DeepIntoDeep

Recurrent Neural Networks

발표자: 박정규

3. **Recurrent** Neural Networks

Jungkyu Park

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

Intro

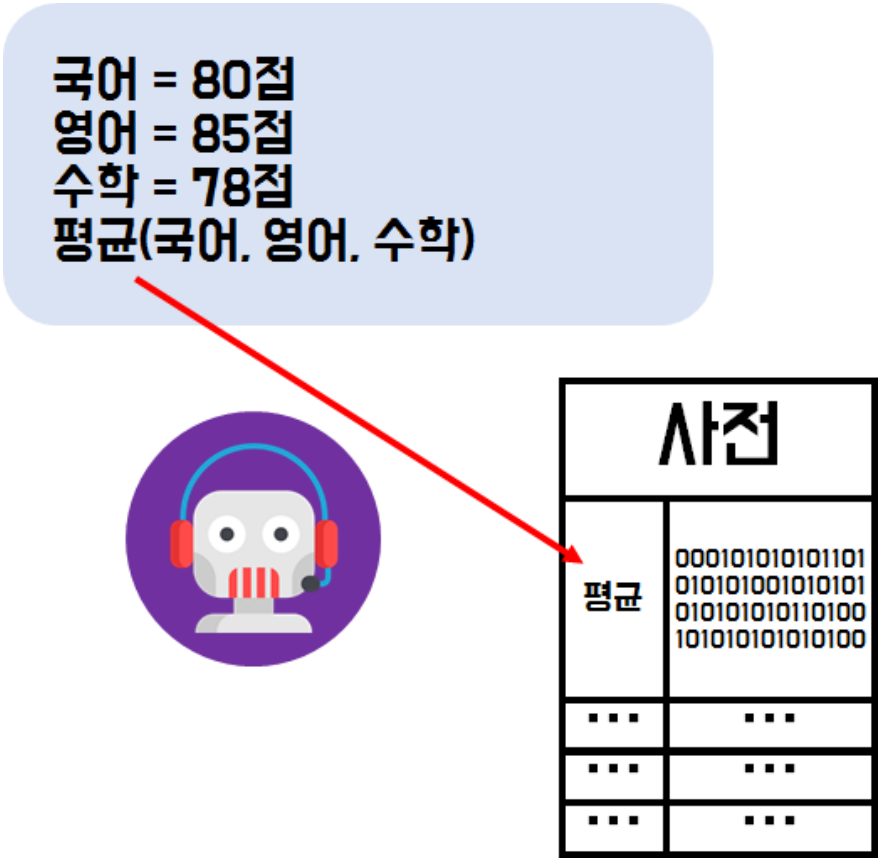
- 이번주는 NLP에 대해 배워봅시다
- 말로만 듣던 ‘attention’, ‘Transformer’가 뭐지… ‘attention’만 있으면 된다던데
- 딥러닝의 대표적인 분야 중 하나지만 처음 들으면 많이 어려울 수 있어요..!
  - D2D 목적에 맞게 어려운 내용은 최대한 배제하고 NLP가 어떤 느낌인지 알려주기 위해 열심히 준비했지만 그래도 질문이 있으면 언제든지 연락주세요!

Contents

- Word Vector
- Language Modeling
- RNNs
  - Basic Concept
  - Various tasks using RNNs
  - LSTMs

Word Vector

Can Computers Understand Language?



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) → Cosine Similarity = 0

  ⇒ 두 벡터는 관계성이 없다!

NaiveBayes Classifier

	Doc(d)	Document (words, w)	Class (c)
Training	1	Image recognition uses convolutional neural networks	CV
	2	Transformer can be used for image classification task	CV
	3	Language modeling uses transformer	NLP
	4	Document classification task is language task	NLP
Test	5	Classification task uses transformer	?

Word	Prob	Word	Prob
$P(w_{\text{"classification"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"classification"}}\|c_{\text{NLP}})$	$\frac{1}{10}$
$P(w_{\text{"task"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"task"}}\|c_{\text{NLP}})$	$\frac{2}{10}$
$P(w_{\text{"uses"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"uses"}}\|c_{\text{NLP}})$	$\frac{1}{10}$
$P(w_{\text{"transformer"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"transformer"}}\|c_{\text{NLP}})$	$\frac{1}{10}$

NaiveBayes Classifier

Word	Prob	Word	Prob
$P(w_{\text{"classification"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"classification"}}\|c_{\text{NLP}})$	$\frac{1}{10}$
$P(w_{\text{"task"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"task"}}\|c_{\text{NLP}})$	$\frac{2}{10}$
$P(w_{\text{"uses"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"uses"}}\|c_{\text{NLP}})$	$\frac{1}{10}$
$P(w_{\text{"transformer"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"transformer"}}\|c_{\text{NLP}})$	$\frac{1}{10}$

$$P(c_{CV}|d_5) = \frac{1}{2} \times \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14}, P(c_{NLP}|d_5) = \frac{1}{2} \times \frac{1}{10} \times \frac{2}{10} \times \frac{1}{10} \times \frac{1}{10}$$

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***

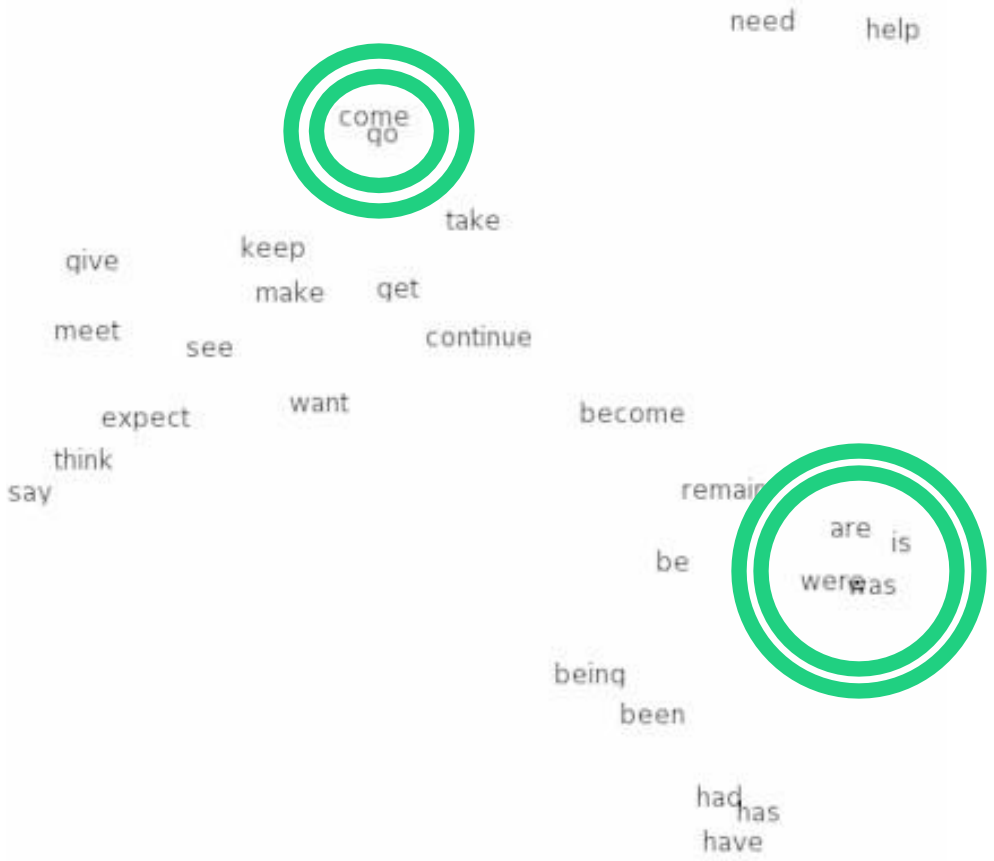
출처 : CS224n (2021) Lecture 1

Word Vectors (word embeddings)

- Distributional Semantics: 단어의 뜻은 주변에 자주 나타나는 단어들로 표현

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

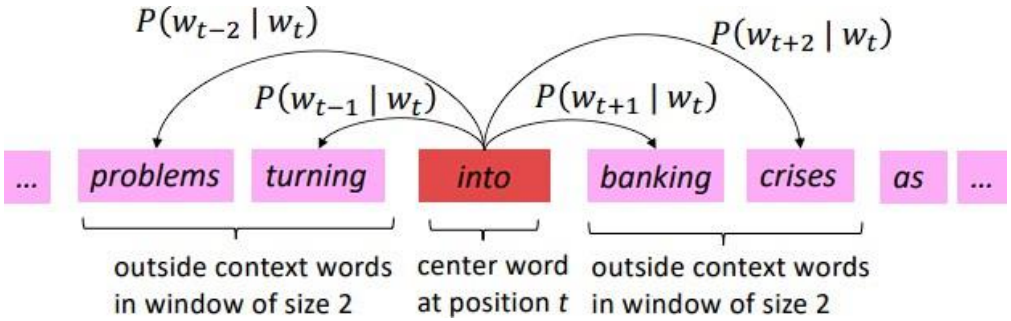
출처 : CS224n (2021) Lecture 1



출처 : CS224n (2021) Lecture 1

How to train word vectors?

- Word2Vec



출처 : CS224n (2021) Lecture 1

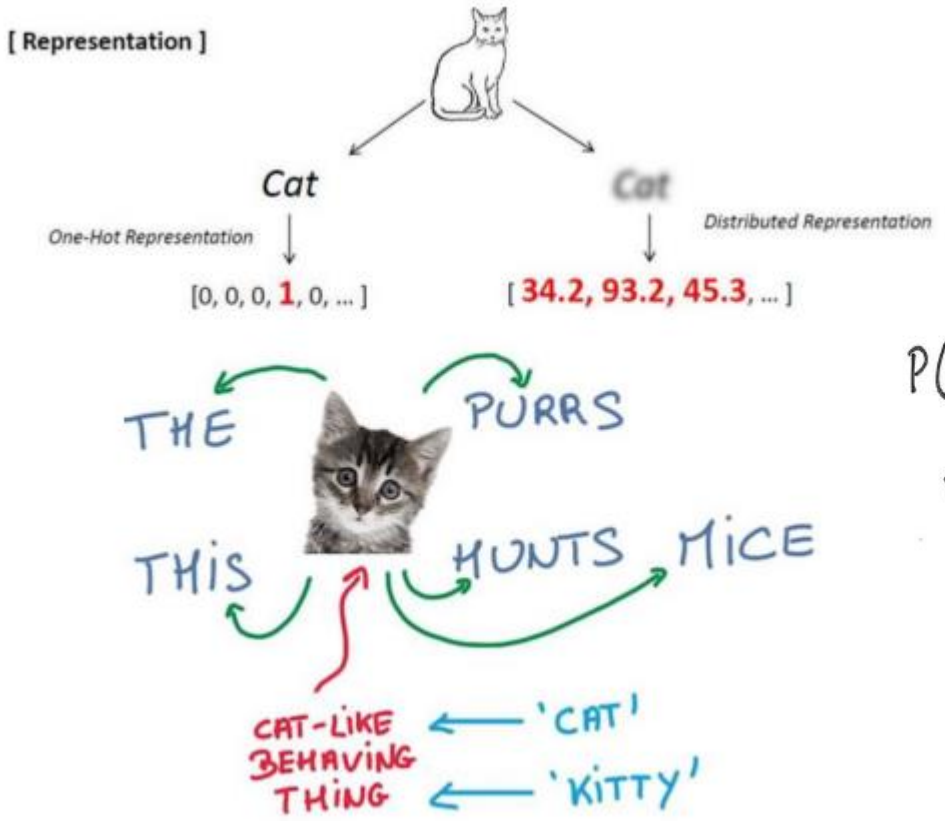
- 중심 단어를 기반으로 주변 단어를 예측하기
- 주변 단어를 기반으로 중심 단어를 예측하기

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 …

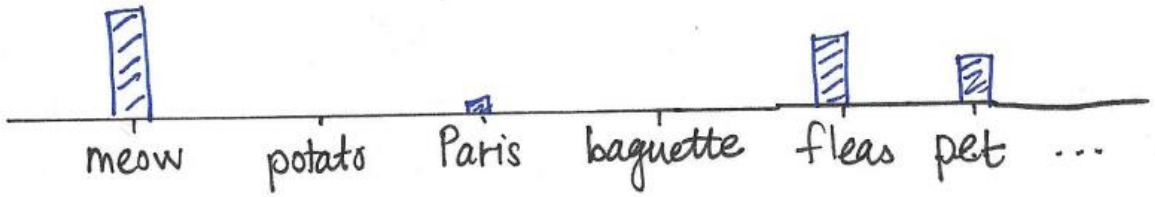
How to train word vectors?

- Word2Vec



$P(w|\text{"cat"})$

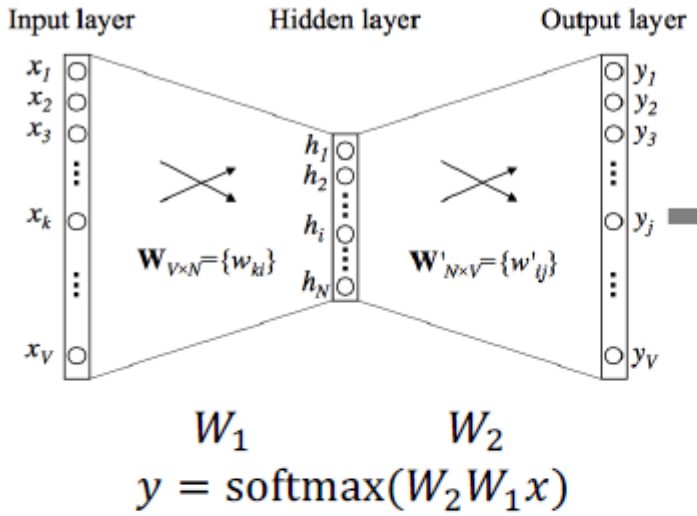
w

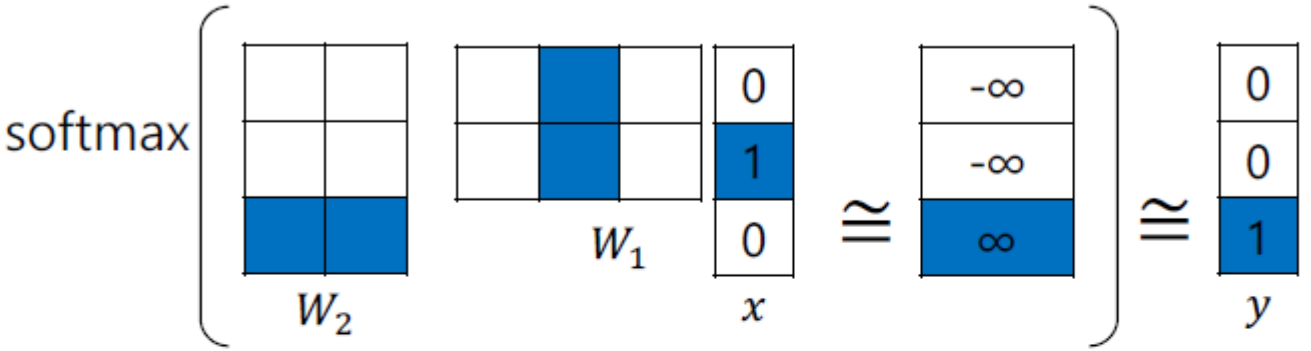


출처 : Distributed Representations of Words and Phrases and their Compositionality, NeurIPS'13

How to train word vectors?

- Word2Vec





- **Sentence** : "I study math."
- **Vocabulary**: {"I", "study" "math"}
- **Input**: "study" [0, 1, 0]
- **Output**: "math" [0, 0, 1]
- Columns of W_1 and rows of W_2 represent each word

- E.g., 'study' vector : 2nd column in W_1, 'math' vector : 3rd row in W_2.
- The 'study' vector in W_1 and the 'math' vector in W_2 should have a high inner-product value.

출처 : Distributed Representations of Words and Phrases and their Compositionality, NeurIPS'13

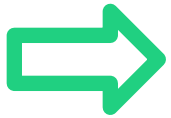
How to train word vectors?

- 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

출처: CS224n (2021) Lecture 2

$$J(\theta) = \frac{1}{2}\sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$



Problem

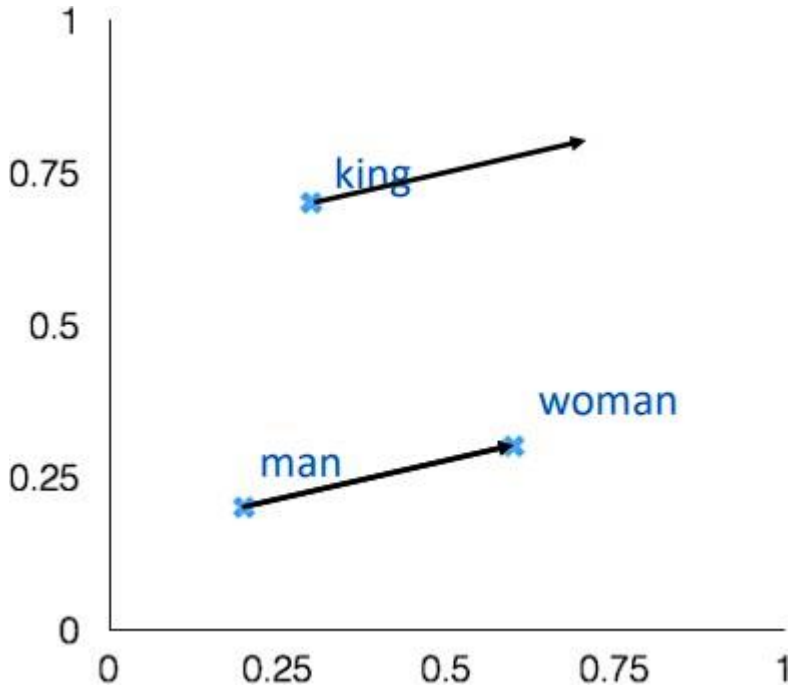
1. 행렬의 크기
   - 문장 속에 등장하는 단어의 수가 많으면?

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

출처 : GloVe: Global Vectors for Word Representation, EMNLP'14

Word Vectors

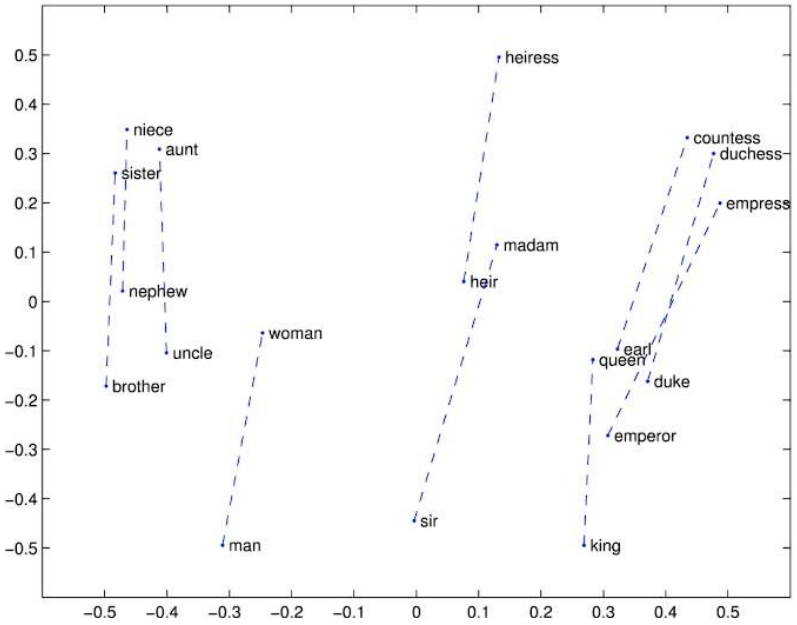
- Word Vector Analogies – 단어들 간의 관계를 잘 유추
  - Man : Woman = King : ??? ('King' – 'Man' + 'Woman' = ???)



출처 : CS224n (2021) Lecture 2

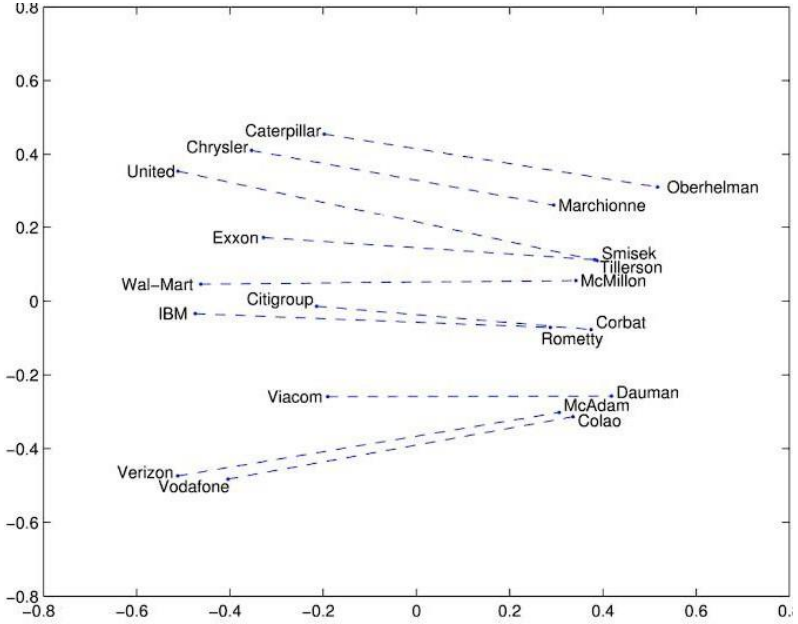
Word Vectors

Man - Woman



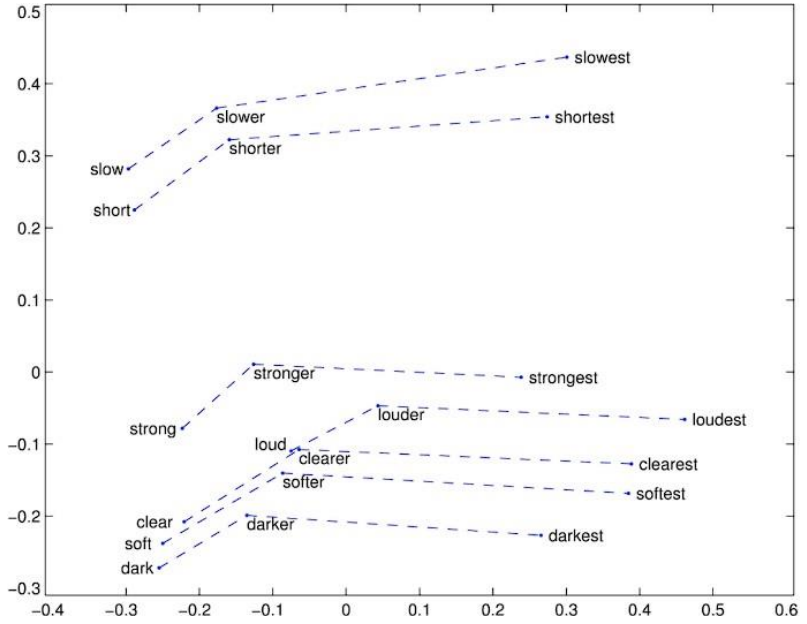
출처 : CS224n (2021) Lecture 2

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 _____

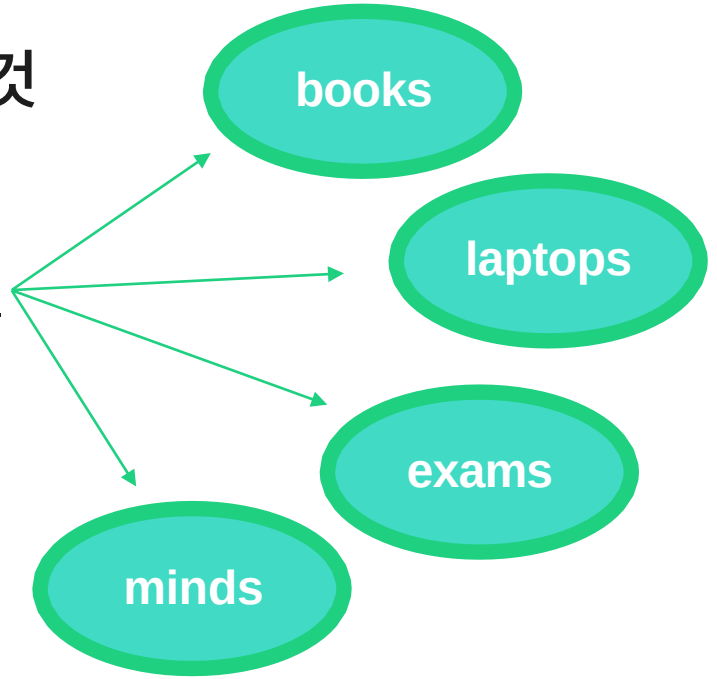
- 문장을 읽을 때는 앞 내용이 중요하다!

Language Modeling

- Language Modeling: 다음에 올 단어를 예측하는 것

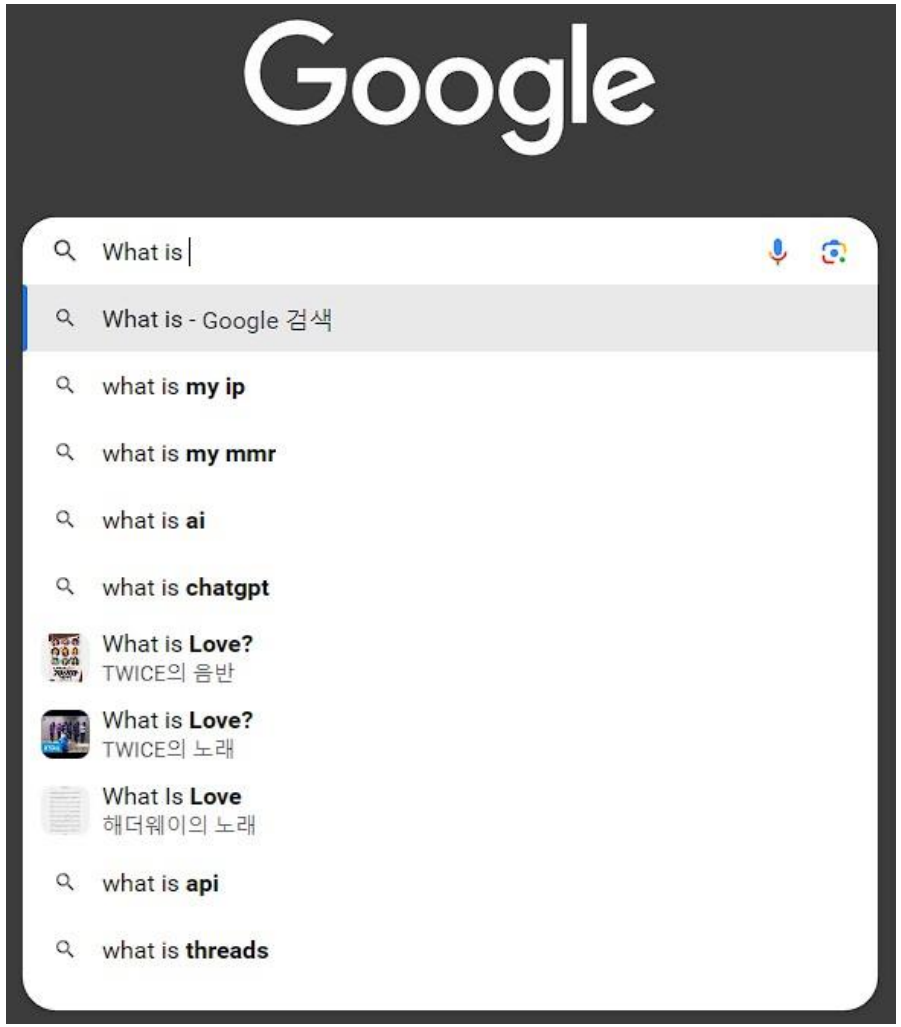
The students opened their ____

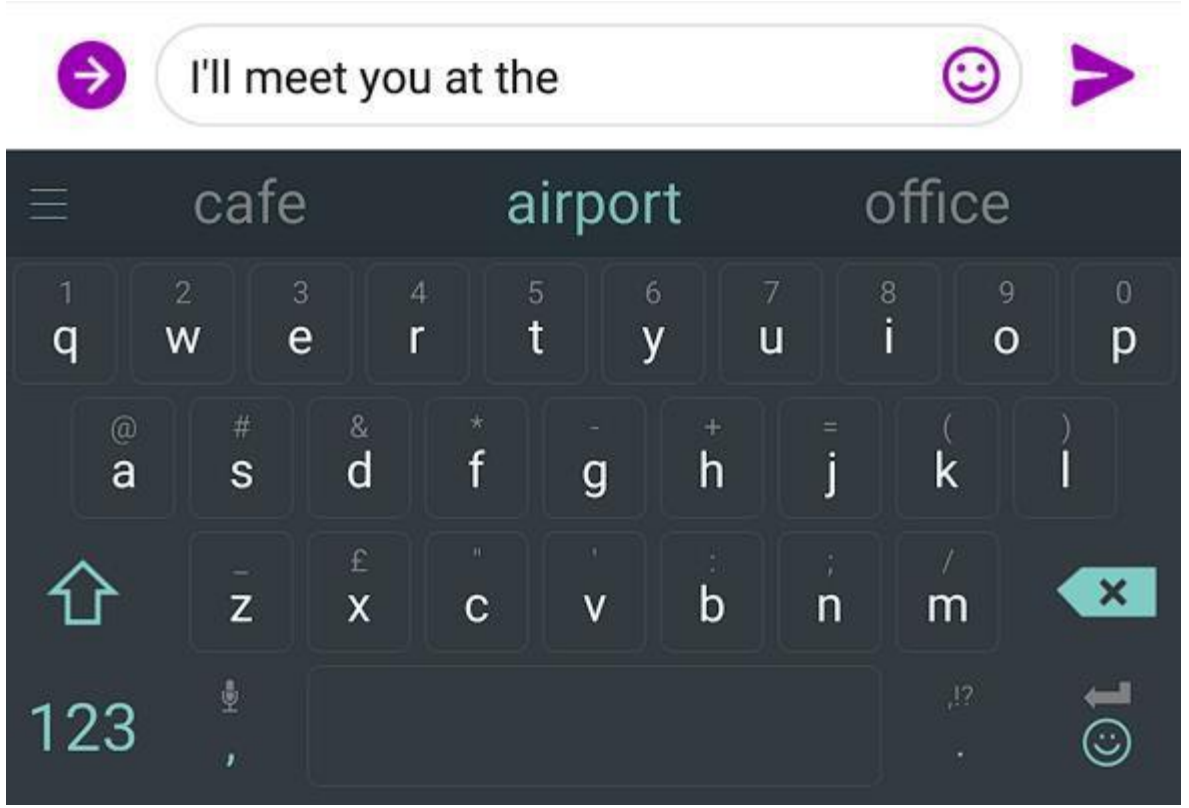
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$



- $P\left(x^{(t+1)} \middle| x^{(t)}, \dots, x^{(1)}\right)$
  - $x^{(1)}, x^{(2)}, \cdots, x^{(t)}$라는 t개의 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-1개의 단어에 의해 결정
- Assumption:
  - $P\left(x^{(t+1)}\middle|x^{(t)},\ldots,x^{(1)}\right) = P(x^{(t+1)}|x^{(t)},\ldots,x^{(t-n+2)})$

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

- Example: 4-gram Language Model
  - ~~As the proctor started the clock, the~~ students opened their _____
  - $P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$

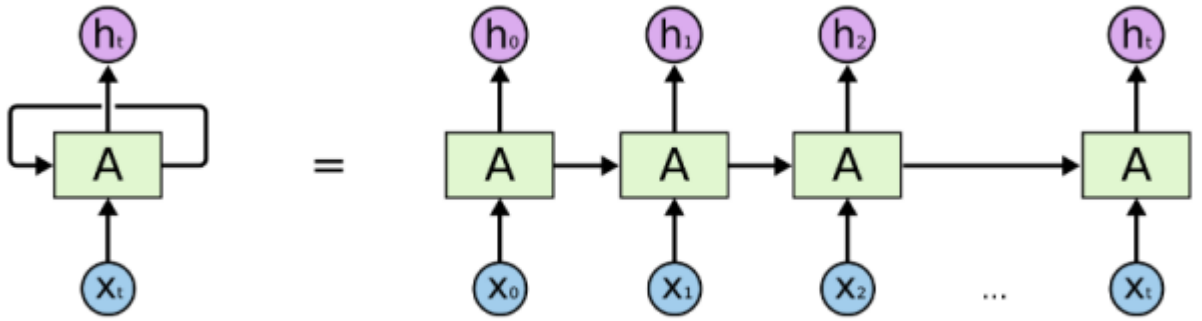
n-gram Language Models

- 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 → $P(books|students\ opened\ their) = 0.4$
    - "Students opened their exams": 100개 → $P(exams|students\ opened\ their) = 0.1$
- Problem
  - Sparsity Problem
  - Storage Problem

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.

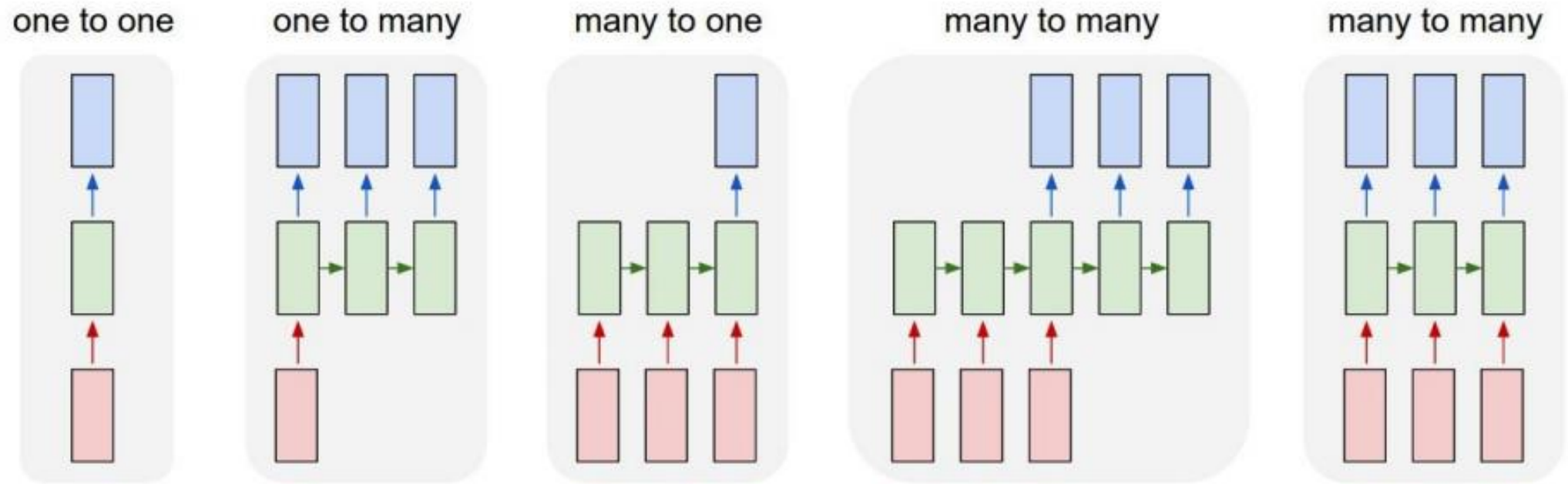


An unrolled recurrent neural network.

출처 : https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks (RNN)

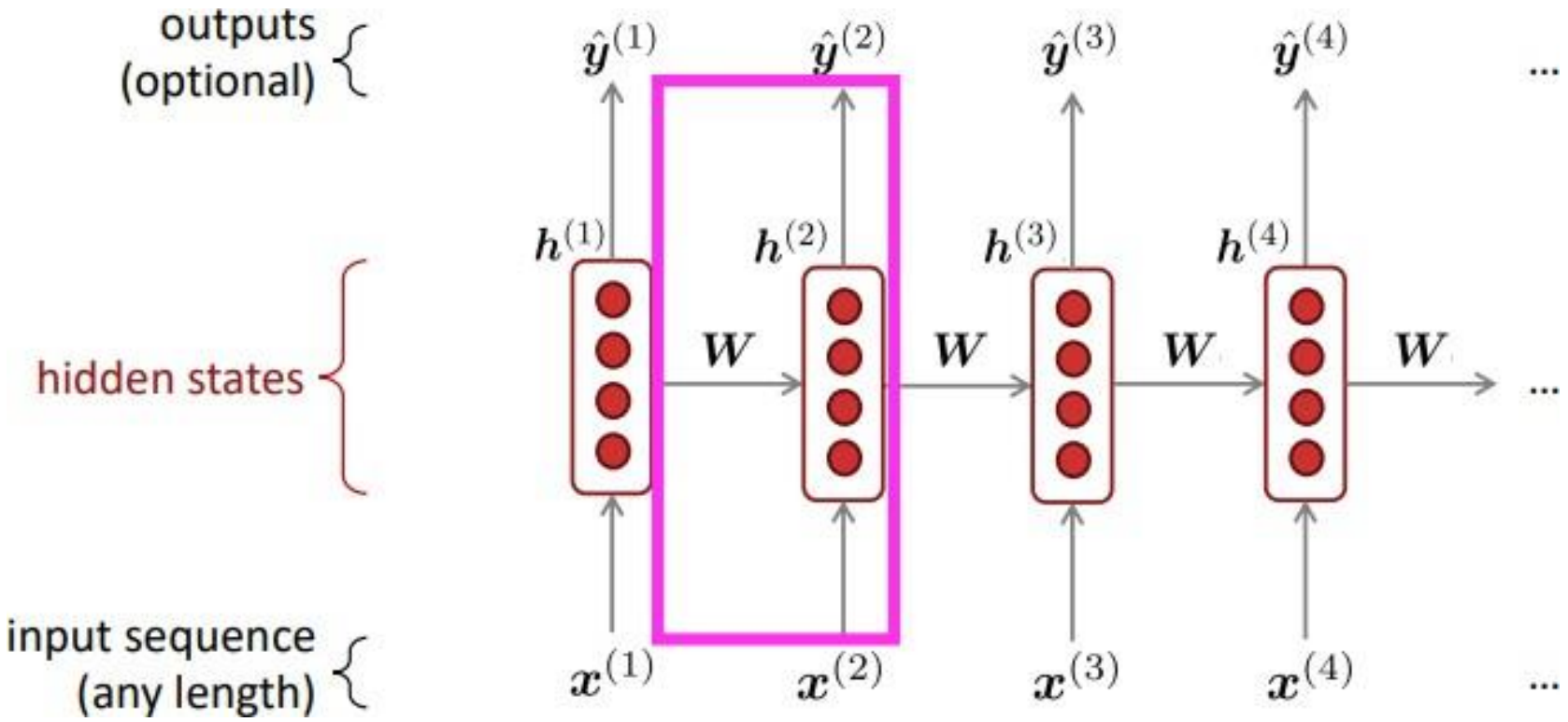
- RNN의 다양한 형태
  - Input과 Output의 길이를 유동적으로 설계할 수 있다.



출처: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Recurrent Neural Networks (RNN)

- Input을 처리할 때, 같은 weights W를 반복적으로 적용하자



출처 : CS224n (2021) Lecture 5

Recurrent Neural Networks (RNN)

Output distribution

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

Hidden layer

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

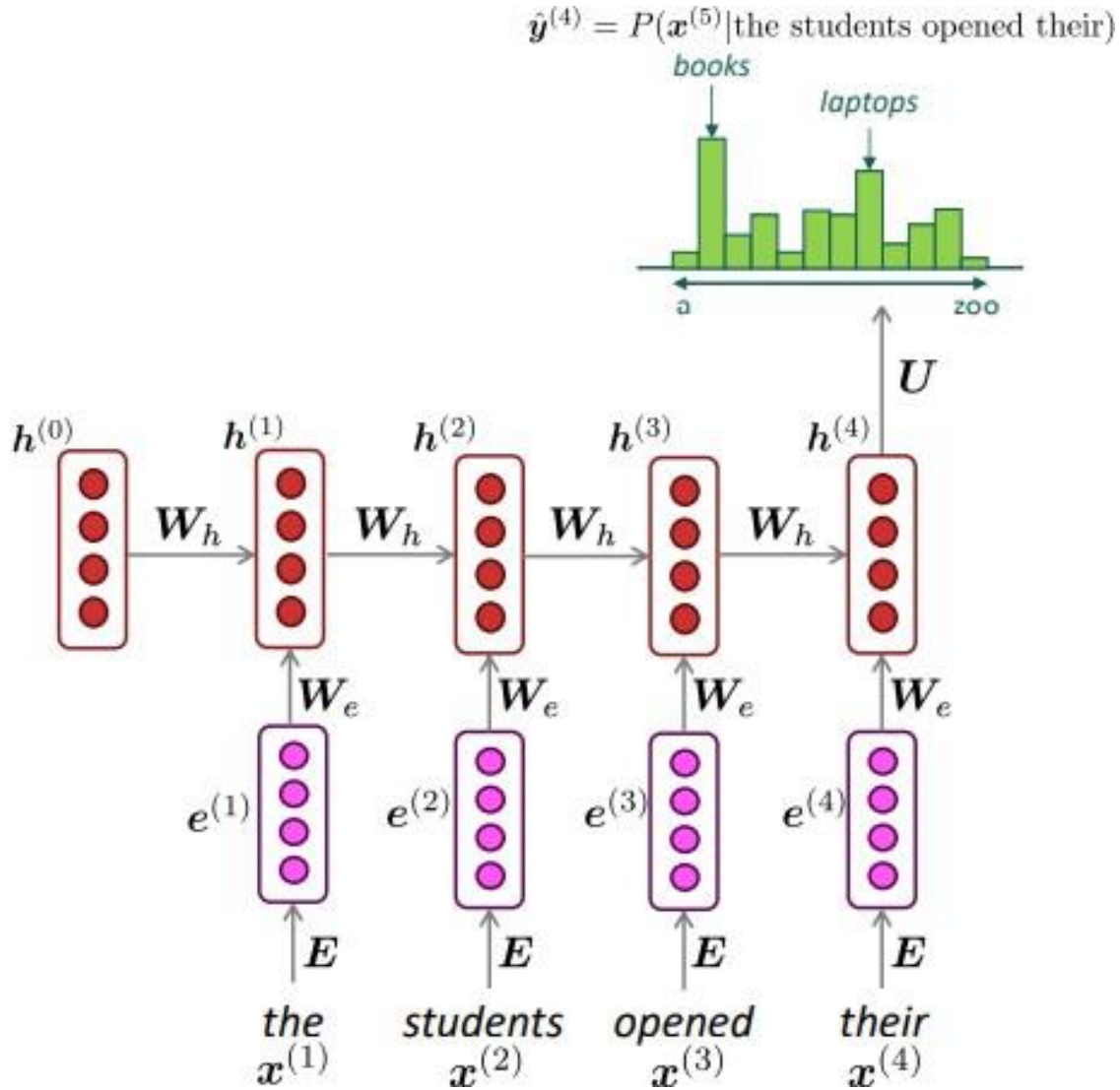
$h^{(0)}$: initial hidden state

Concatenated word embeddings

$$e^{(t)} = Ex^{(t)}$$

words / one-hot vectors

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

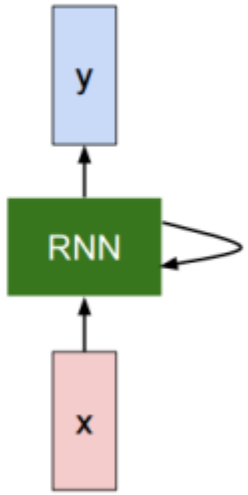


출처 : CS224n (2021) Lecture 5

Recurrent Neural Networks (RNN)

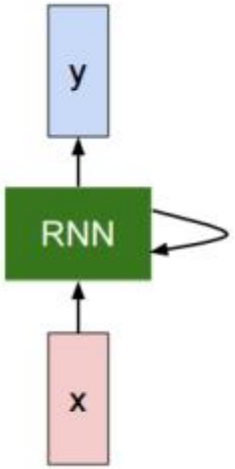
How to calculate the hidden state

$$h_t = f_W(h_{t-1}, x_t)$$



- h_{t-1}: old hidden-state vector
- x_t: input vector at some time step
- h_t: new hidden-state vector
- f_W: RNN function with parameters W
- y_t: output vector at time step t

Recurrent Neural Networks (RNN)

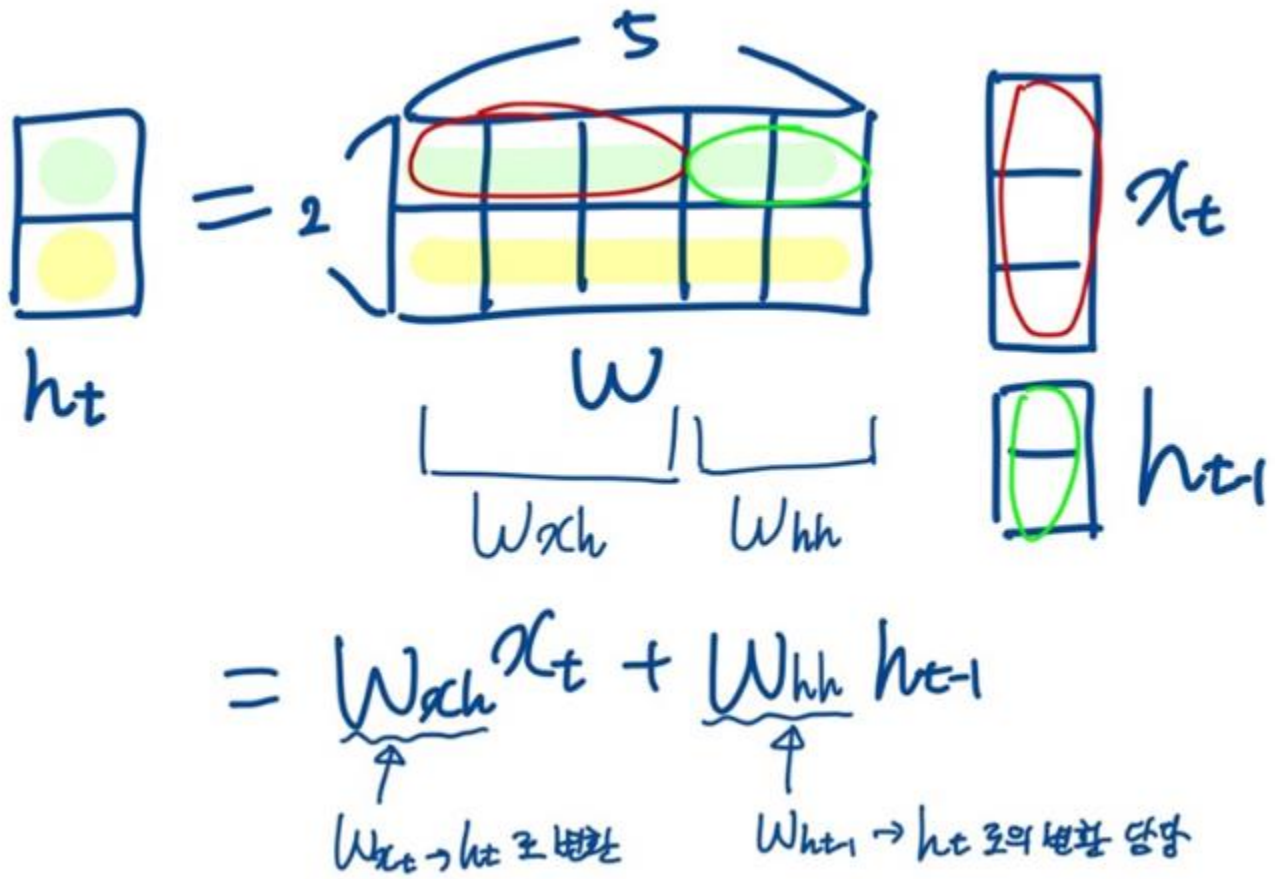


$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

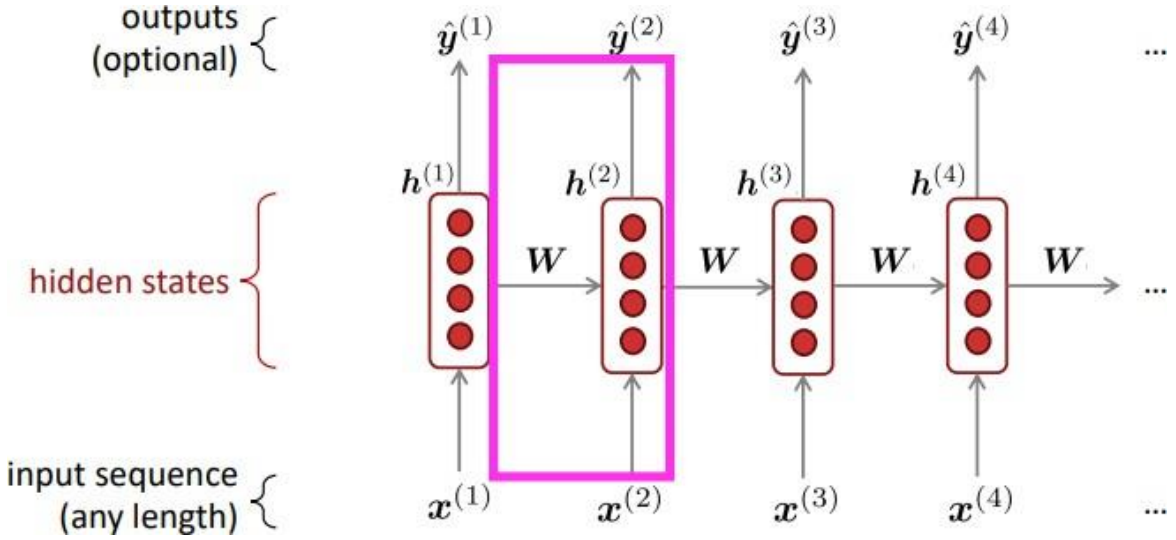
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$



출처 : https://taeuk-kim.tistory.com/26?category=886962

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

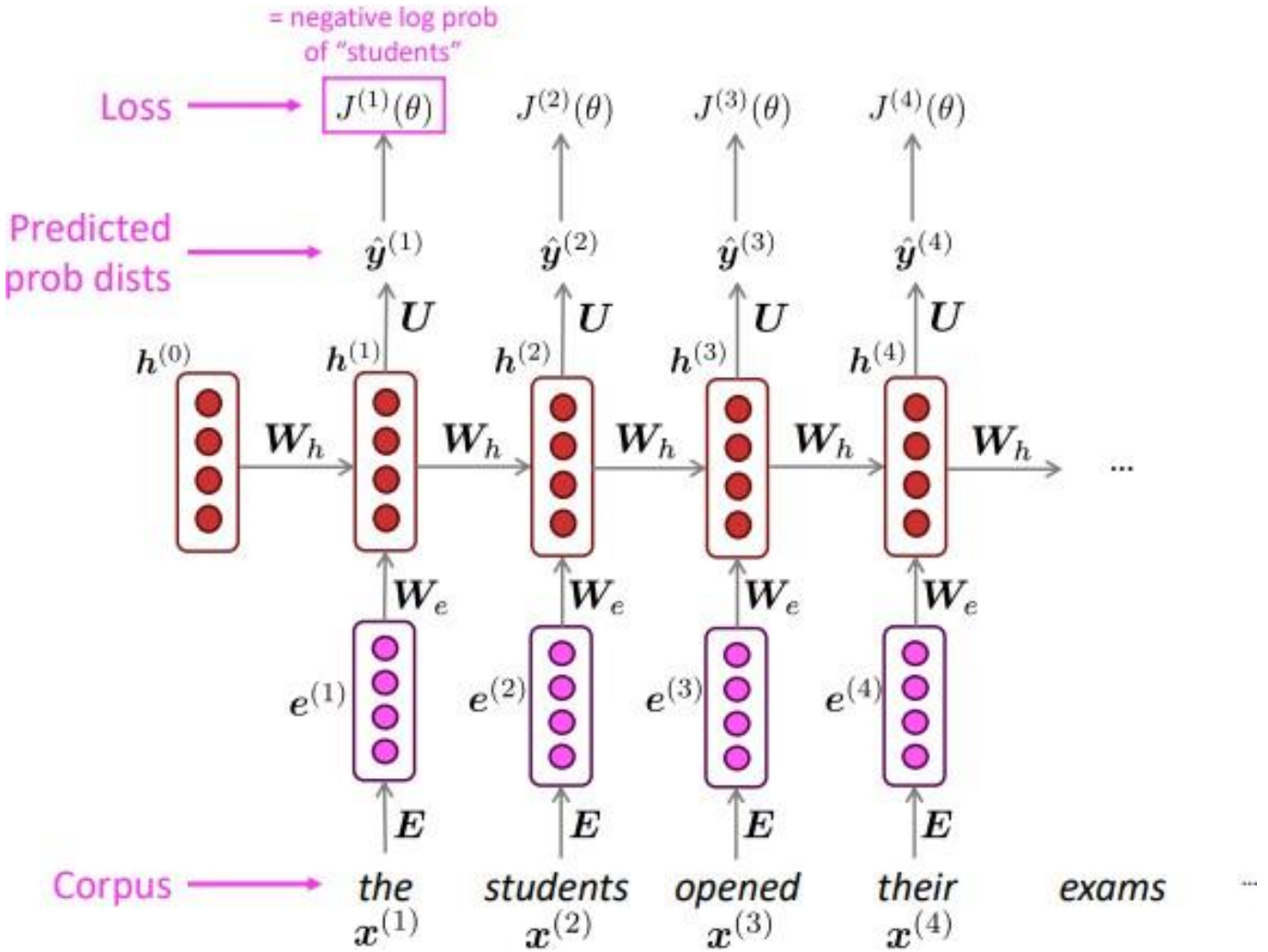
Training an RNN Language Model

- 각 step t마다 output distribution $\hat{y}^{(t)}$를 계산
  - t-1개의 단어들을 바탕으로 t번째 단어의 확률을 예측

- 각 step t마다 예측한 값과 실제 정답을 비교
  - 이 때, Loss function은 Cross-Entropy Loss를 사용
  - $J^{(t)}\theta = CE(y^{(t)}, \hat{y}^{(t)} = -\sum_{w \in V} y_w^{(t)} \log \hat{y}_w^{(t)}$

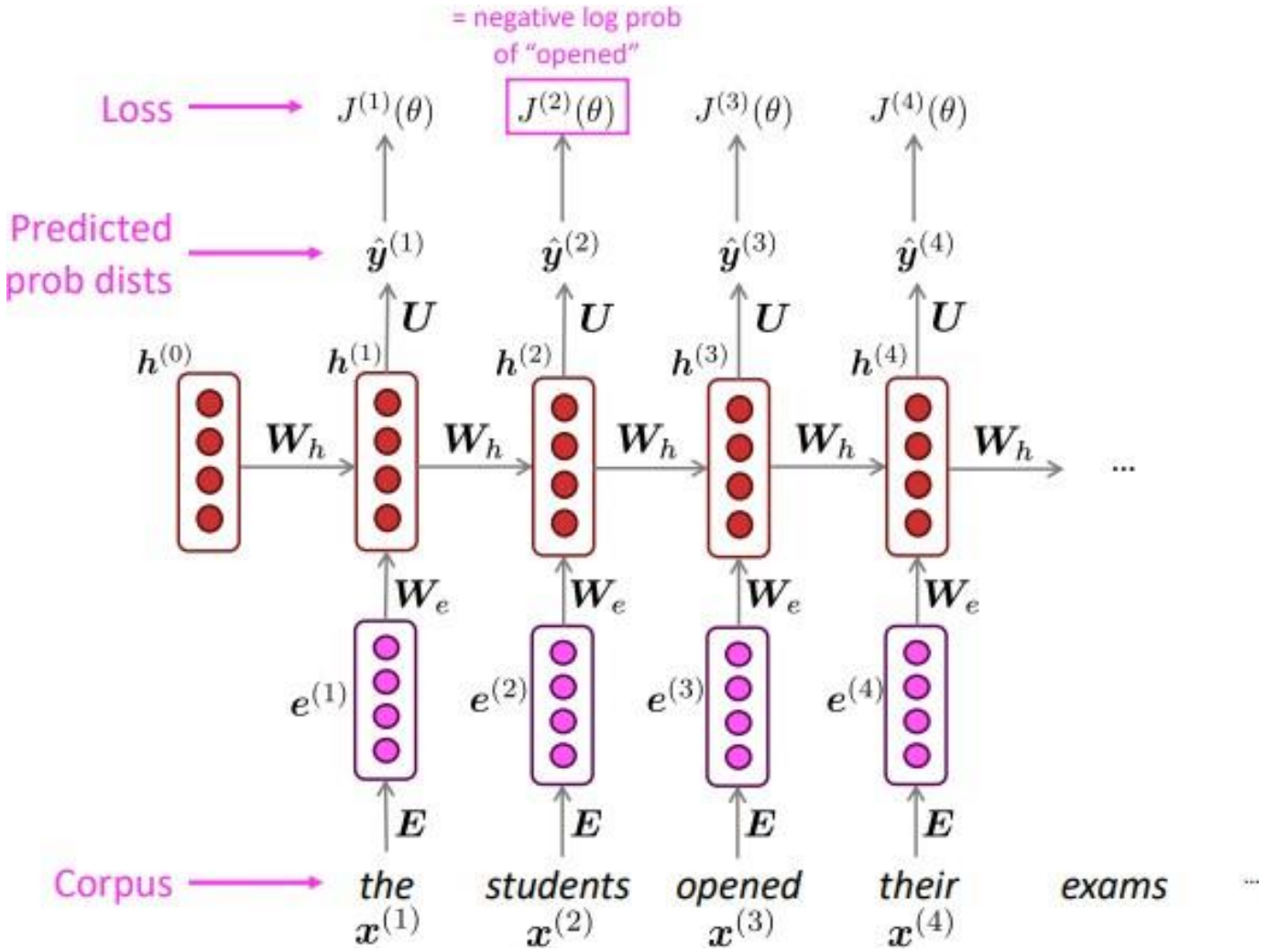
- 전체 학습 데이터에 대한 Loss는 각 Loss들의 평균
  - $J(\theta) = \frac{1}{T}\sum^{T} J^{(t)}(\theta)$

Training an RNN Language Model



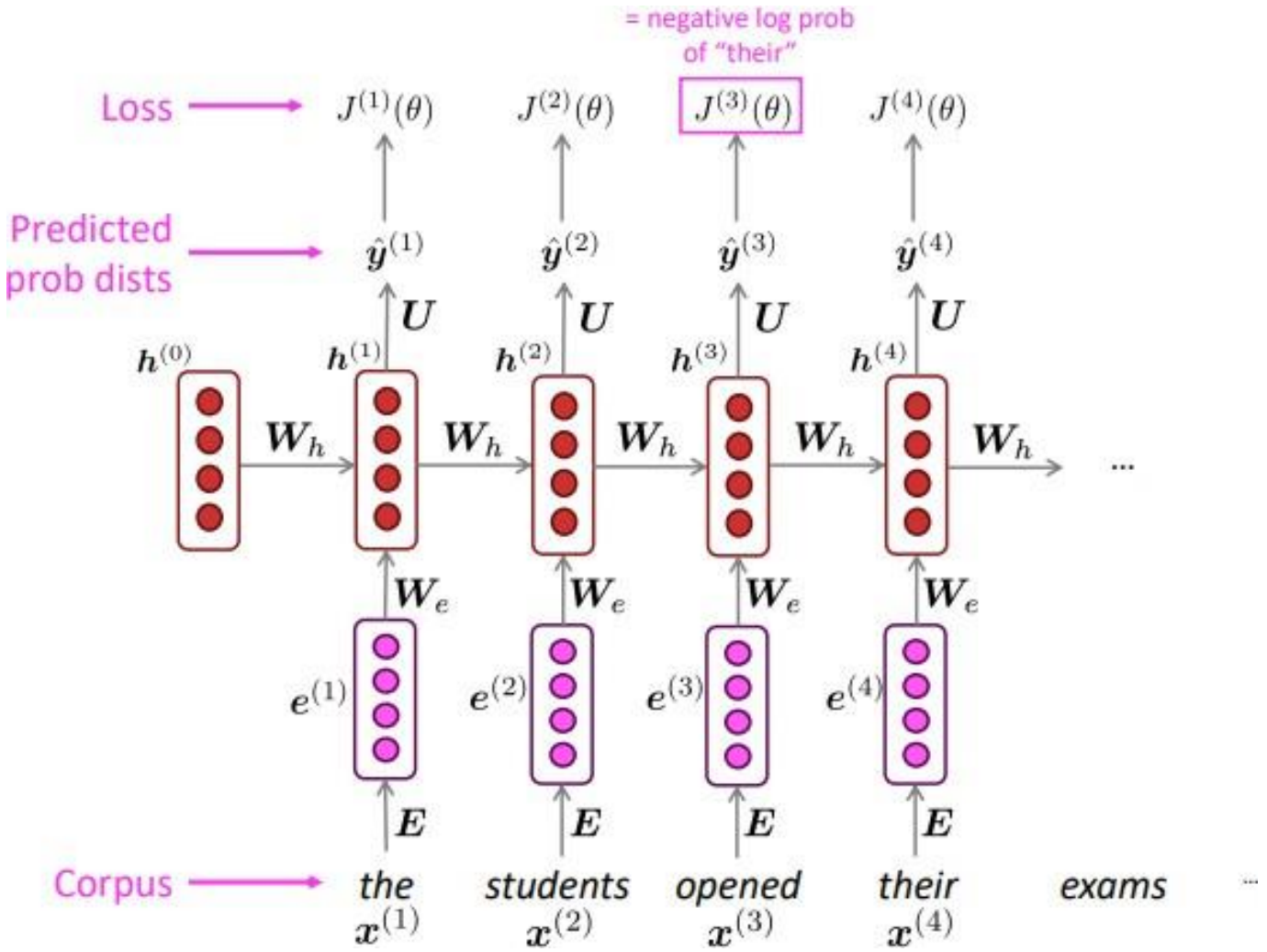
출처 : CS224n (2021)

Training an RNN Language Model



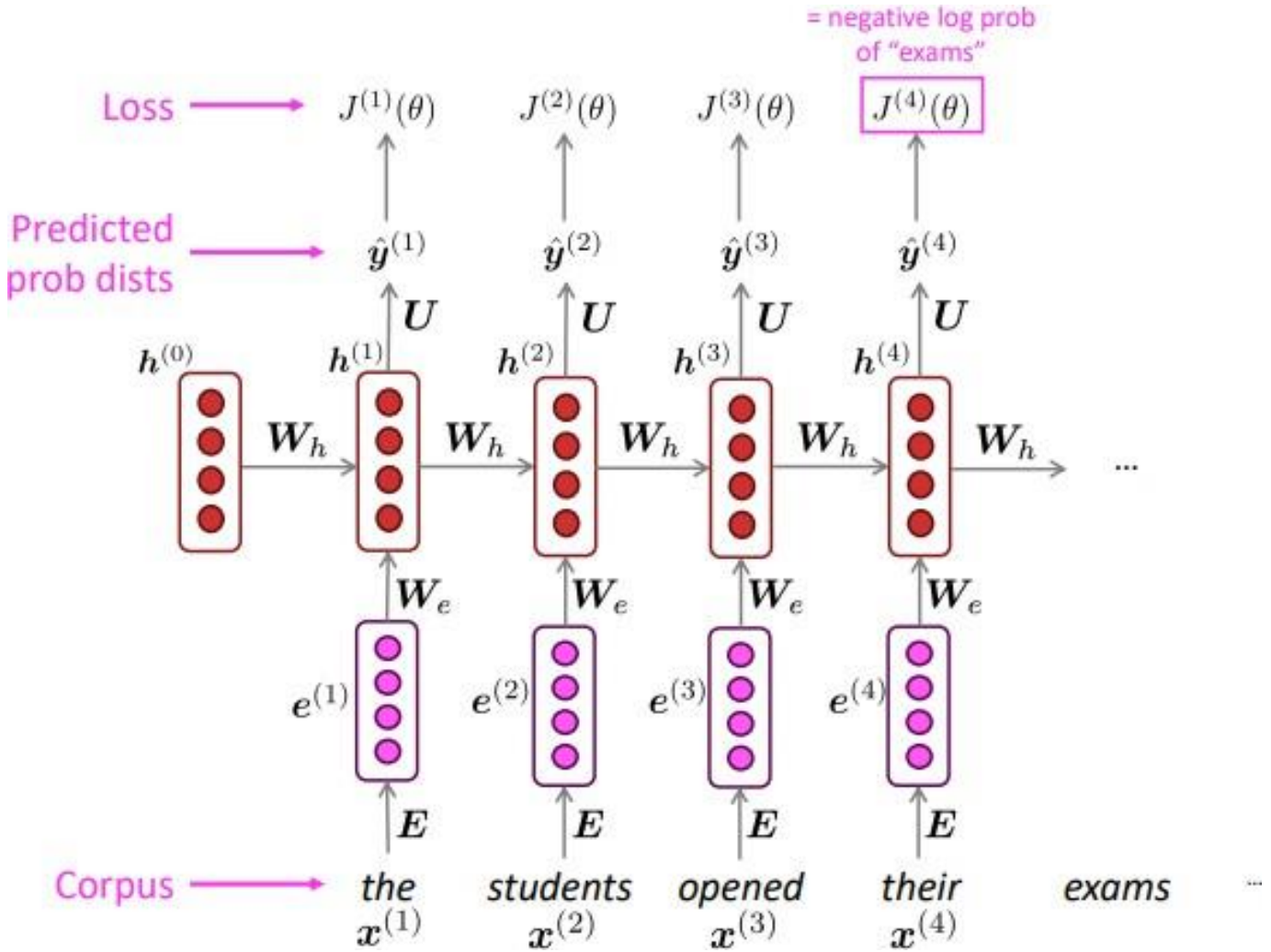
출처 : CS224n (2021)

Training an RNN Language Model



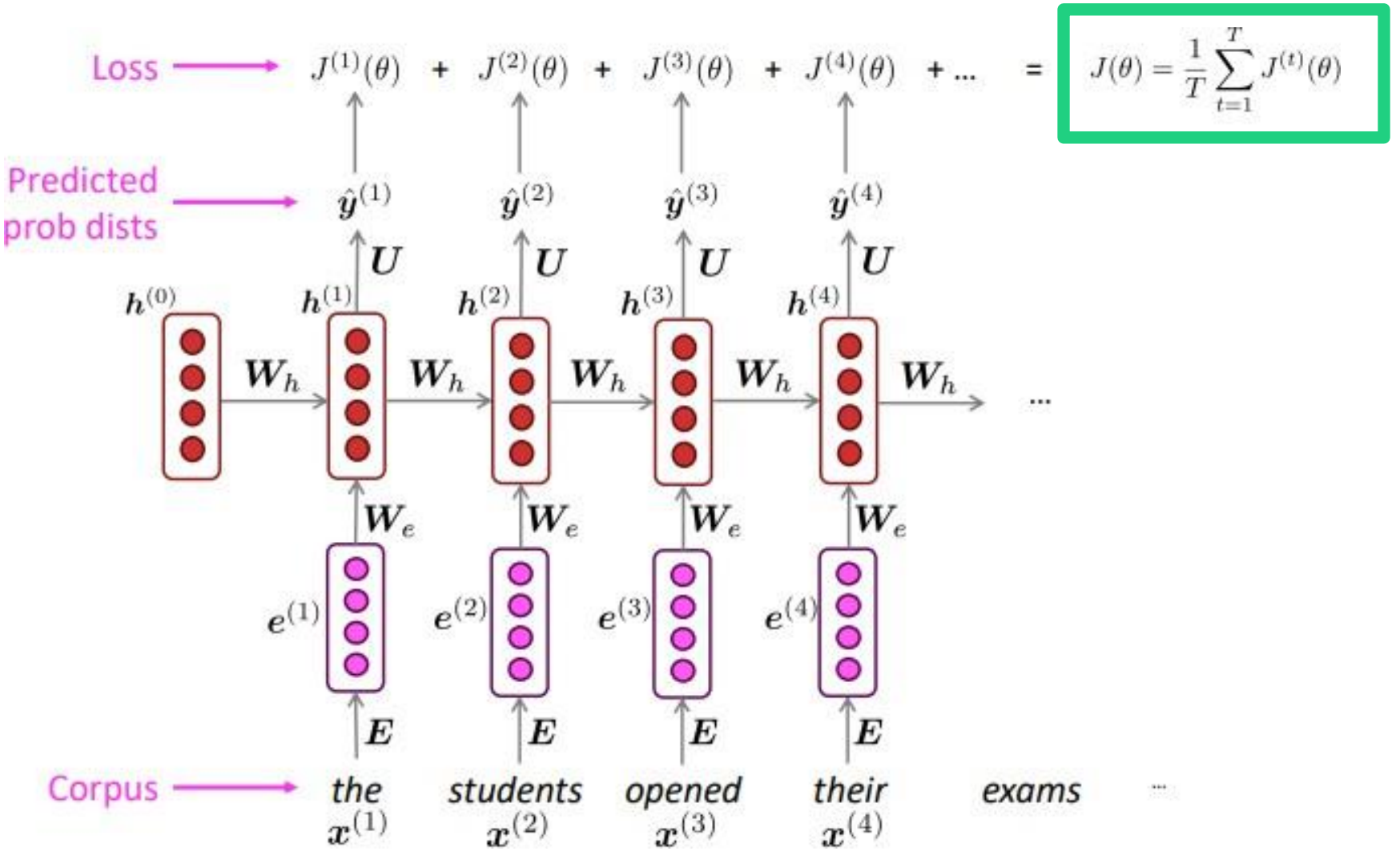
출처 : CS224n (2021)

Training an RNN Language Model



출처 : CS224n (2021)

Training an RNN Language Model



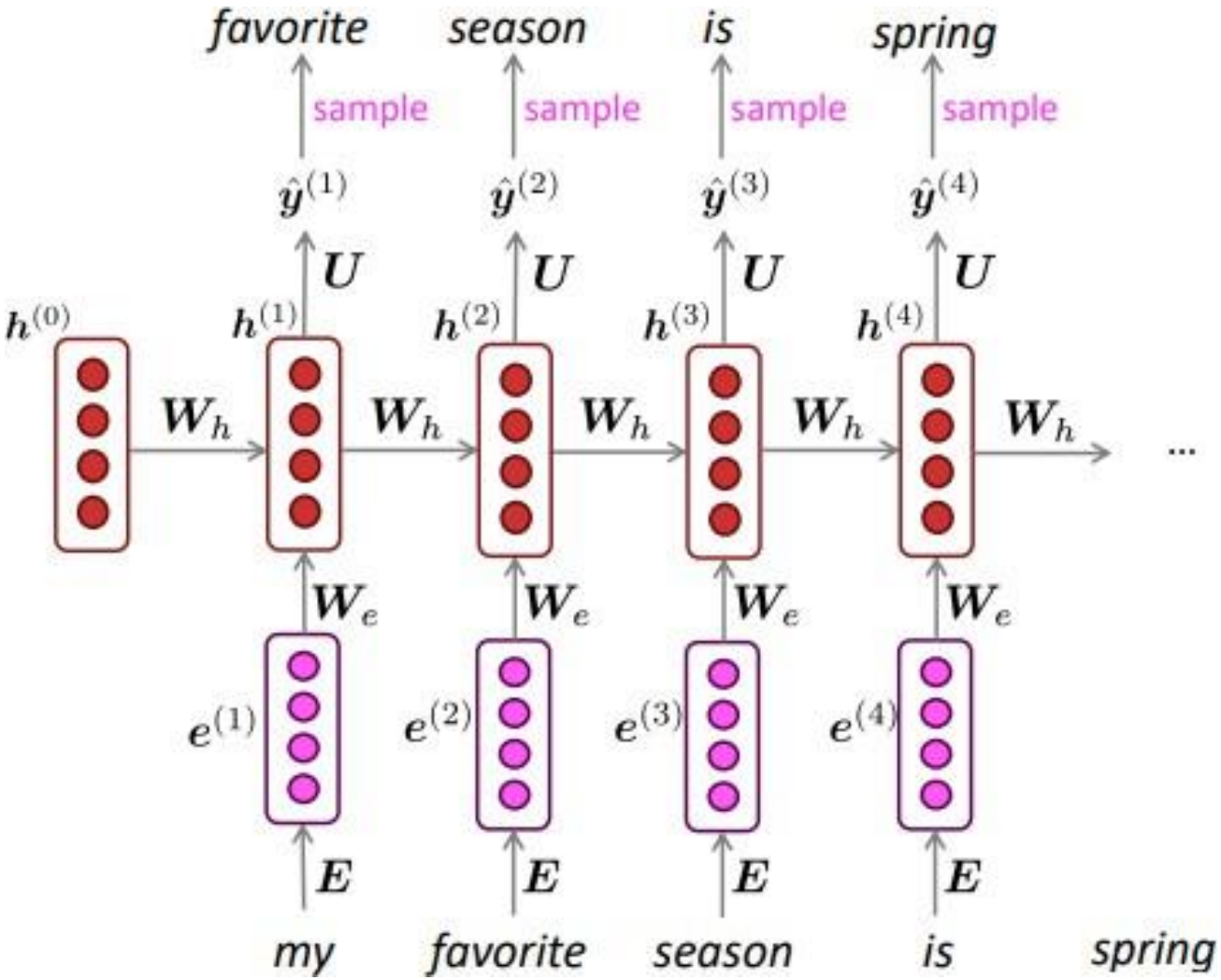
출처 : CS224n (2021)

Teacher Forcing

- 학습 과정에서 t-1개의 input으로 다음 단어를 예측하는데 만약 틀린다면?
  - 오늘 _____
  - 오늘 수업이 1교시인데 늦잠을 자서 수업에 _____

- 모델이 잘못된 문장을 학습하지 않도록 예측한 단어가 아닌 실제 정답을 다음 input으로 넣어준다. 마치 선생님(Teacher)가 정답을 알려주듯이…
  - '오늘' → '오늘 나는' 이라고 예측한다면?
  - 2번째 input으로 '나는'이 아니라 '수업이'를 넣어준다

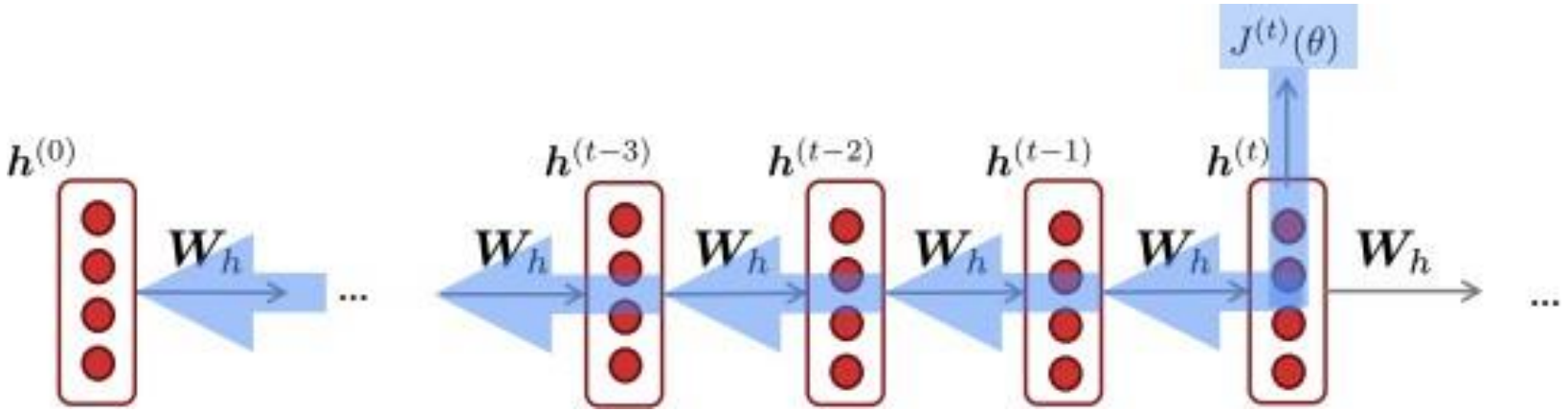
Generating text



출처 : CS224n (2021)

Backpropagation for RNNs

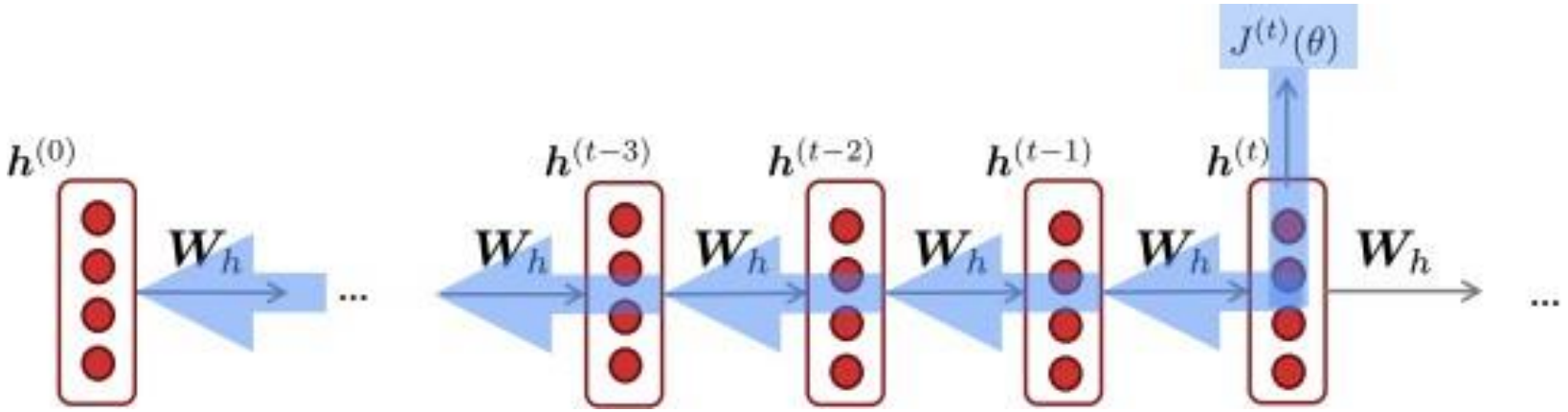
- Backpropagation through time
  - RNN을 학습하는 과정은 다른 딥러닝 모델과 크게 다르지 않다
  - 다만, W_h가 반복돼서 곱해지는 만큼 Loss를 W_h에 대해 미분하고 W_h를 업데이트 하는 방식에 조금 차이가 있다.



출처 : CS224n (2021) Lecture 5

Backpropagation for RNNs

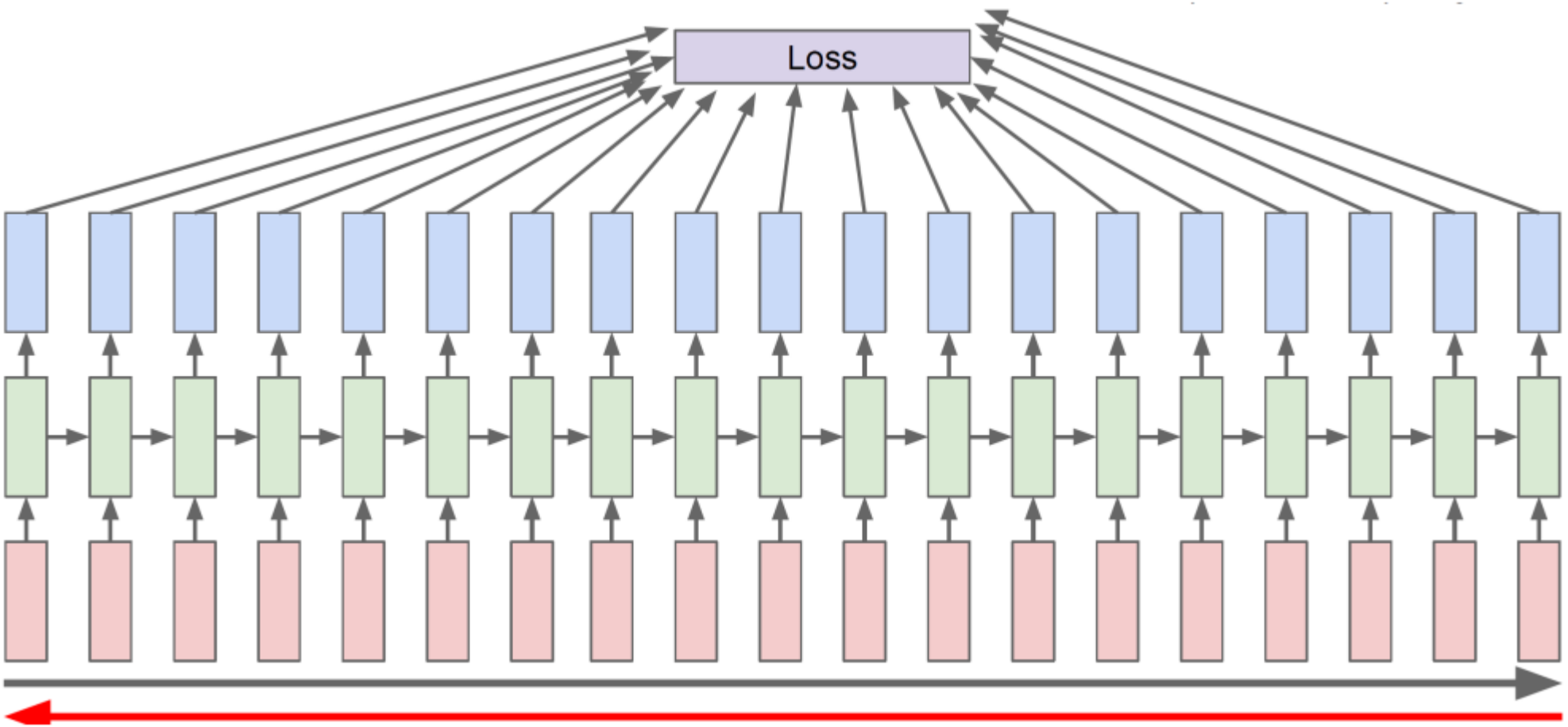
- Backpropagation through time
  - $\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial W_h}\Big|_{(i)}$
  - W_h를 각 step마다 바로 업데이트 하지 않고 나중에 한 번에 업데이트 한다.
  - 순전파를 계산할 때 W_h는 일종의 constant였기 때문이다.
  - 값을 바로 업데이트 해주면 이후의 역전파 계산 과정에서 W_h의 값이 달라지게 된다.



출처 : CS224n (2021)

Backpropagation for RNNs

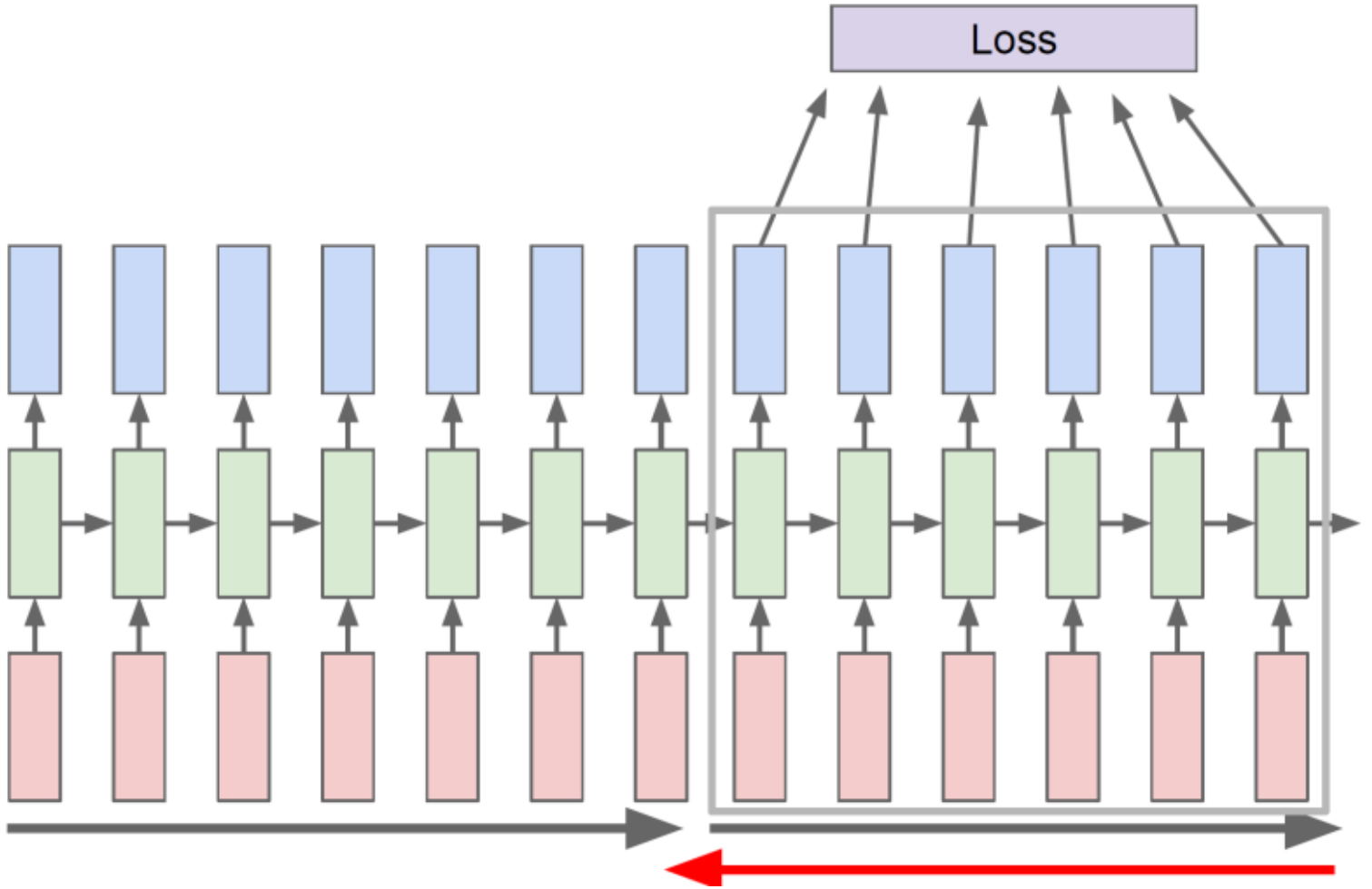
- Backpropagation through time



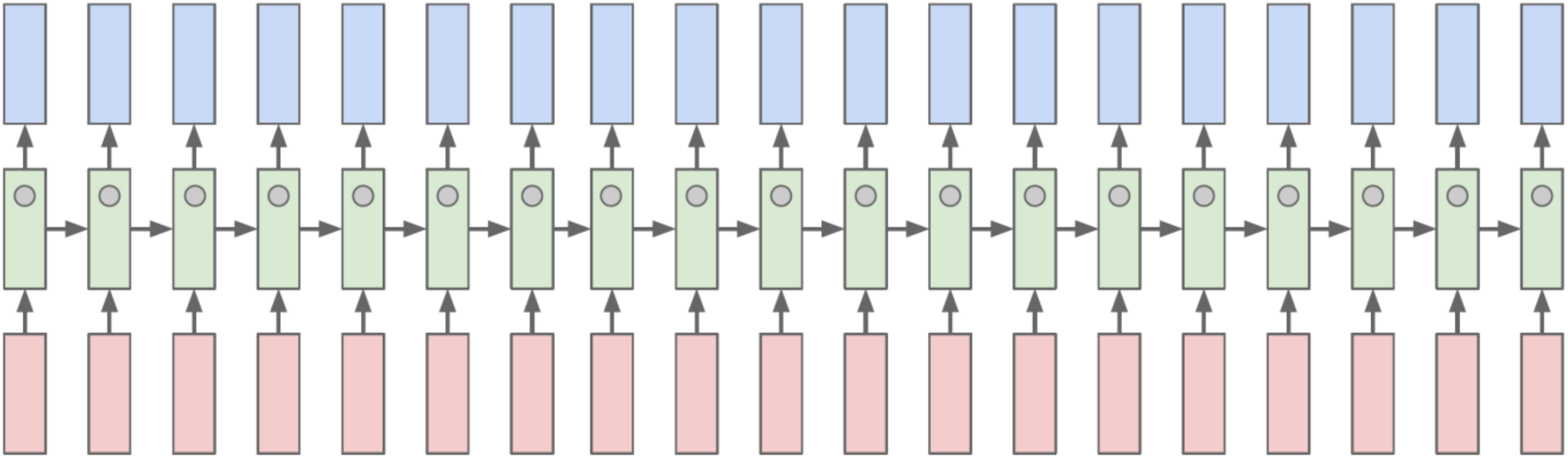
출처 : http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Backpropagation for RNNs

- Backpropagation through time



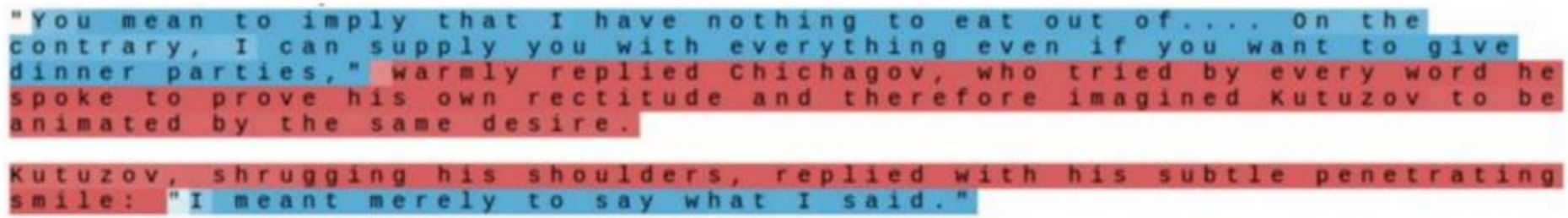
How RNN works



출처 : http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

How RNN works

- Quote detection cell



- If statement cell

출처 : http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Evaluating Language Models

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

- $perplexity = \Pi_{t=1}^{T} \left(\frac{1}{P_{LM}(x^{(t+1)}|x^{(t)},...,x^{(1)})} \right)^{\frac{1}{T}}$
  - 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 헷갈린다는 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) ↓

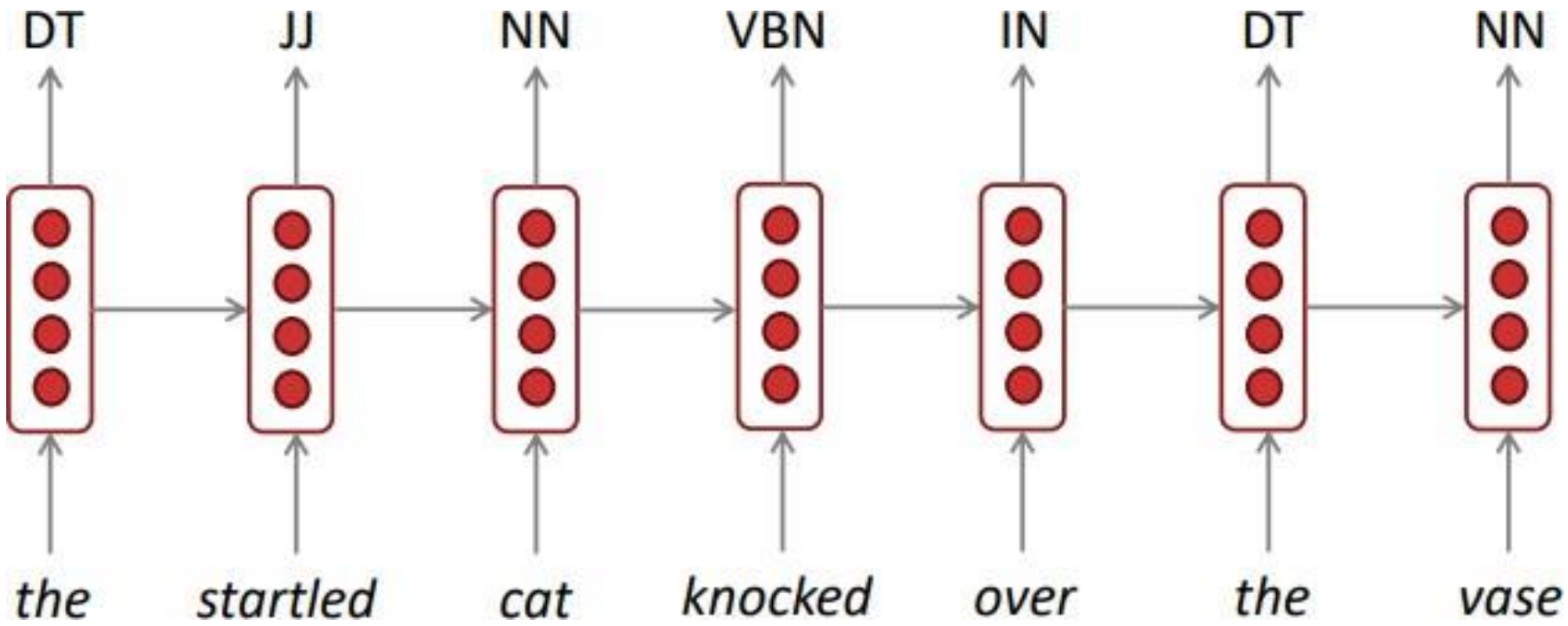
출처 : CS224n (2021) Lecture 5

Various NLP tasks using RNNs

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

Various NLP tasks using RNNs

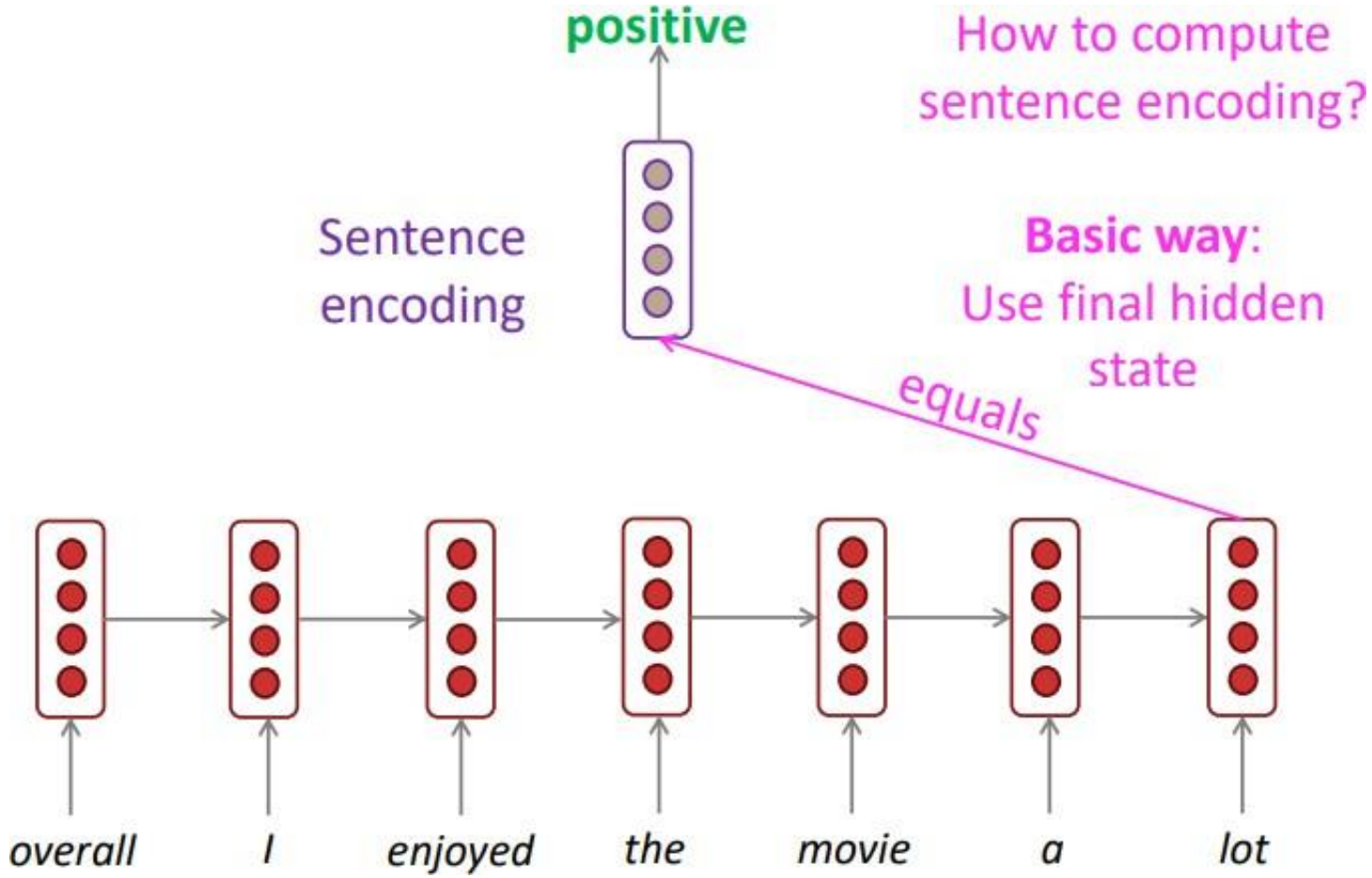
- POS tagging, NER recognition



출처 : CS224n (2021)
Lecture 5

Various NLP tasks using RNNs

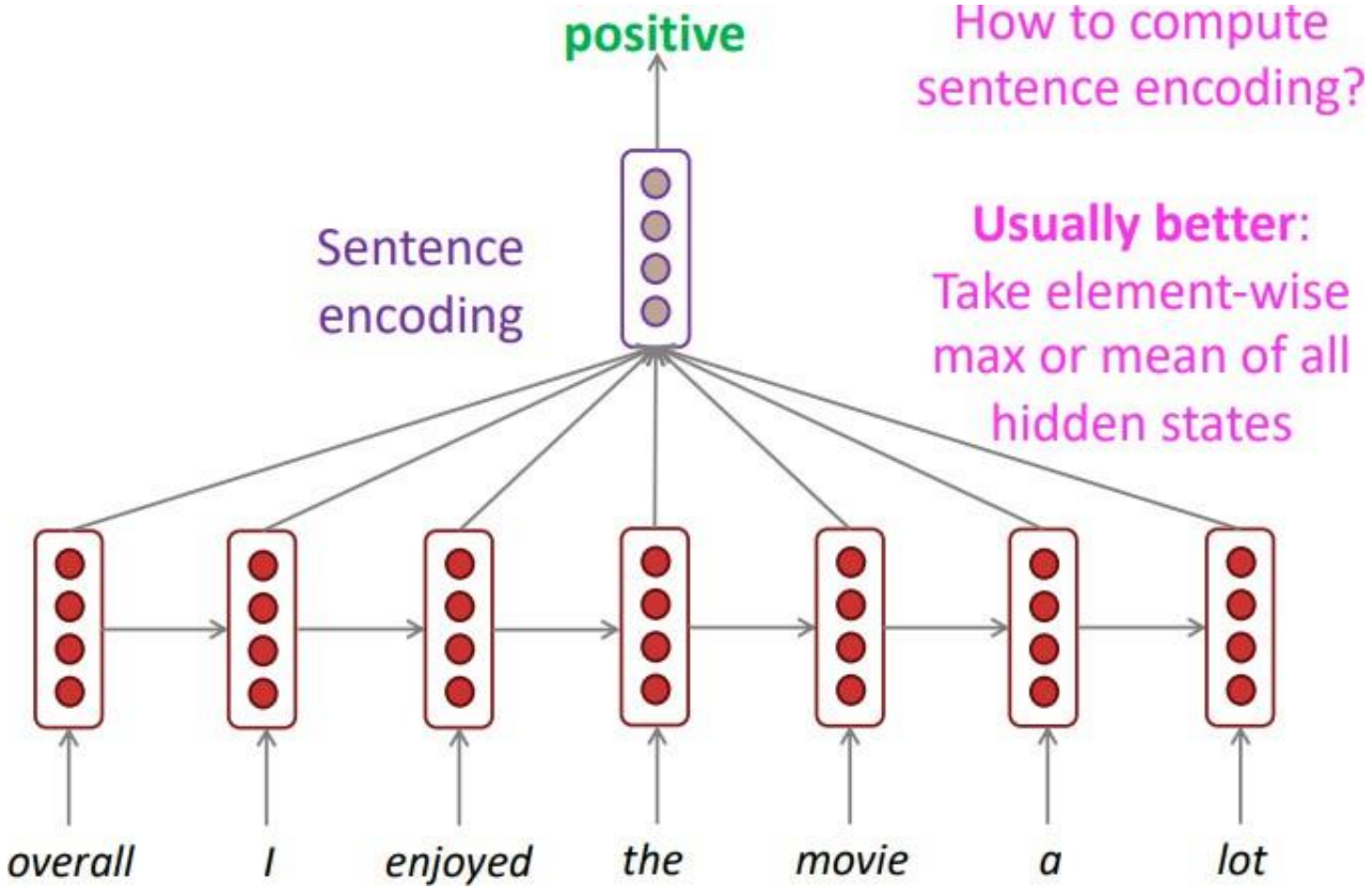
- Sentiment classification



출처 : CS224n (2021) Lecture 5

Various NLP tasks using RNNs

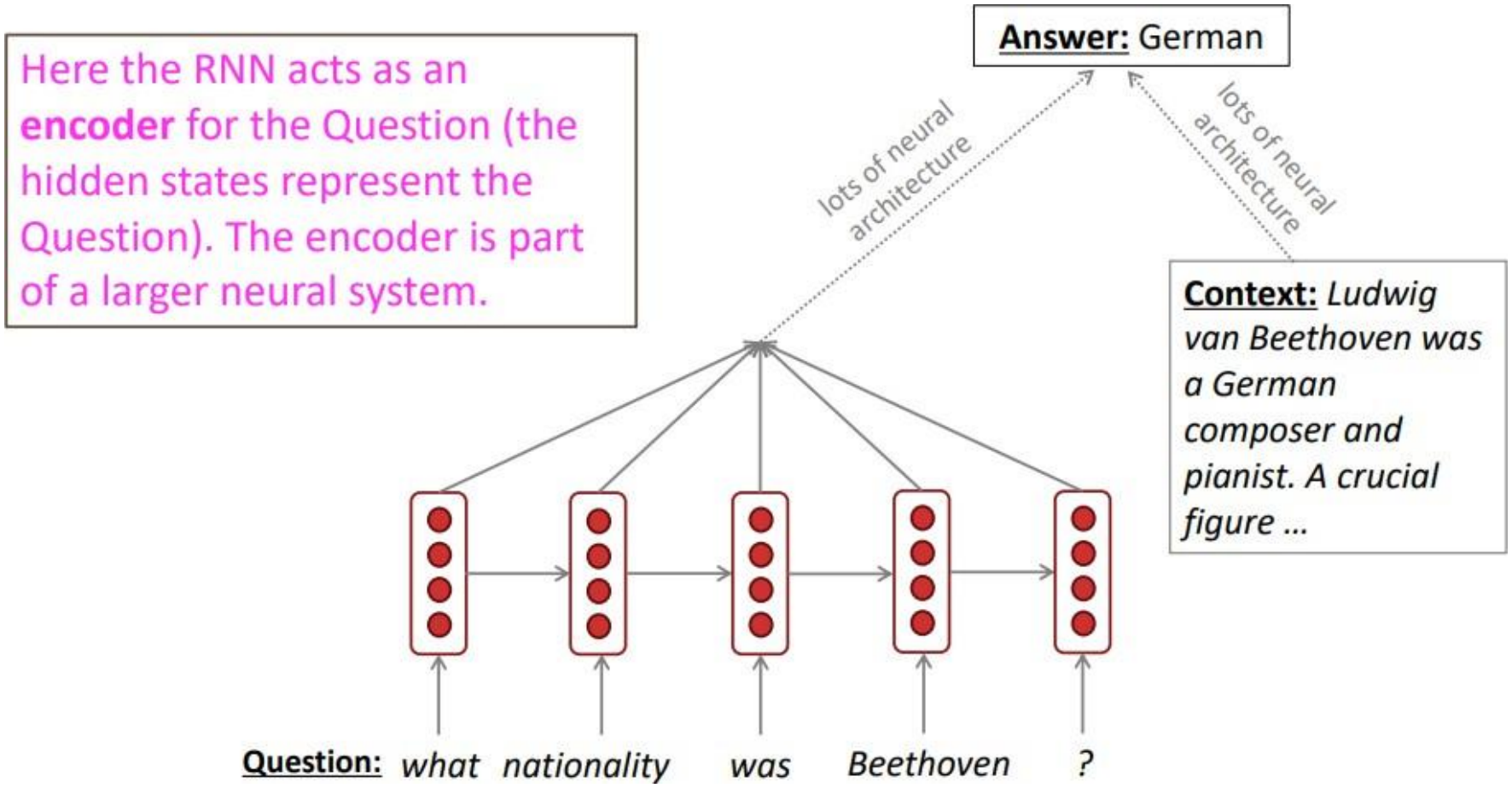
- Sentiment classification



출처 : CS224n (2021) Lecture 5

Various NLP tasks using RNNs

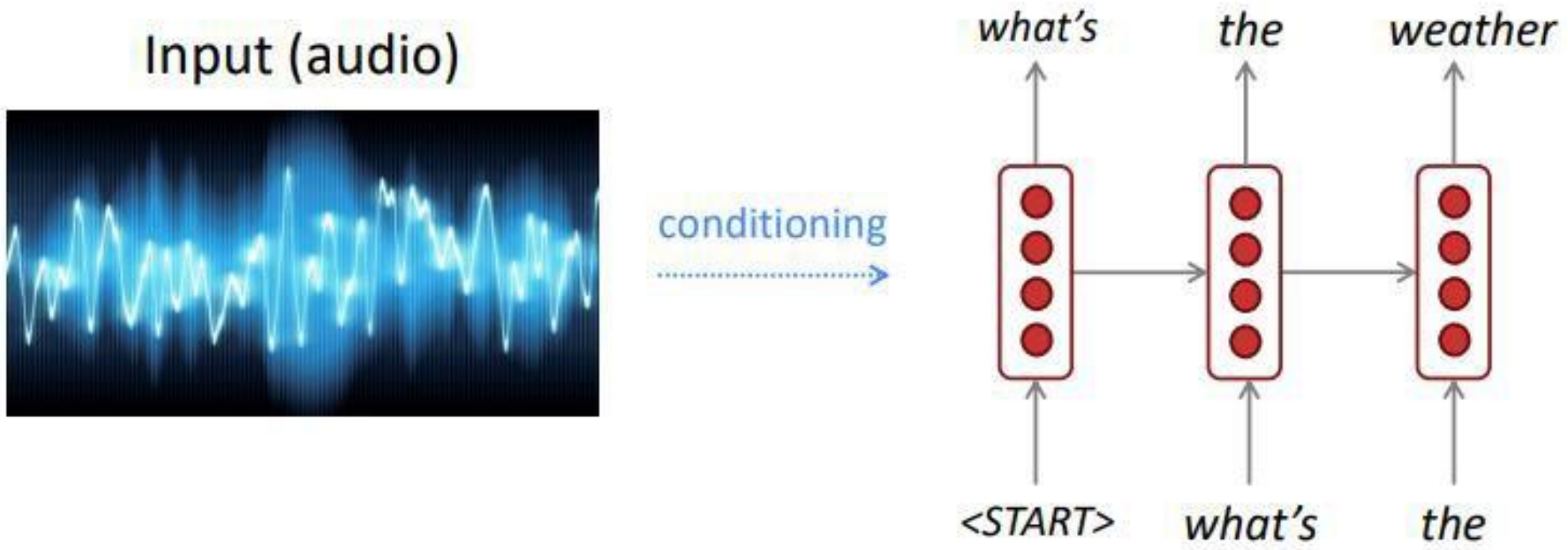
- Question answering, Machine translation



출처 : CS224n (2021) Lecture 5

Various NLP tasks using RNNs

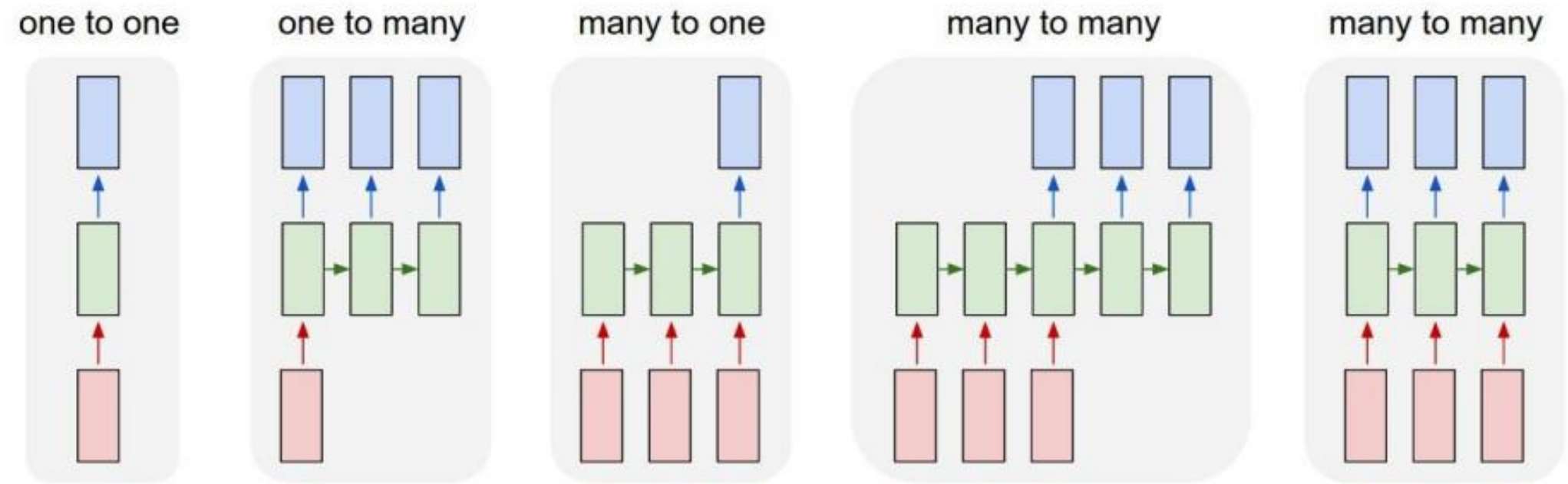
- Speech recognition



출처 : CS224n (2021) Lecture 5

Recurrent Neural Networks (RNN)

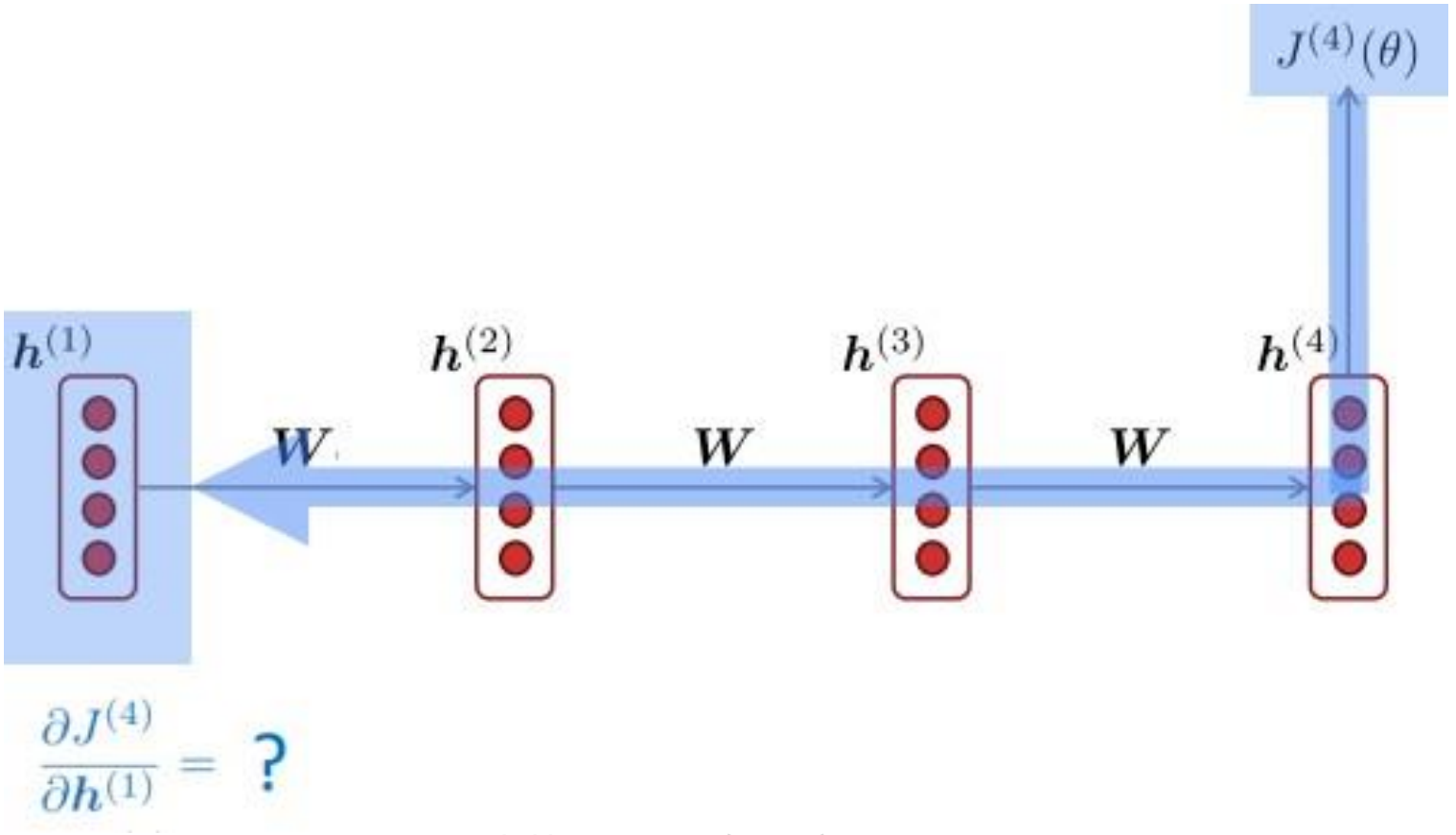
- RNN의 다양한 형태



출처: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Problems of RNNs

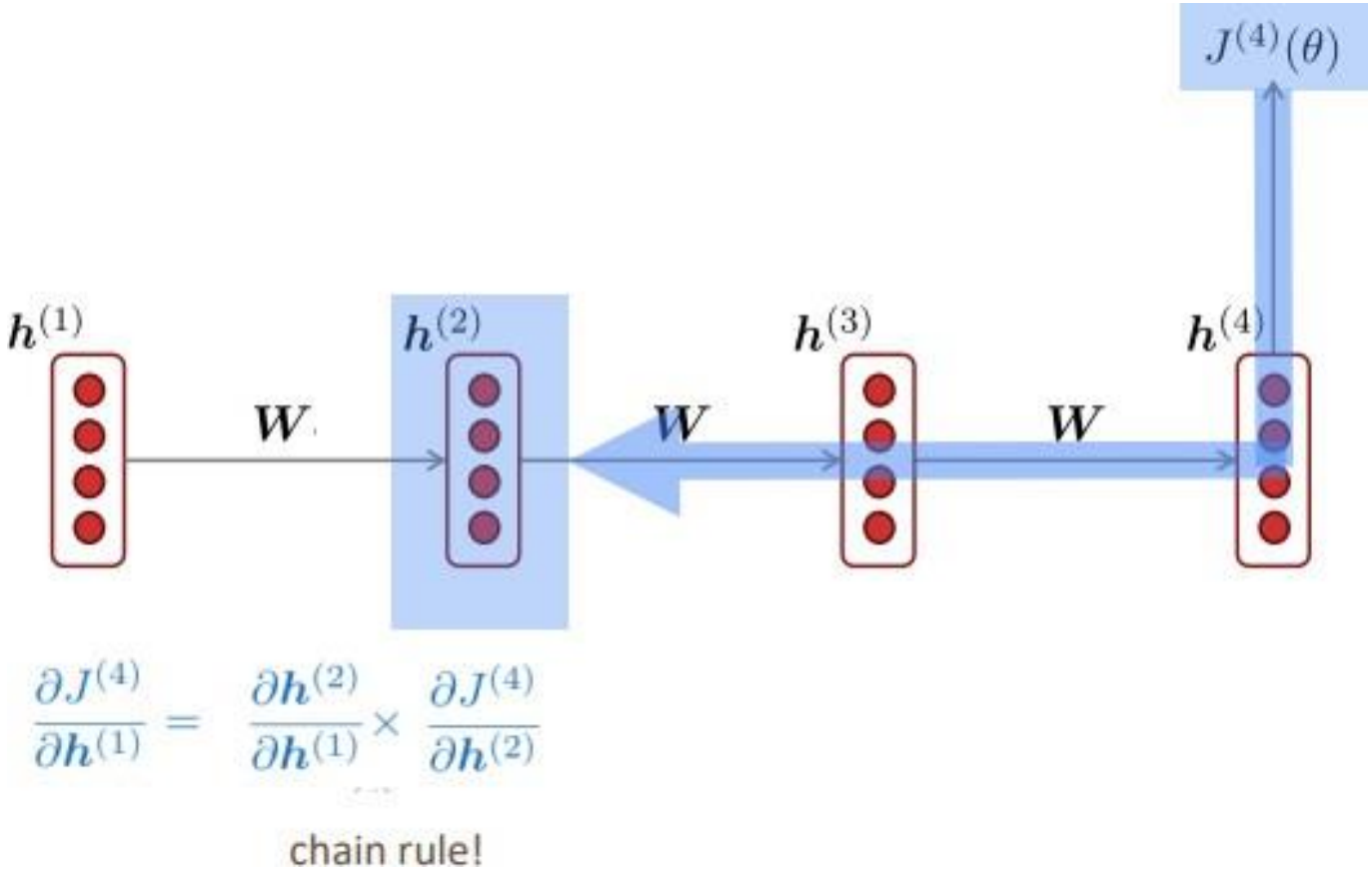
- RNN에서 backpropagation을 계산할 때…



출처 : CS224n (2021)

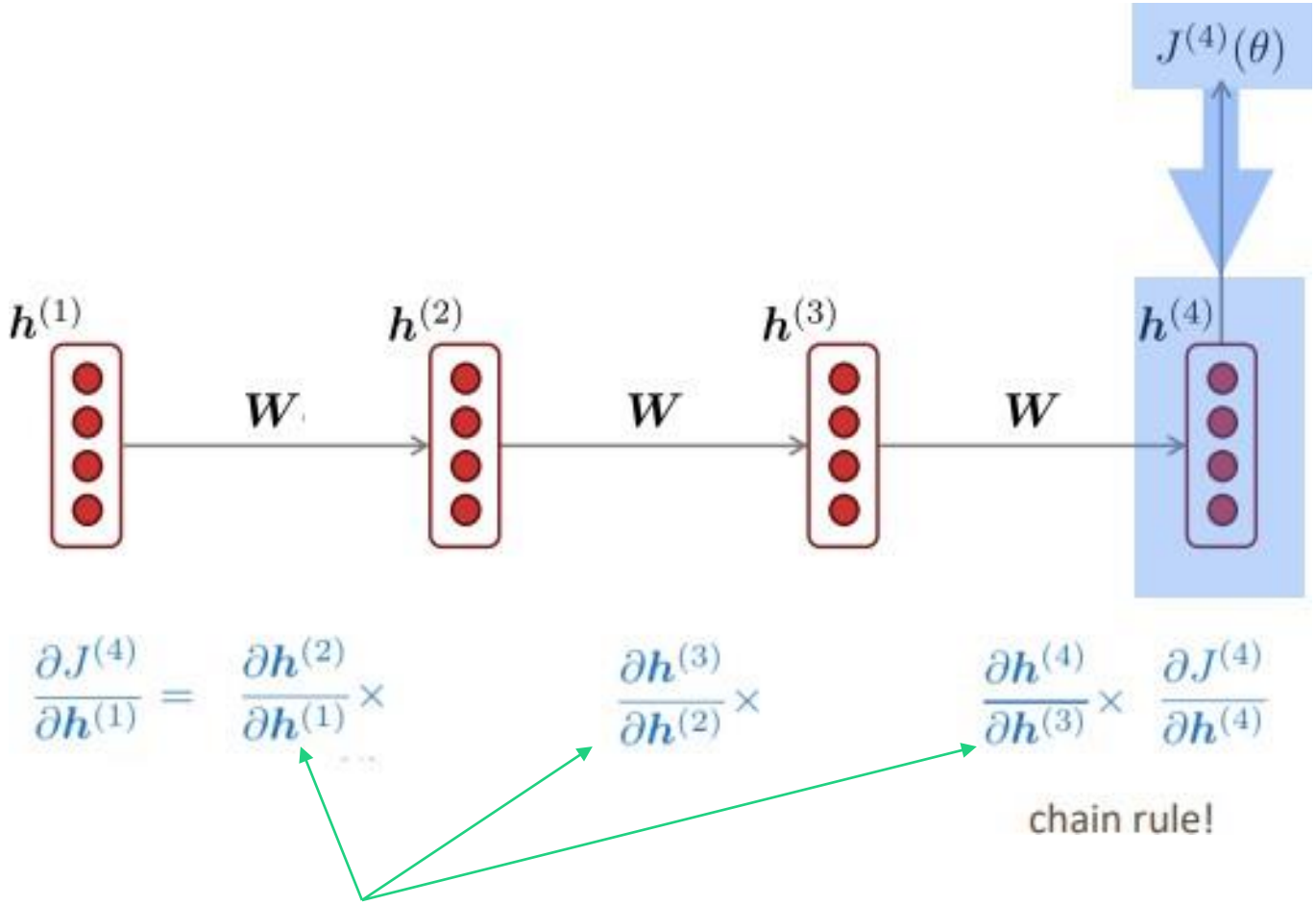
Problems of RNNs

- Chain Rule을 적용해서 구하게 된다



출처 : CS224n (2021) Lecture 6

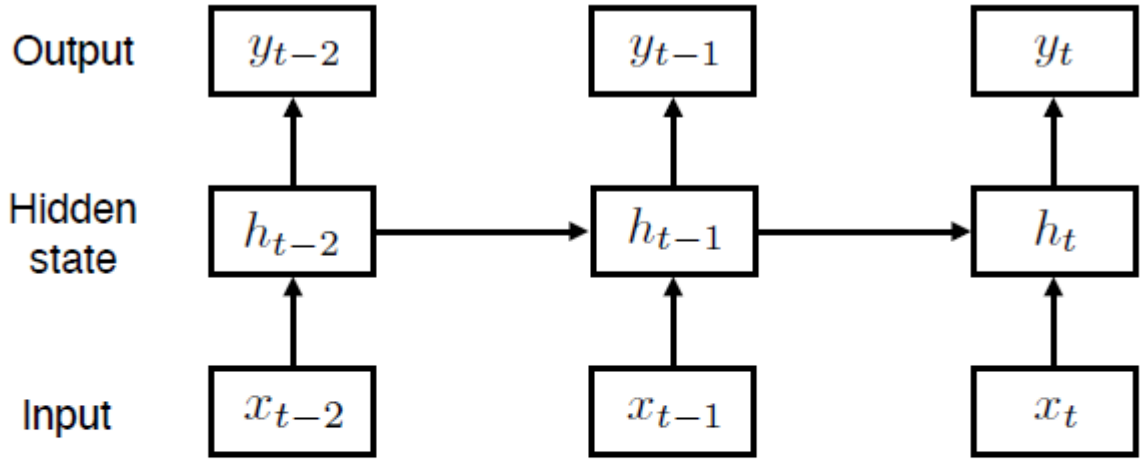
Problems of RNNs



이 값들에 따라 문제가 생길 수 있다.

출처: CS224n (2021) Lecture 6

Problems of RNNs



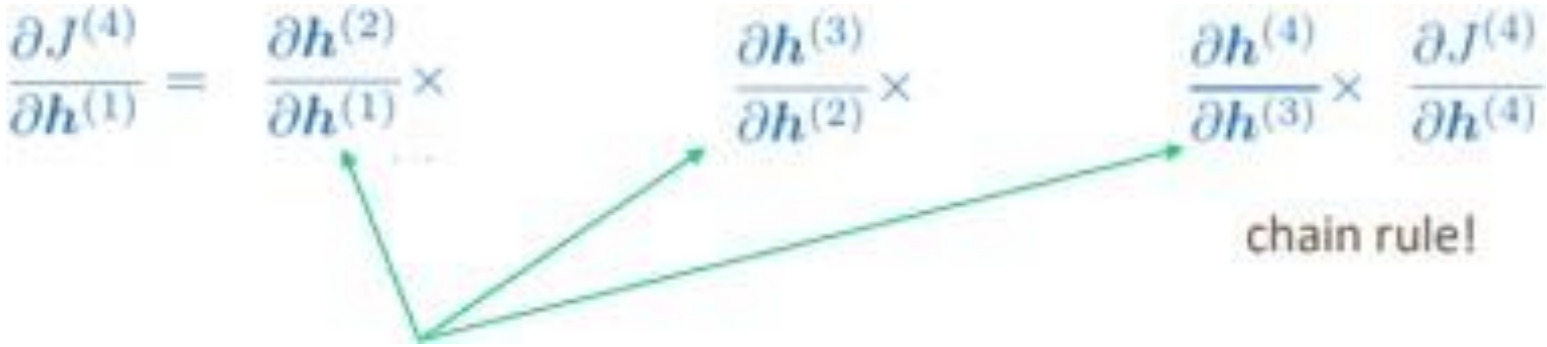
$$h_1 = \phi(W^T h_0 + U^T x_1)$$
$$h_2 = \phi(W^T \phi(W^T h_0 + U^T x_1) + U^T x_2)$$
$$h_3 = \phi(W^T \phi(W^T \phi(W^T h_0 + U^T x_1) + U^T x_2) + U^T x_3)$$
$$h_4 = \phi(W^T \phi(W^T \phi(W^T \phi(W^T h_0 + U^T x_1) + U^T x_2) + U^T x_3) + U^T x_4)$$

Vanishing / exploding gradient

출처 : https://taeuk-kim.tistory.com/26?category=886962

Problems of RNNs ① Exploding Gradient

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

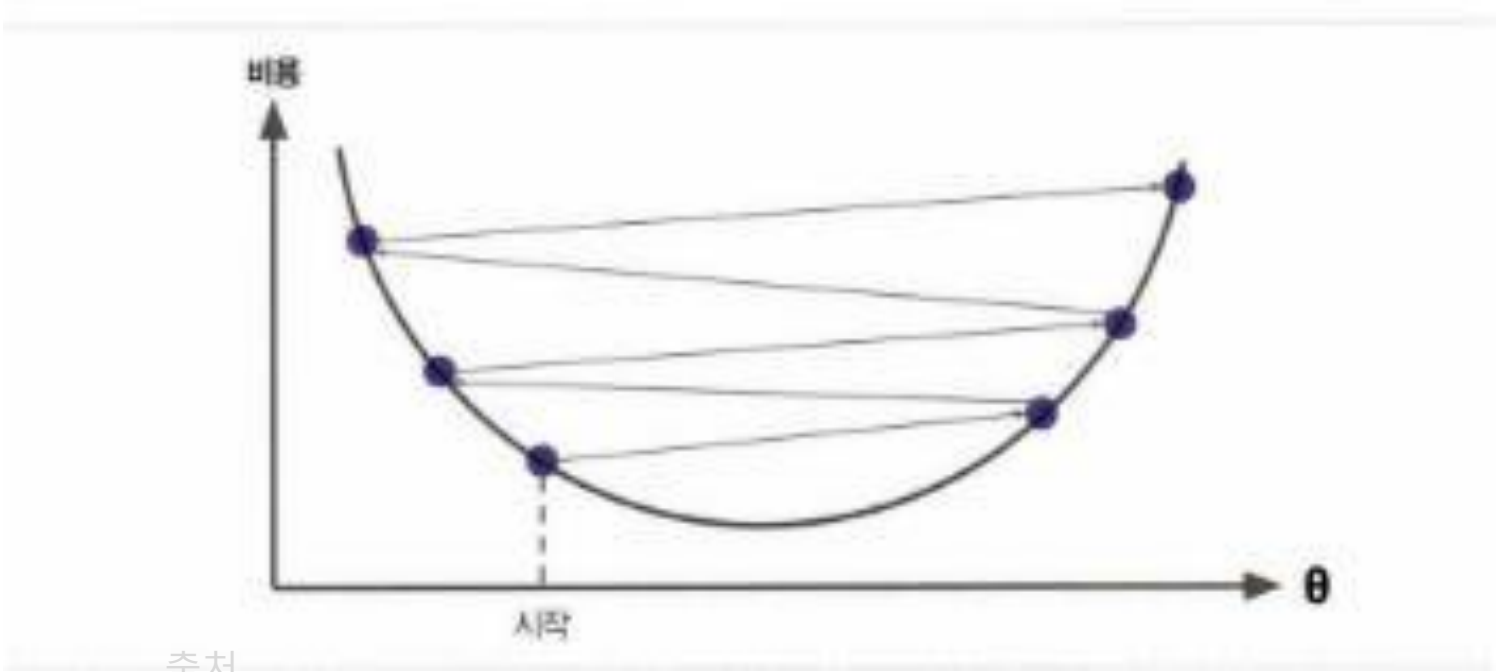


각 값이 너무 커지면 전체 gradient의 값이 커진다

Problems of RNNs ① Exploding Gradient

- SGD update step이 커지면 bad updates일 가능성이 높다

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



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

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 유지하되 그 크기를 줄인다.

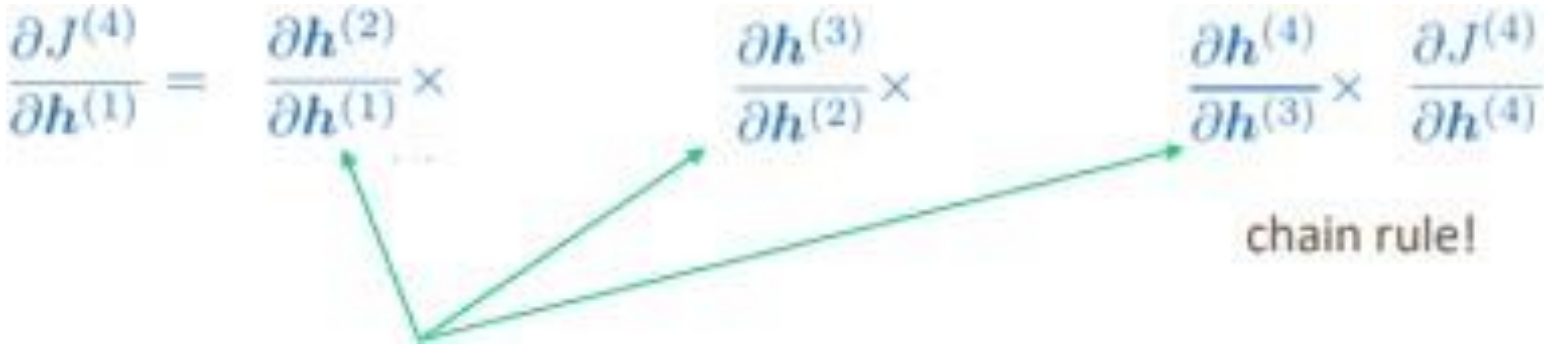
Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
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

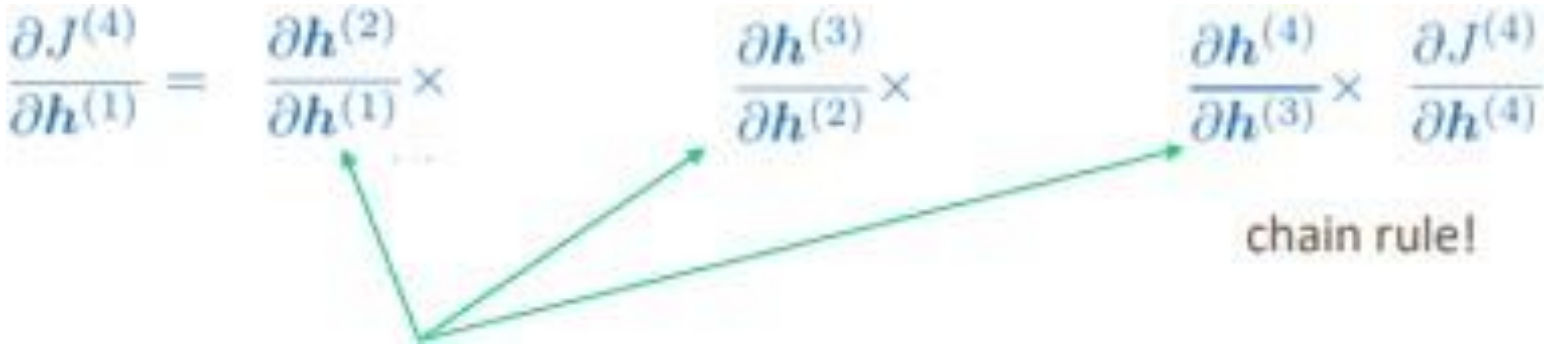
- 각각의 값이 너무 작으면 gradient의 값이 0에 가까워지고 SGD update step가 일어나지 않는다.



각 값이 너무 작으면 전체 gradient의 값은 0에 가까워진다

Problems of RNNs ② Vanishing Gradient

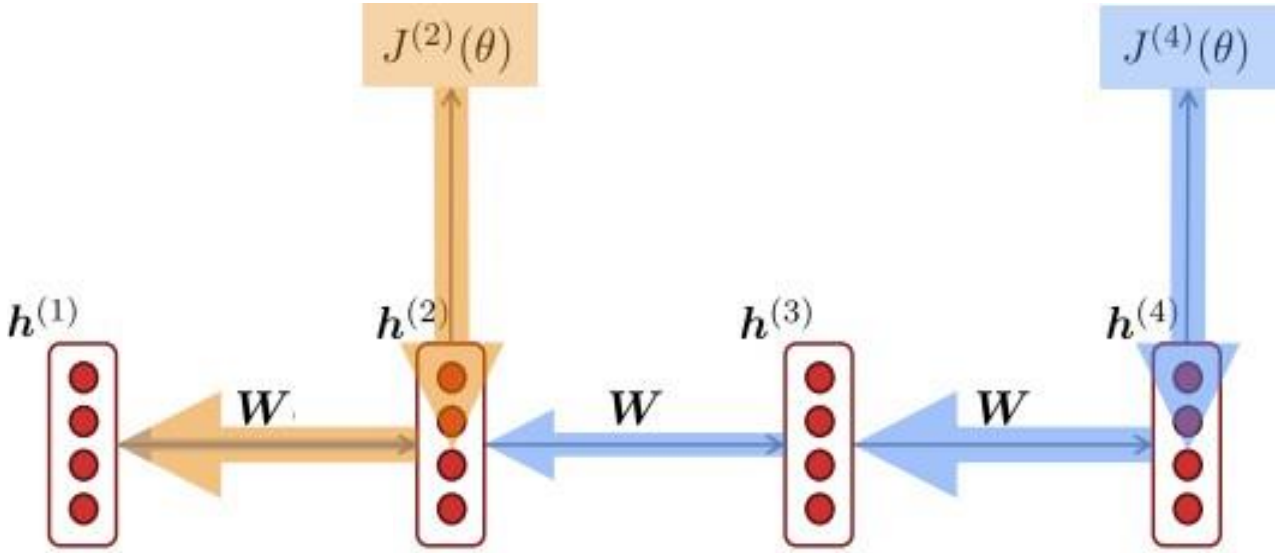
- 각각의 값이 너무 작으면 gradient의 값이 0에 가까워지고 SGD update step가 일어나지 않는다.



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.



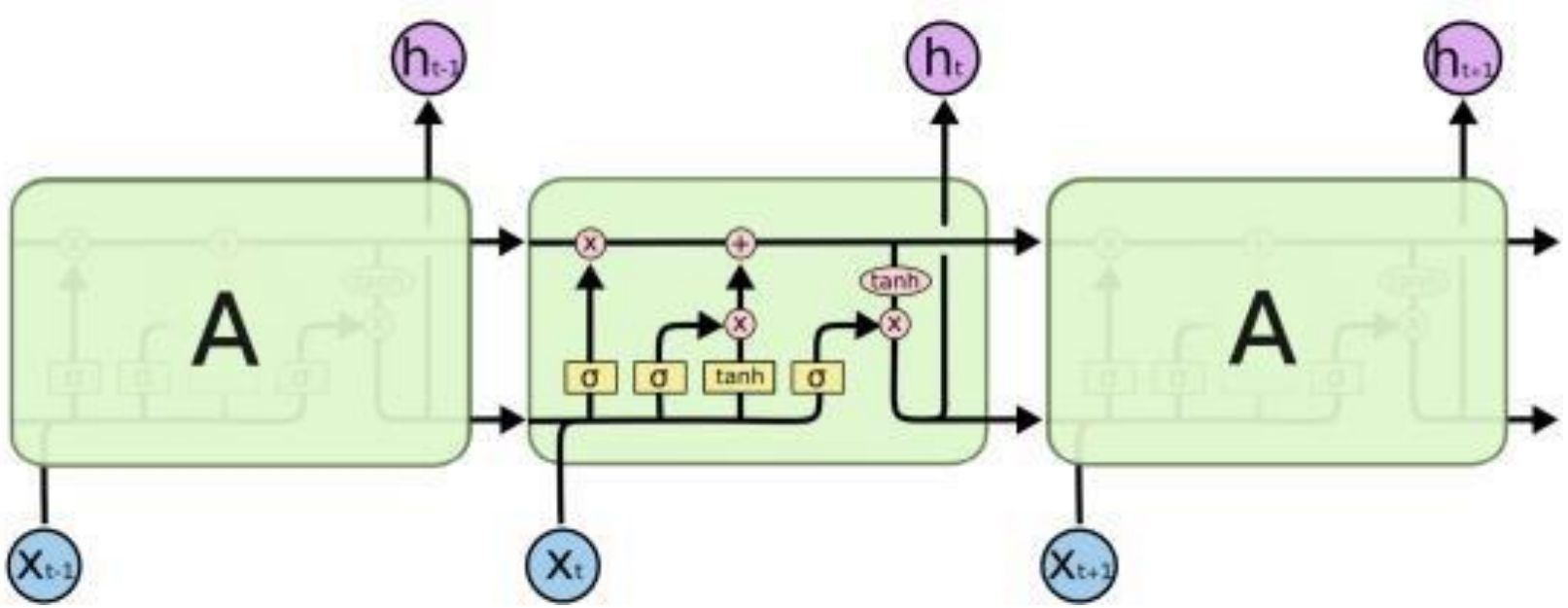
출처 : CS224n (2021) Lecture 6

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 state를 추가해 메모리의 역할을 하게 하고, hidden state는 더 이상 이전 정보는 저장하지 않도록 해보자.

Long Short-Term Memory RNNs (LSTMs)

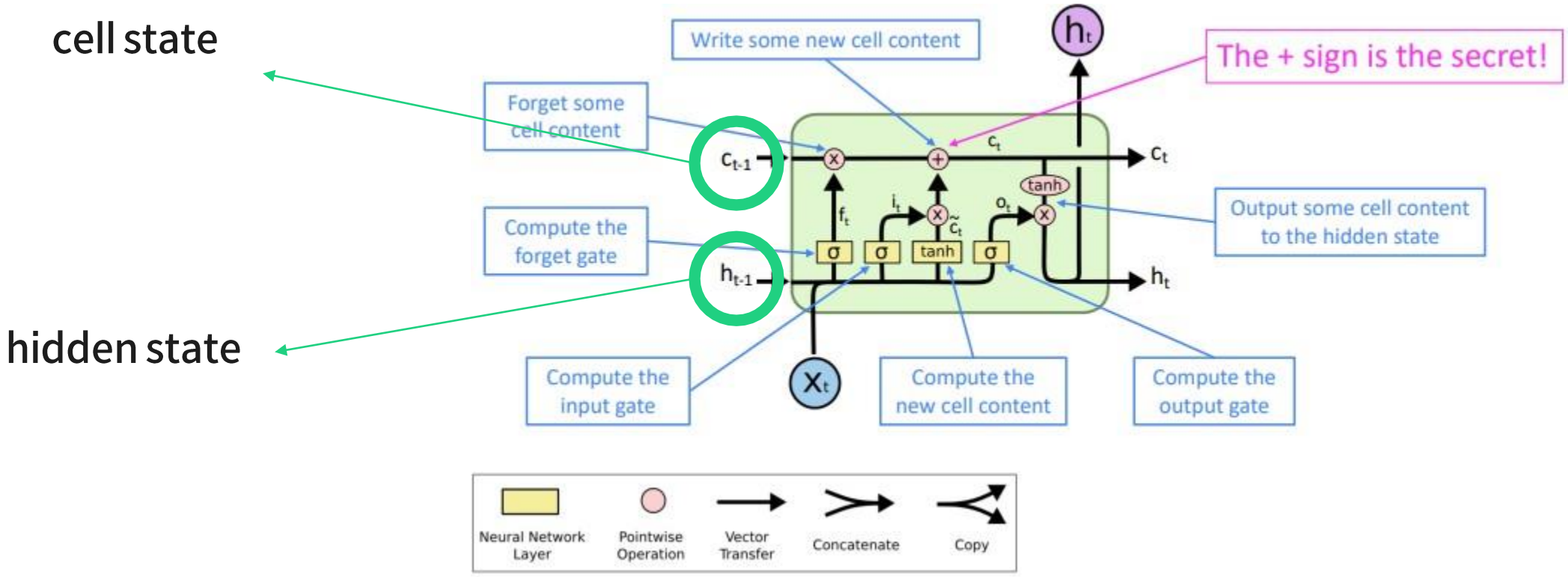
- LSTM의 구조



출처 : CS224n (2021)
Lecture 6

Long Short-Term Memory RNNs (LSTMs)

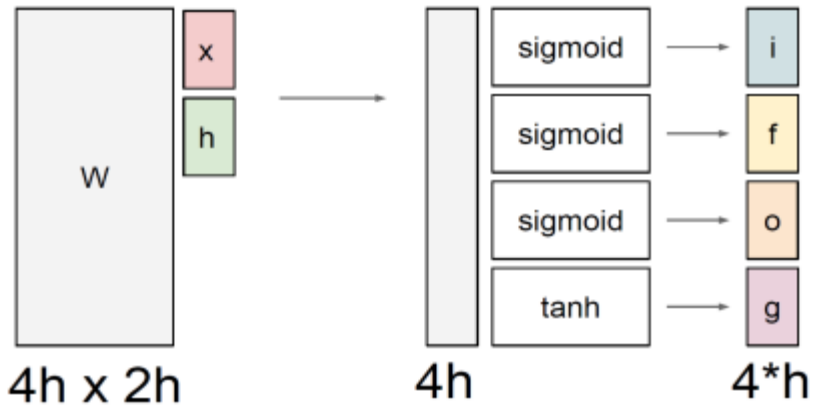
- LSTM에서 각 step에서의 computation
  - 가장 중요한 것은 RNN의 hidden state에서 이전 정보만 따로 기억하는 cell state가 존재



출처 : CS224n (2021) Lecture 6

Long Short-Term Memory RNNs (LSTMs)

- LSTM에서 각 step에서의 computation
  - Input gate (I)
  - Forget gate (F)
  - Output gate (O)



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

출처 : Long short-term memory, Neural computation'97

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:

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

New cell content: this is the new content to be written to the cell

Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content

Hidden state: read ("output") some content from the cell

Sigmoid function: all gate values are between 0 and 1

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}$$

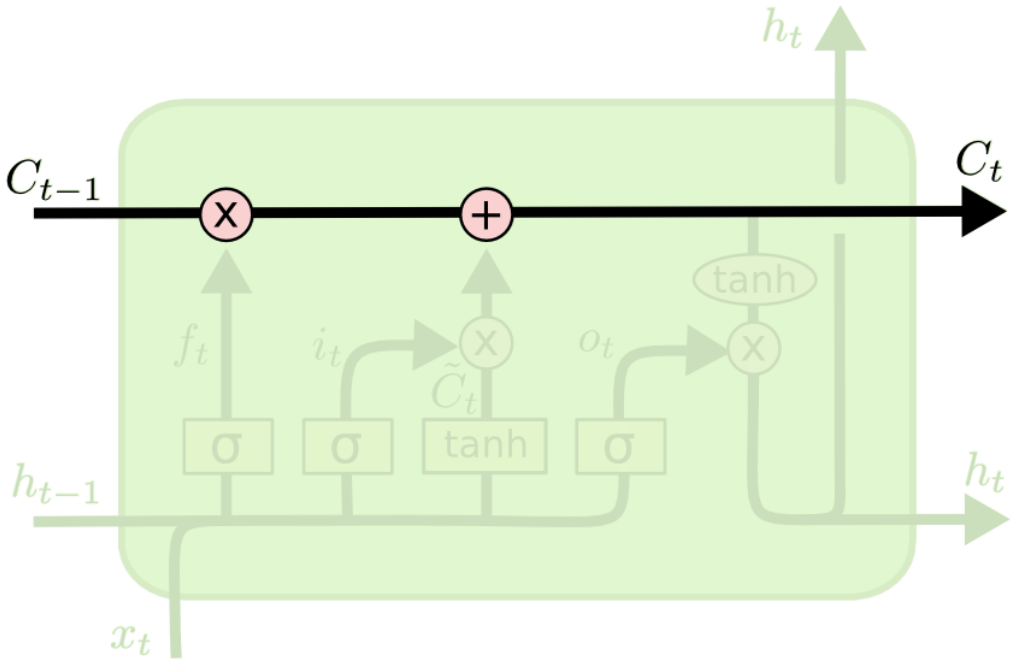
All these are vectors of same length n

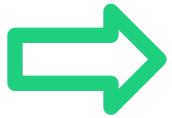
Gates are applied using element-wise (or Hadamard) product: $\odot$

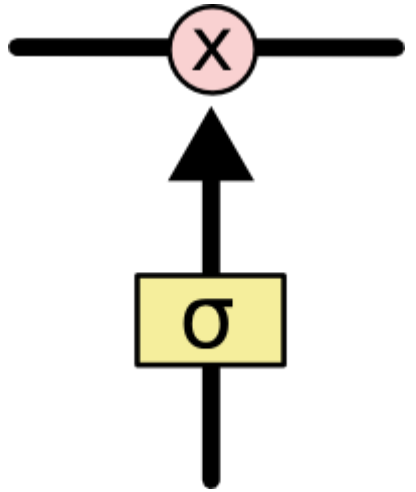
출처 : CS224n (2021)

Long Short-Term Memory RNNs (LSTMs)

- Cell state
  - 이 cell state를 적절하게 변환하기 위해 각 gate들이 사용된다.



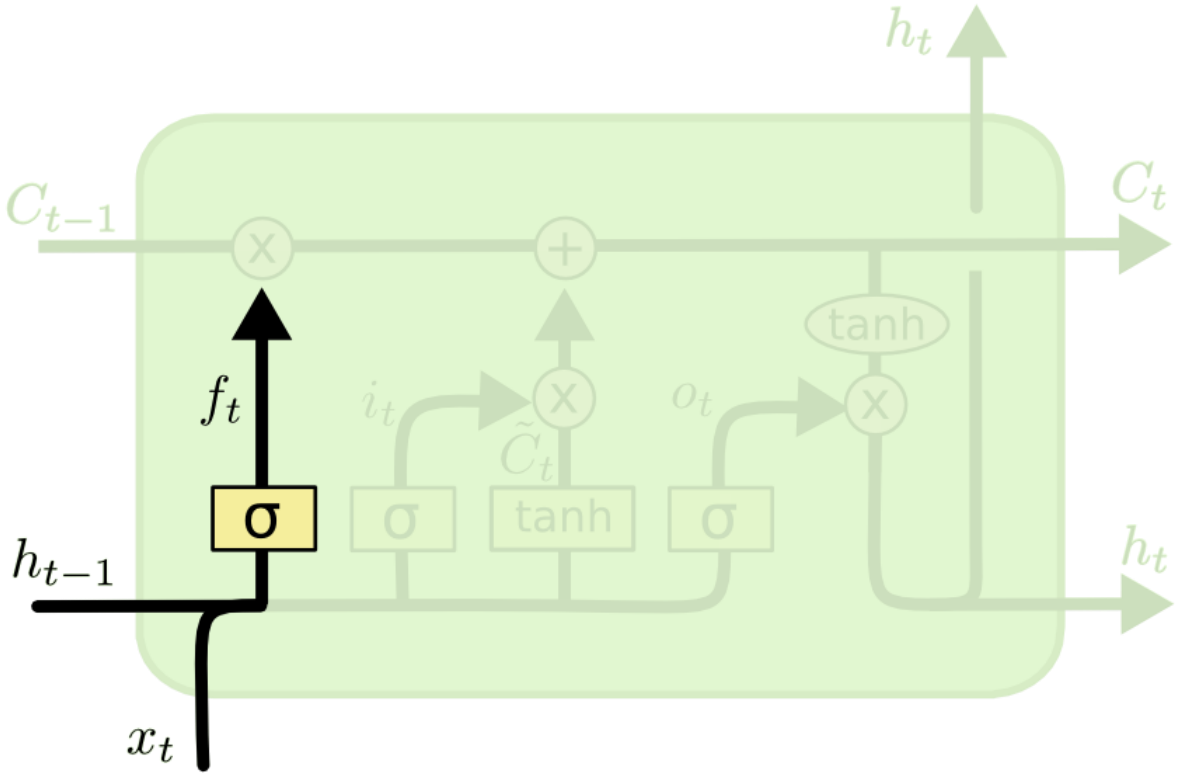




출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory RNNs (LSTMs)

- Forget gate (F)

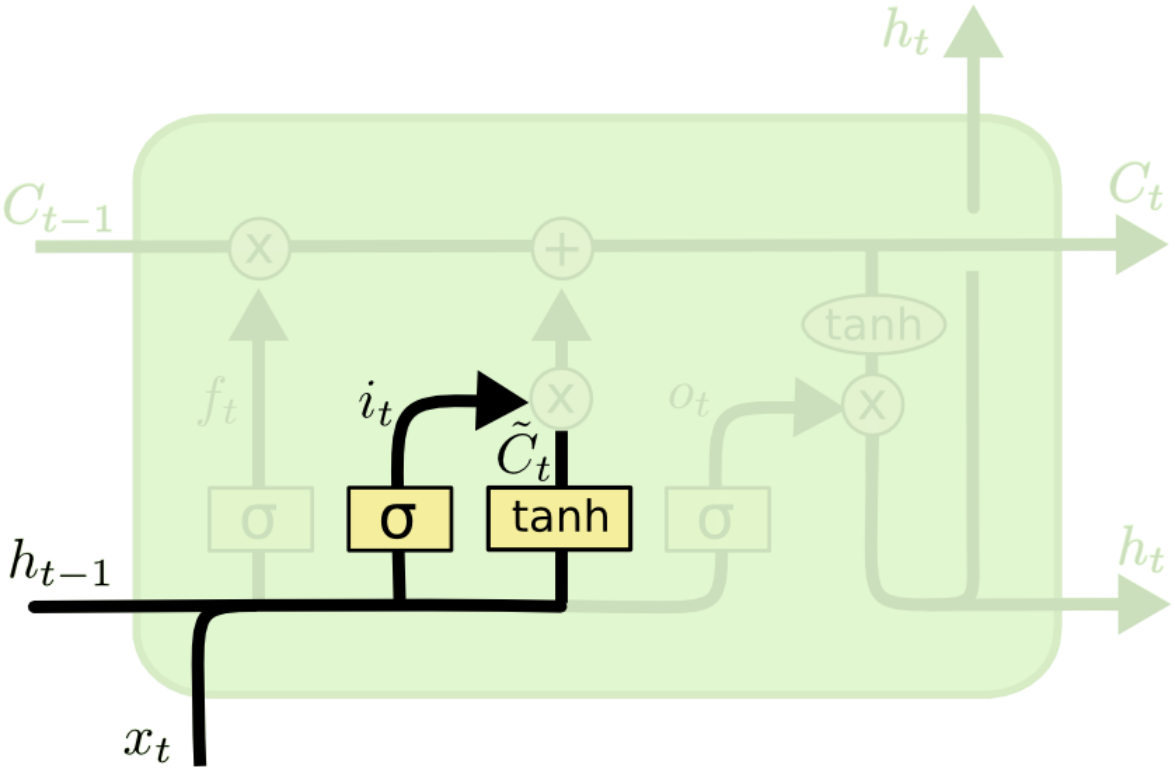


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory RNNs (LSTMs)

- Input gate (I)



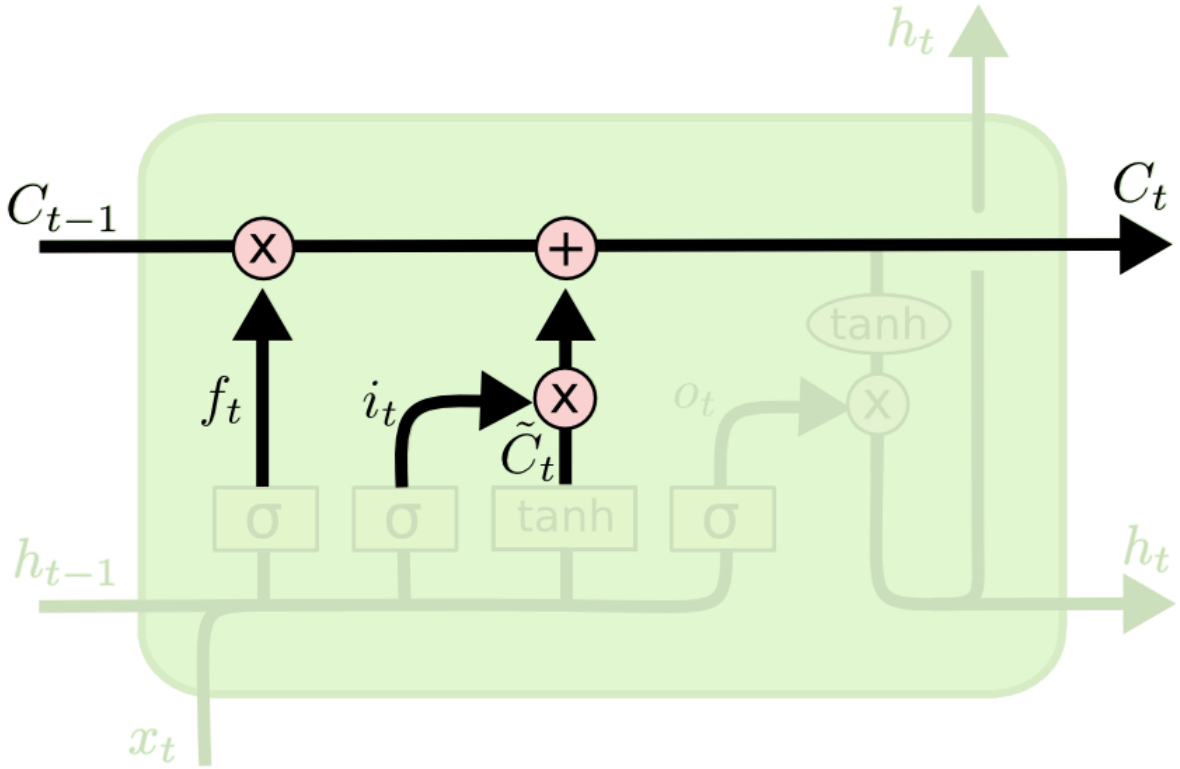
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory RNNs (LSTMs)

- Update cell

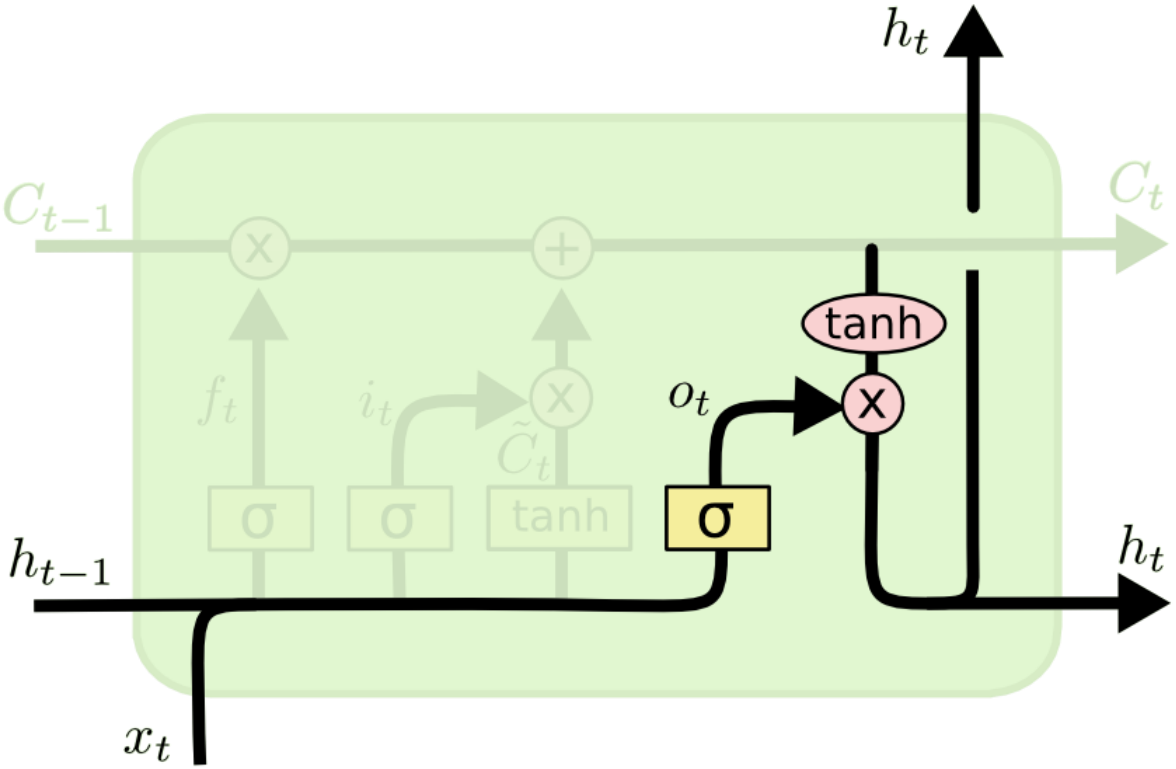


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory RNNs (LSTMs)

- Output gate (O)



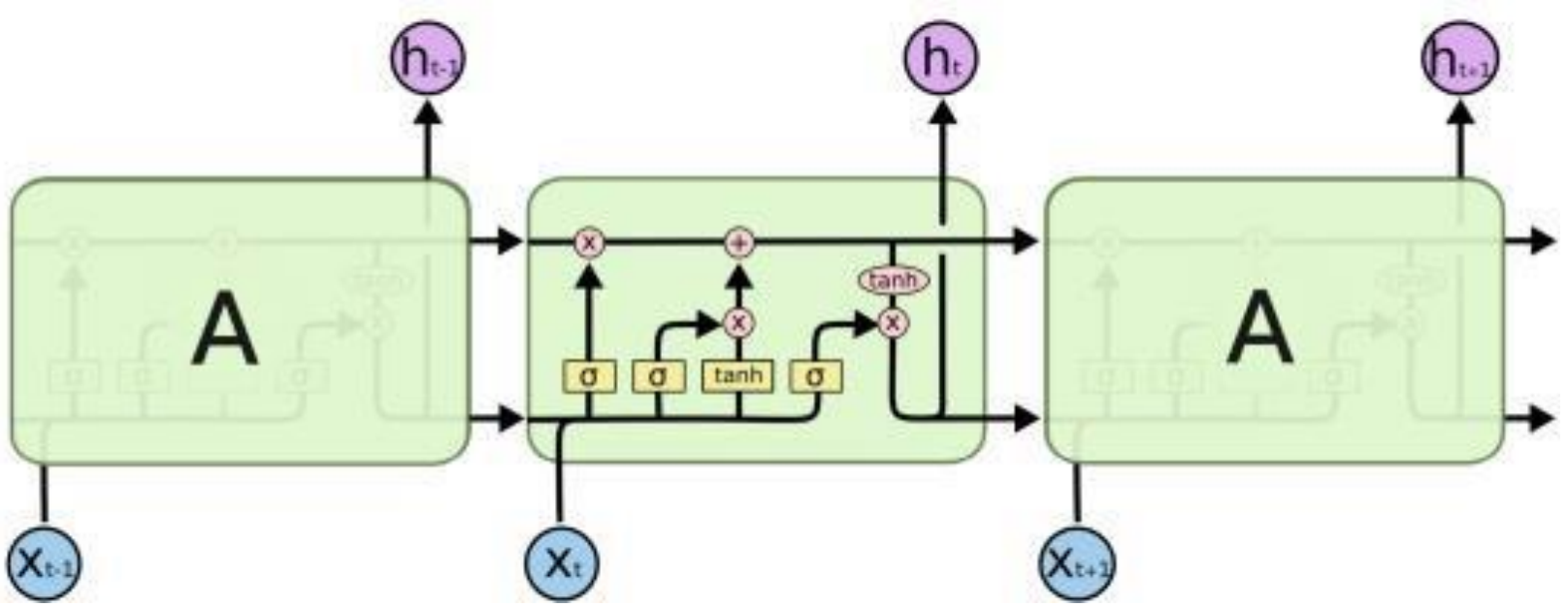
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

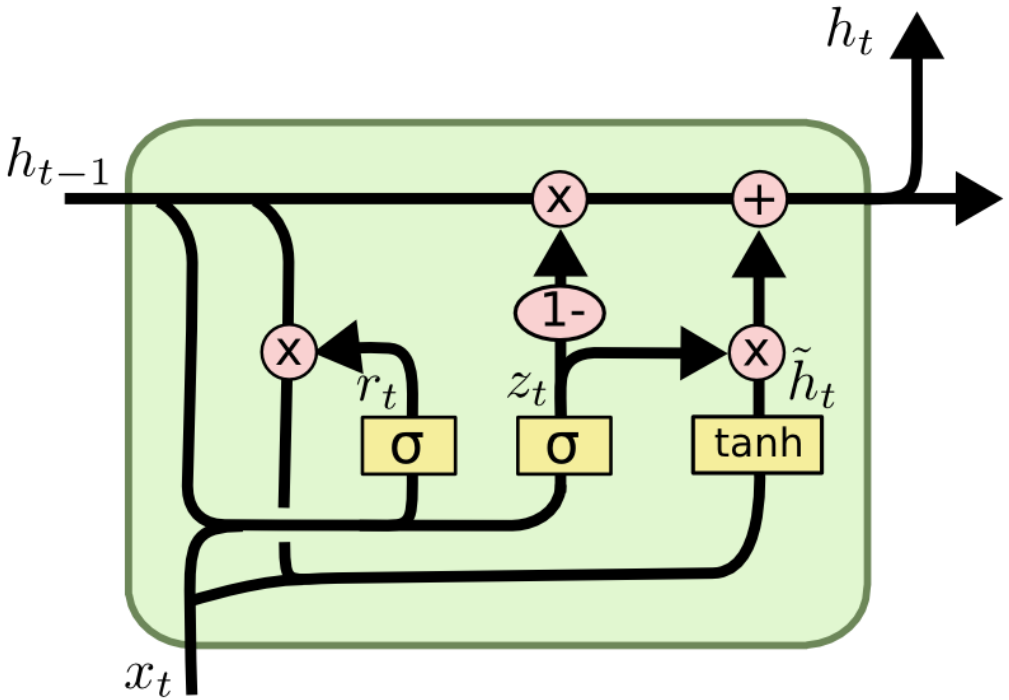
More about LSTM

- https://wikidocs.net/22888



출처 : CS224n (2021)
Lecture 6

GRU..?



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

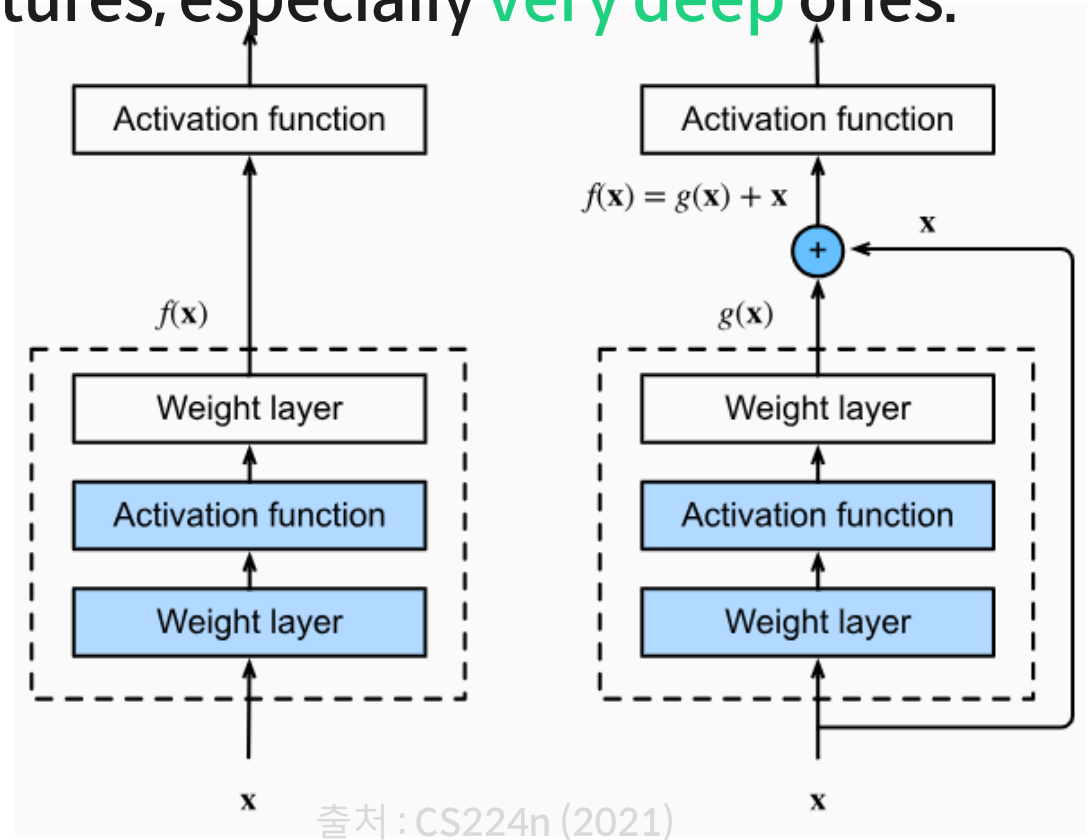
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

ResNet

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



출처 : CS224n (2021)

Discussion questions

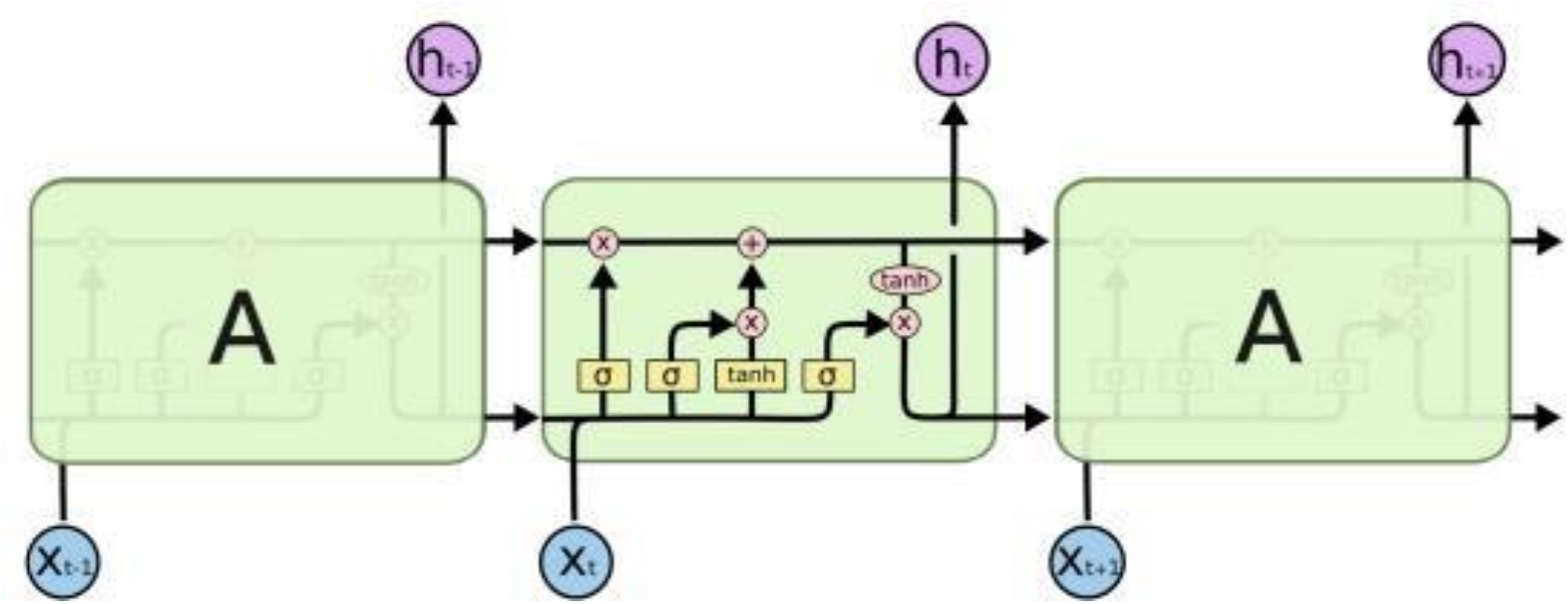
- RNN은 아래와 같은 문제에 잘못된 답을 낸다. 그 이유를 Vanishing Gradient Problem의 관점에서 각각 설명해보자.

  - 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

- LSTM의 구조를 hidden state와 cell state의 변화를 중심으로 이해해봅시다!
- 각 gate의 역할도 함께 생각해보자.



출처 : CS224n (2021)

감사합니다.