좋을까? 항상 좋은 방법이 있을까? 아님 상황에 따라? 자유롭게 생각을 말해주세요!



감사합니다.