CS224N 2023 | WEEK 2-1 박병민

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

"학생들은 그들의 _____를 열었다."

- 책 (책을 열었다)
- 노트북 (노트북을 열었다)
- 시험 (시험지를 열었다)
- 마음 (마음을 열었다)

More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|V|}\}$

Language Model

- 문장에 빈칸을 채울려면? 이걸 기계가 인식하고 예측할려면?
- 주어진 word들이 주어질 때 다음 word를 예측해야함!

주어진 단어 시퀀스

- $x^{\{(1)\}}, x^{\{(2)\}}, \ldots, x^{\{(t)\}}$는 " 학생들은 그들의 " 에 해당

다음에 나올 단어 $\boldsymbol{x}^{\{(t+1)\}}$

- 책, 노트북, 시험지, 마음과 같은 단어 중 하나!

우리가 구하고자 하는 확률 분포?

- $P(x^{t+1} \mid x^{(t)}, \ldots, x^{(1)})$로, 앞선 문맥에 따라 각 단어가 나올 확률을 계산

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$
$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} | \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

This is what our LM provides

전체 문장을 생성!

- 첫 번째 토큰을 예측
- 예측이 완료된 첫 번째 토큰을 기반으로 두 번째 토큰을 예측!
- ...
- 첫 번째부터 (t-1) 번째 토큰을 기반으로 (t)번째 토큰을 예측
- 이 모두를 고려한다면 전체 문장을 생성하게 됨

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

the students opened their _____

- **Question**: How to learn a Language Model?
- **Answer** (pre- Deep Learning): learn an *n*-gram Language Model!

Definition: A *n*-gram is a chunk of *n* consecutive words.

- 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"

전체 코퍼스 상에서 n그램 빈도를 측정하도록 함!

Bigrams를 예시로 들자...

- 코퍼스 상에 존재하는 "the students" 문장을 검색
- "the student" 다음에 존재하는 토큰은 무엇일까??
- 다음에 나올 토큰들에 대한 빈도를 측정 → 통계 정보를 토대로 다음 단어를 예측!

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

먼저 $x^{(t+1)}$는 앞의 n-1 단어에만 의존한다는 마르코프 가정을 진행!

t+1 번째 단어는 t-1, t-2, ..., t-(n-1) 단어들에만 영향을 받으며, 그 이전의 단어들은 무시된다는 가정!

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n-1 \text{ words}})$$

(assumption)

n-1-gram 확률을 기반으로 n-gram와의 비율을 측정

prob of a n-gram

prob of a (n-1)-gram

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

(definition of conditional prob)

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

(statistical approximation)

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

Let's practice!!

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ *students opened their* _____

discard

condition on this

4-gram을 기준

3-gram을 조건부로 삼음! "student opend their"

그럼 다음 토큰으로 어떤 것이 오는가?

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

코퍼스 상에서 확률적으로 다음 토큰으로 "book"이 나올 것임!

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

N-gram model의 문제점...

1. 희소성 문제 (Sparsity Problem)

- 특정 n-gram(예: "students opened their w")이 데이터에서 한 번도 등장하지 않은 경우, 그 n-gram의 확률은 0
  → 해당 구문을 예측할 수 없음...

부분적 해결 방법 (Smoothing 기법)

- 모든 단어에 대해 작은 값 δ를 추가하도록 하여 확률이 0이 되지 않도록 함
- "students opened their"와 같은 구문 자체가 데이터에 없는 경우, 그 구문에 대한 모든 단어 w의 확률을 계산할 수 없다는 점이다.

"opened their"와 같이 더 짧은 구문을 조건으로 사용하는 백오프(Backoff) 기법을 사용하여 확률을 계산

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

2. 저장 문제 (Storage Problem)

- n-gram 모델은 훈련된 코퍼스에서 등장한 모든 n-gram에 대해 등장 횟수를 저장해야 함
  → n이나 코퍼스의 크기가 증가하면 저장해야 할 데이터 양도 기하급수적으로 늘어난다.

"students opened their ___"라는 n-gram에 대해 계산하기 위해서...

- 그 구문이 등장한 횟수와 그 구문 뒤에 이어지는 w에 대한 모든 경우의 수를 저장해야하므로 저장 문제는 더 심각해진다.

3. 문맥 부족 문제 (Context Limitation)

- n-gram 모델은 보통 3개에서 5개의 단어까지만 고려한다.
- 그 이상의 단어를 고려할 경우, **희소성 문제와 저장 문제가 더욱 악화**된다.
- 문맥이 부족하면 문장은 문법적으로는 맞을지라도, 의미적으로 **일관성 감소 문제가** 발생한다.

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

그럼 n-gram model이 아닌 neural model을 적용할 수 있도록 하자!

Basic idea

- Fixed window에 대한 entitiy를 인식
- 인식한 entity를 기반으로 다음 단어에 대한 확률 분포를 얻을 수 있음

Problem

만약 window를 넓힌다면...? → 학습을 위한 가중치도 급격하게 증가!

입력 벡터 $x^{(1)}$와 $x^{(2)}$ 간에 가중치 공유가 안된다는 문제점이 있음..?

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

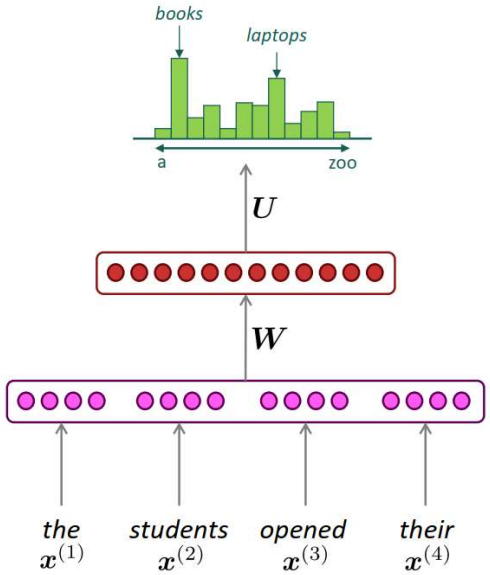
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

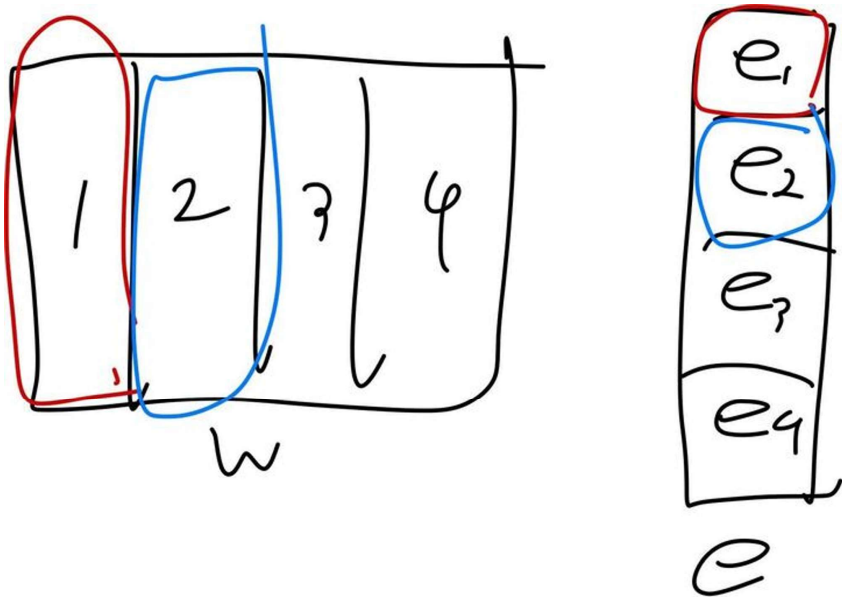
$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



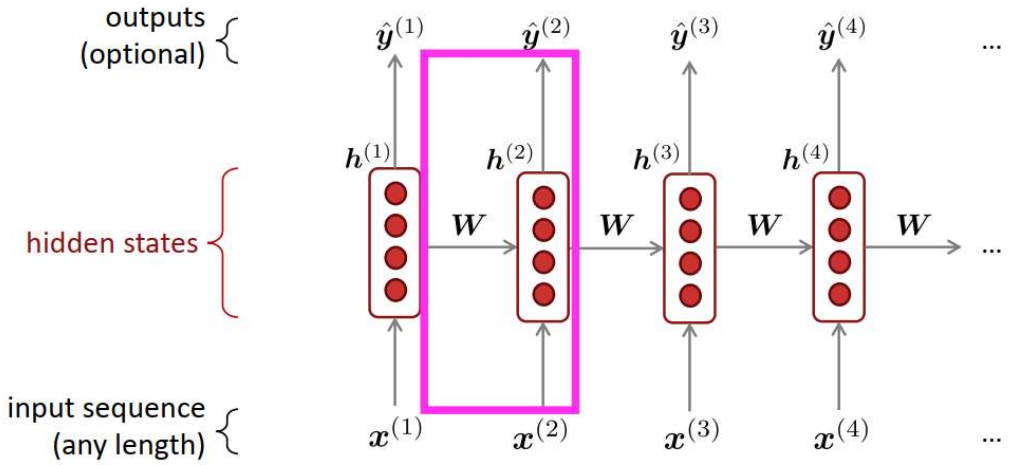
LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

입력 벡터 $x^{(1)}$와 $x^{(2)}$ 간에 가중치 공유가 안된다는 문제점이 있음

- 단어 임베딩을 처리하는 섹션이 공유되어야 단어간의 연관성을 찾을 수 있음
- MLP 기반 문제점은 단어 간의 연관성이 공유되지 않는다는 점!



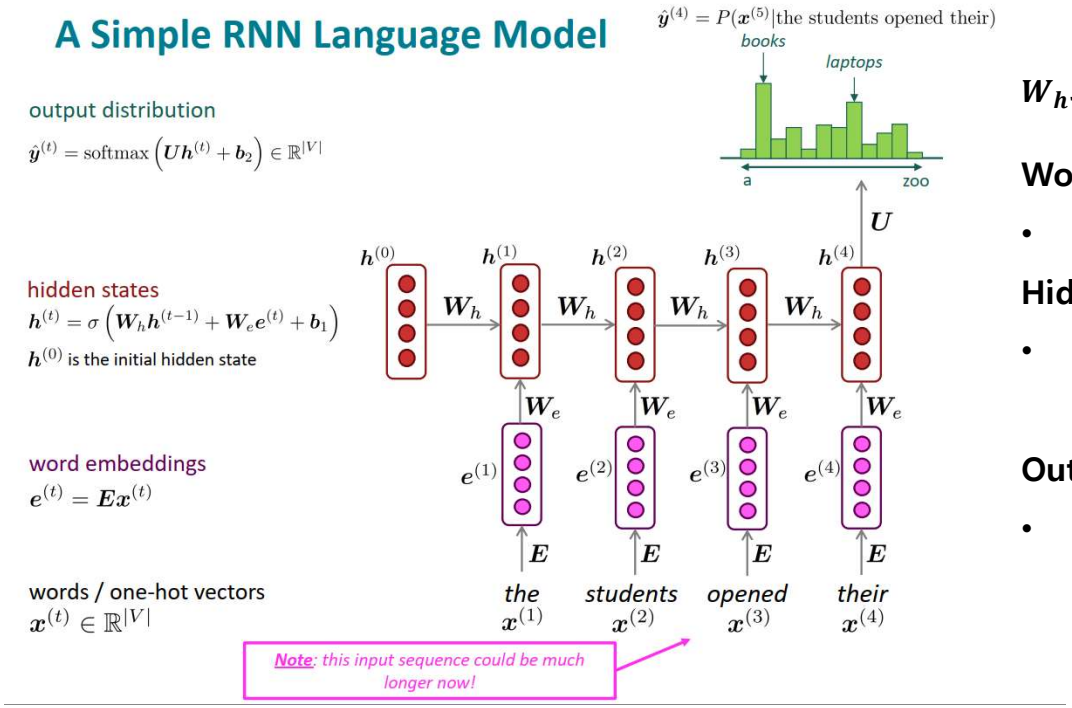
LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS



Basic Idea

- 모든 단계에서 동일한 가중치로 선형 변환 과정을 진행하도록 함
- 시간이 변함에 따라서 hidden state와 입력 벡터가 변함!!
- 이전 hidden state와 해당 단계 입력으로 이뤄짐

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS



W_h와 W_e를 학습하도록 함!

Word embedding

- Glove를 통해서 임베딩 과정을 진행! 필요에 따라서 파인튜닝

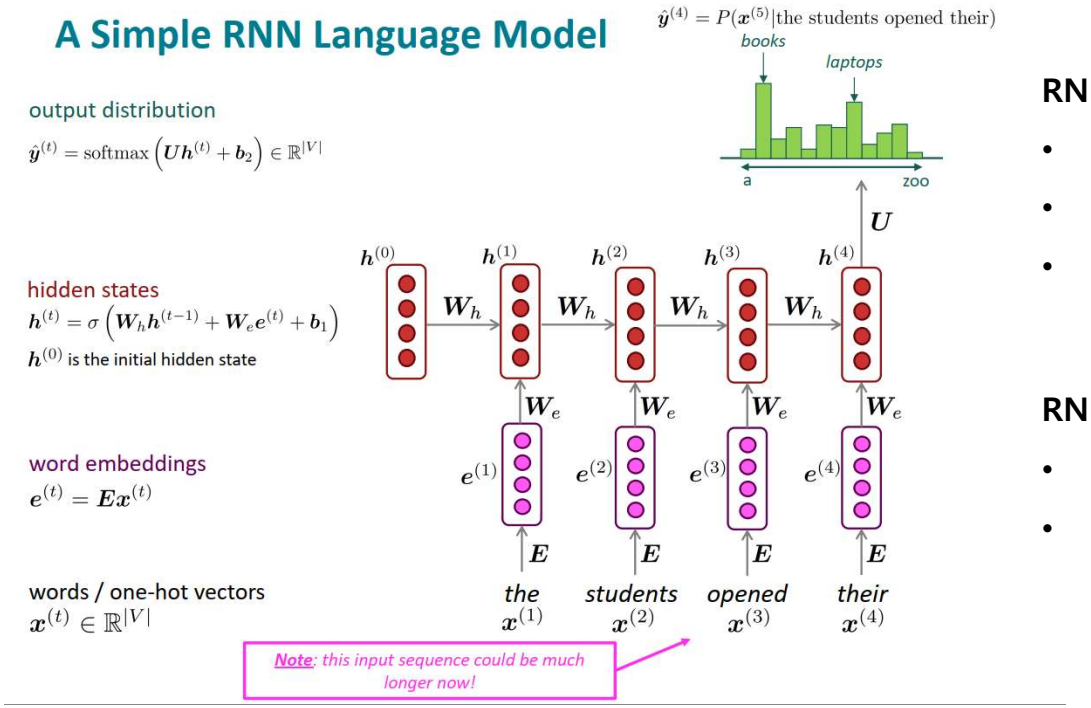
Hidden state

- 이전 hidden state와 현재 input vector를 기반으로 현재 hidden state를 결정

Output distribution

- 전체 시퀀스에 대한 입력 과정이 끝났으면, softmax를 기반으로 어휘안에 있는 단어 중에 다음 예측 단어를 생성

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS



RNN Advantage

- 입력 시퀀스의 길이 제한이 없음! (no need window)
- 이전 단어를 참고하여 다음 단어를 예측 가능함
- 각 입력 벡터에 대한 동일한 가중치를 공유하므로 단어간에 연관성을 찾을 수 있음

RNN Disadvantage

- Recurrent computiation이 느리다...
- 입력 시퀀스의 길이가 길어질수록 시퀀스의 정보를 획득하는 것이 매우 어렵다

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

How to train…?

굉장히 큰 코퍼스 $(x^{\{(1)\}}, x^{\{(2)\}}, \ldots, x^{\{(T)\}})$ 를 사용

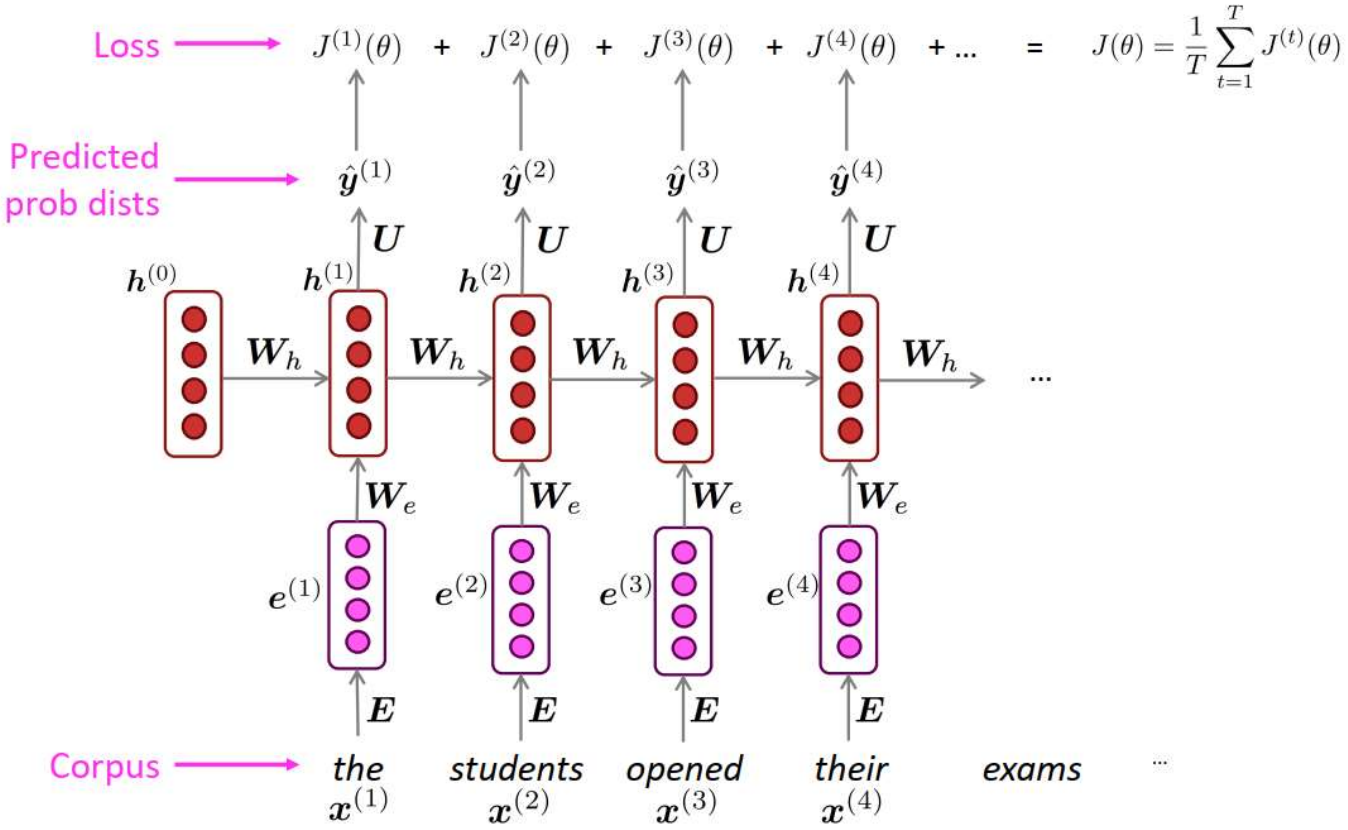
- 코퍼스를 RNN 언어 모델에 입력하여 각 시간 t에서 다음 단어의 확률 분포 $\hat{y}^{\{(t)\}}$를 계산

손실 함수로는 예측 확률 분포와 실제 다음 단어 간에 cross-entropy를 측정하도록 함

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS



신나는 역전파 계산 시간...

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

$$\frac{\partial J^{(4)}(\theta)}{\partial h^{(1)}} = (\text{residual Error}) \times \text{Inital hidden state.}$$

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_x x^{(t)} + b)$$

(1) W_h : $h^{(t-1)}$를 미분 ← focus!.

(2) W_x : $x^{(t-1)}$를 미분

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

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

LECTURE 5:
LANGUAGE MODELS AND RECURRENT NEURAL NETWORKS

- The standard evaluation metric for Language Models is perplexity.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

- This is equal to the exponential of the cross-entropy loss $J(\theta)$:

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} - \log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

Lower perplexity is better!

LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS

$$\frac{\partial J^{(4)}(\theta)}{\partial h^{(1)}} = (\text{residual Error}) \times \text{Inital hidden state.}$$

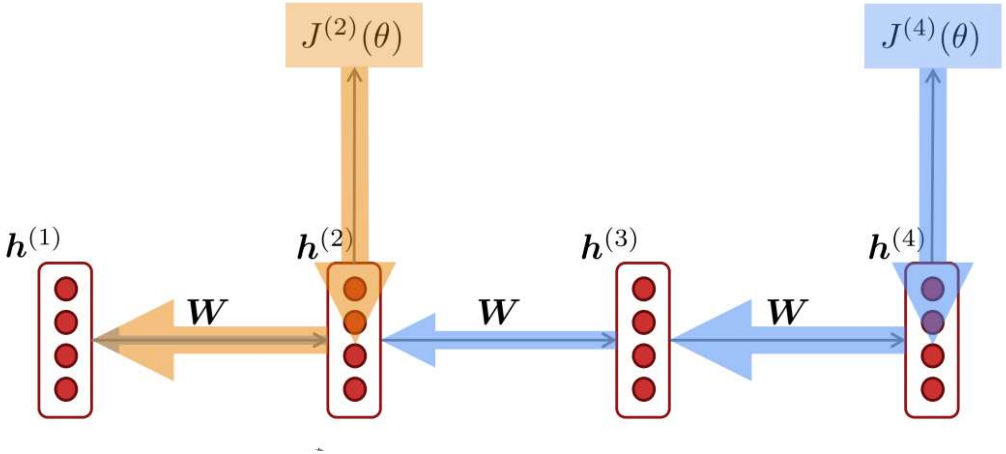
$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_x x^{(t)} + b)$$

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

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

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

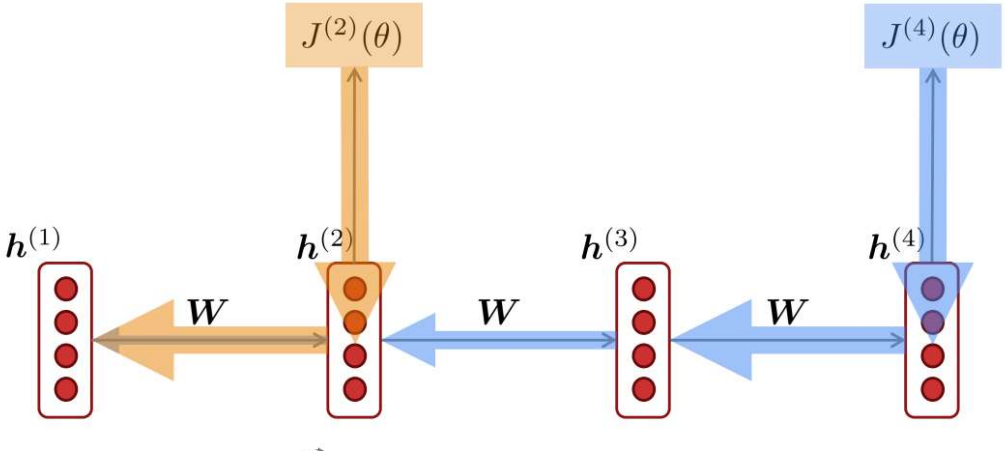
LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS



가중치 업데이트 시에 가까운 신호에 의해 결정될 수 있는 문제가 발생

- Short-term effect에 의한 가중치 업데이트 의존도가 매우 높음
- 반대로 Long-term effect에 대한 가중치 업데이트 의존도가 매우 낮다는 문제가 발생

LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS



가중치 업데이트 시에 가까운 신호에 의해 결정될 수 있는 문제가 발생

- Short-term effect에 의한 가중치 업데이트 의존도가 매우 높음
- 반대로 Long-term effect에 대한 가중치 업데이트 의존도가 매우 낮다는 문제가 발생

문장이 길어질 경우, dependency를 학습할 수 없다는 문제점이 발생함!

LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS

Lone Short-Term memory (LSTM)을 통해서 lone-term dependency 문제를 해결하도록 함!

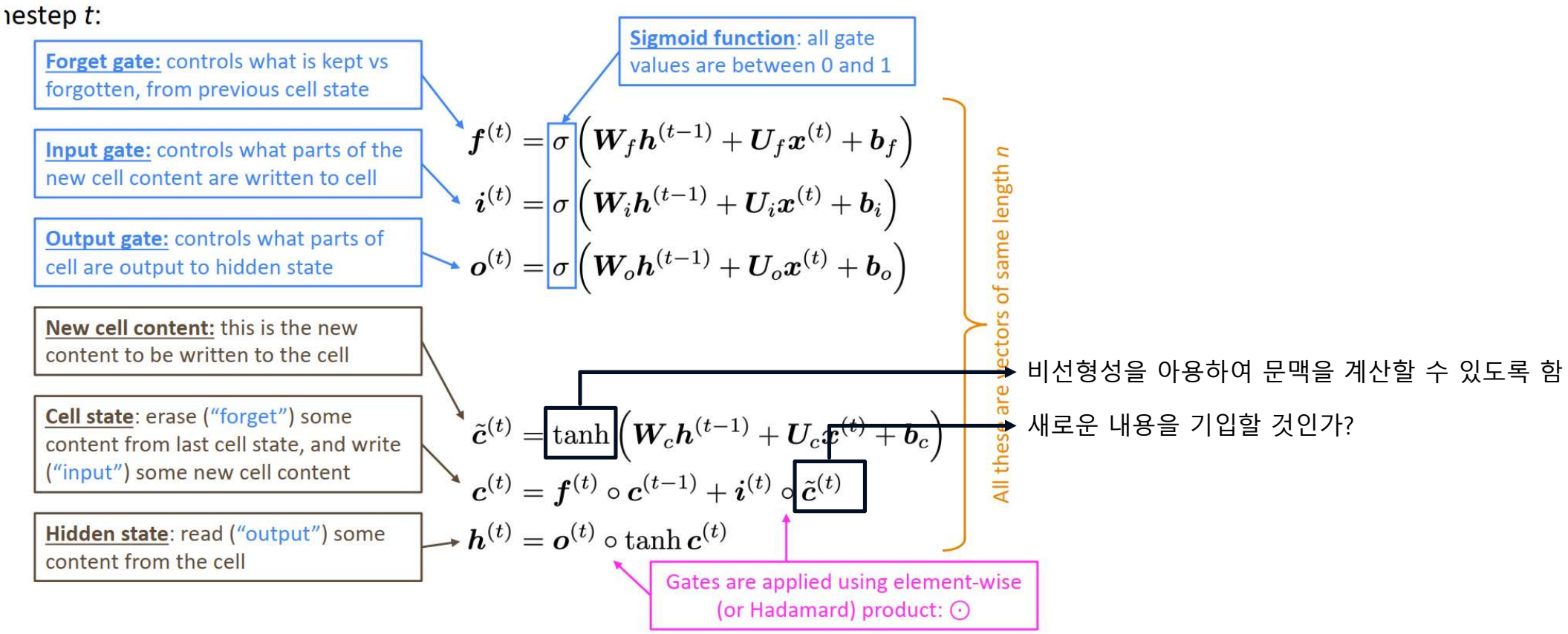
Hidden state와 cell state

- Cell state는 long-term information을 유지할 수 있도록 하는 뱅크
- Cell을 통해서 information을 다음과 같이 조작할 수 있음: erase, write and read!

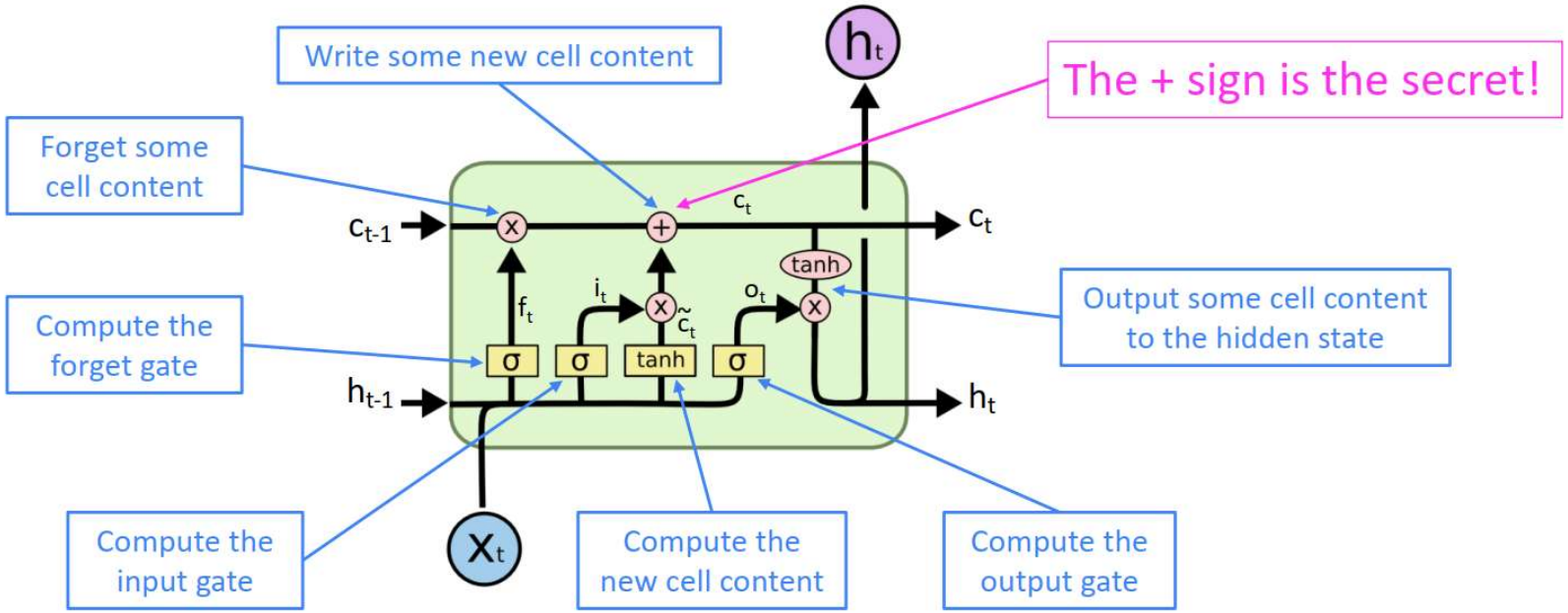
Information을 토대로 erase, write 그리고 read 기능은 gate에 의해서 결정됨

- 각 timestep마다 게이트는 open (1) 그리고 closed (0)를 통해서 information을 다루도록 함
- 각 step마다 gate 동작이 다름

LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS



LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS



RNN의 long-term dependency 문제를 **어느정도** 해결해줌

- 시점마다 변하도록 설계된 아키텍쳐의 한계로 인해 100% 보장할 수 없다는 점 (아키텍쳐 자체의 한계!)

LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS

Is vanishing/exploding gradient just a RNN problem?

- 기본적으로 Deep neural network에서 발생하는 문제!!
- Layer가 많아질수록, chain-rule에 의해서 gradient의 값이 점차 작아짐 → lower layer에서 계산되는 gradient가 매우 작아 학습이 어려움

Add more direct connection

$$F'(x) = F(x) + x$$

$$\frac{\partial F'(x)}{\partial x} = \frac{\partial F(x)}{\partial x} + 1 \quad \Rightarrow \quad \frac{\partial F(x)}{\partial x} = \frac{\partial F'(x)}{\partial x} - 1$$

이전의 information update로 Sensitive하게 반영.

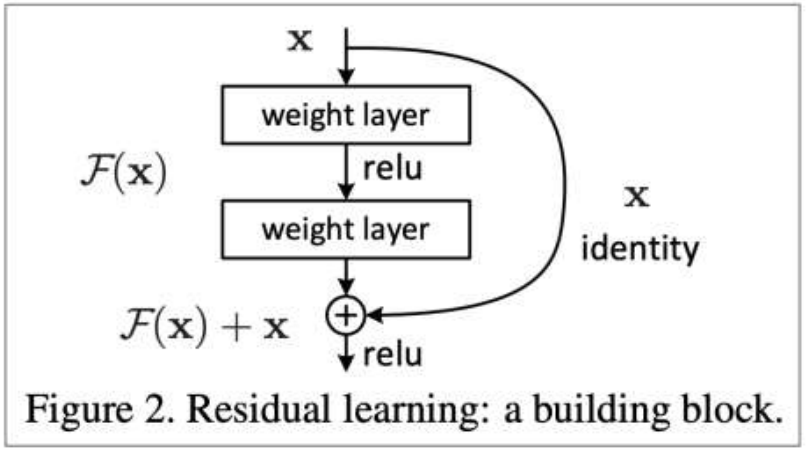
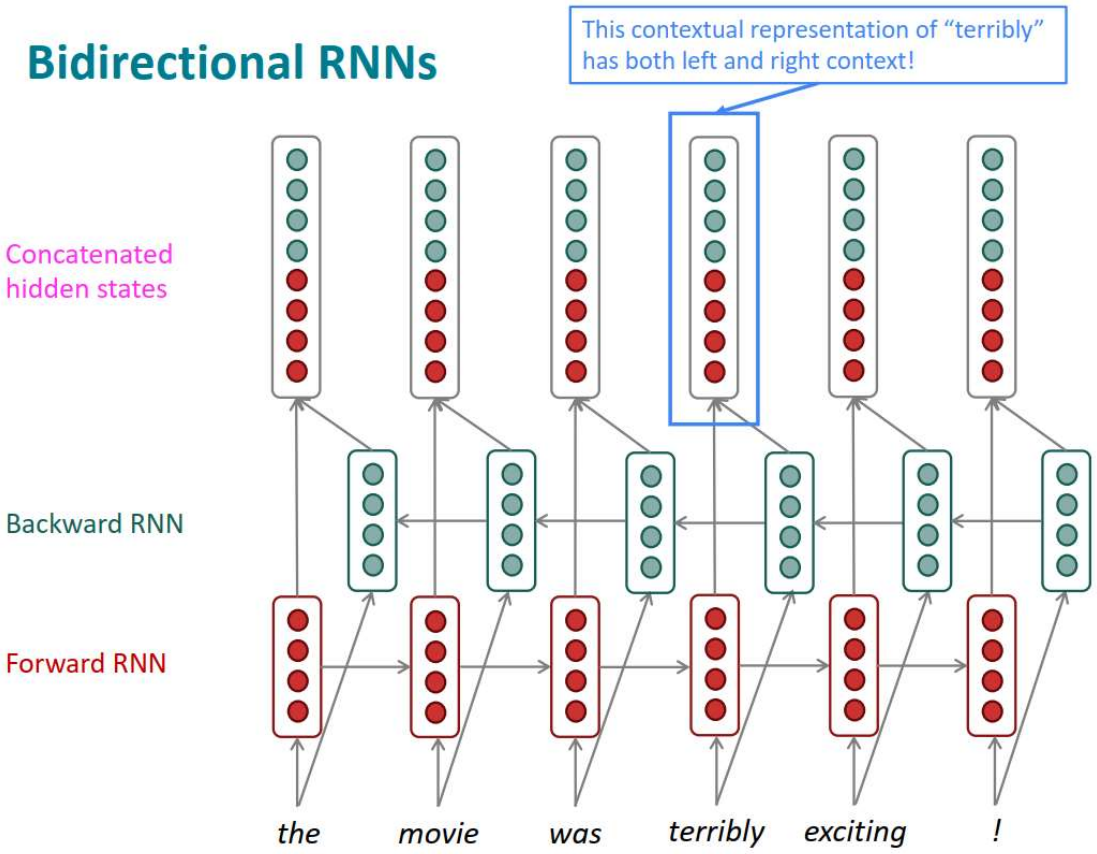


Figure 2. Residual learning: a building block.

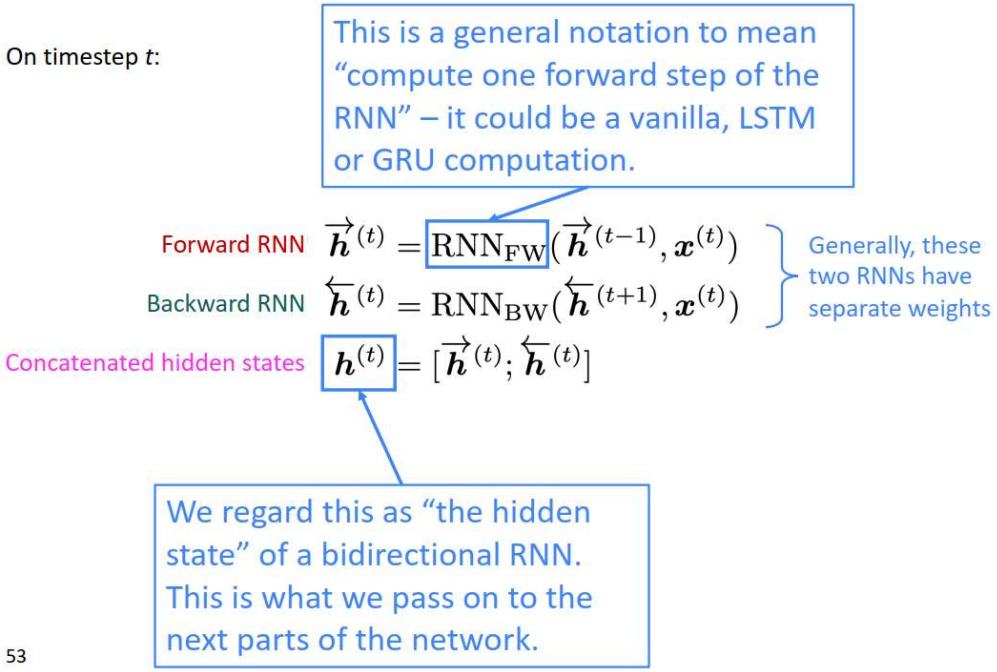
LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS



위의 예시처럼 감정을 기계가 제대로 이해하려면?

관계성을 (왼쪽→오른쪽) 뿐만 아니라 (오른쪽→왼쪽)도 분석하도록! Bidirectional RNNs

LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS



53

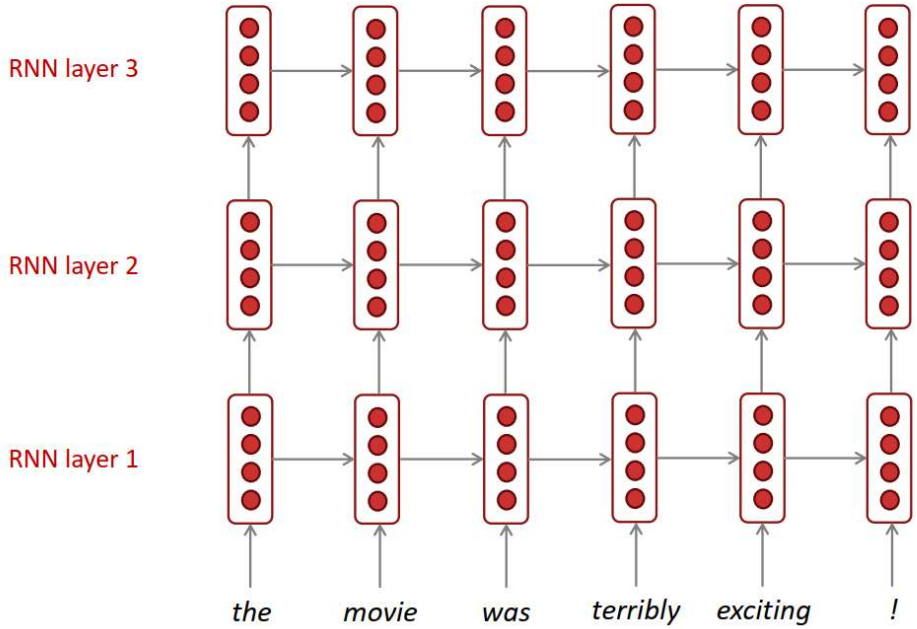
Bidirectional RNN은 전체 입력 시퀀스에 액세스할 수 있는 경우에만 적용 가능

- LM에서는 왼쪽 컨텍스트만 사용할 수 있으므로 언어 모델링에는 적용되지 않음
- 인코딩과 같은 전체 입력 시퀀스가 있는 경우, 강력한 성능을 제공 (따라서 이게 국룰)
- BERT(Bidirectional Encoder Representations from Transformers)는 Bidirectional을 기반으로 구축된 pretrained contextual representation system!

LECTURE 6:
SIMPLE AND LSTM RECURRENT NEURAL NETWORKS

Multi-layer RNNs

The hidden states from RNN layer *i* are the inputs to RNN layer *i*+1



여러 개의 RNN 레이어를 적용한 Multi-layer RNN

- 단일 레이어로는 처리할 수 없는 더 복잡한 표현을 학습할 수 있다.
- 하위 RNN 레이어는 더 낮은 수준의 특징을 학습하고, 상위 RNN 레이어는 더 높은 수준의 특징을 학습하여 복잡하고 추상적인 표현을 처리할 수 있다.