

연 세 대 학 교 통 계 학 회 E S C

20FALL Week3

LM and RNN, Fancy RNN

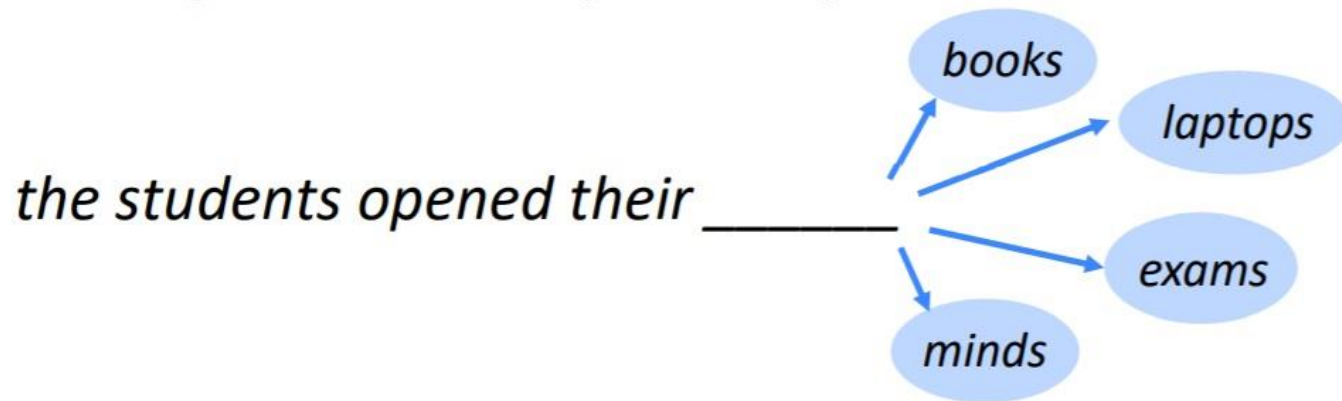


INDEX

Natural Language Processing

1. Language Modeling
2. RNN
3. RNN variants

LANGUAGE MODELING



Language Model

문장에 이어지는 다음 낱말을 예측하는 Task

LANGUAGE MODELING

$$P(\mathbf{x}^{(t+1)} \mid \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})$$

Conditional Distribution

앞의 단어들에 대한 다음 단어의 조건부 확률 분포 구하기

LANGUAGE MODELING

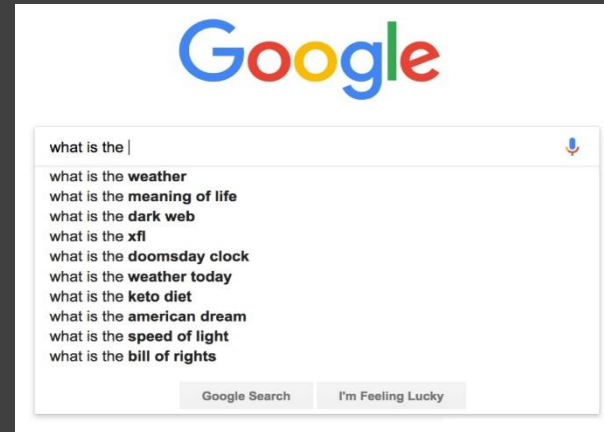
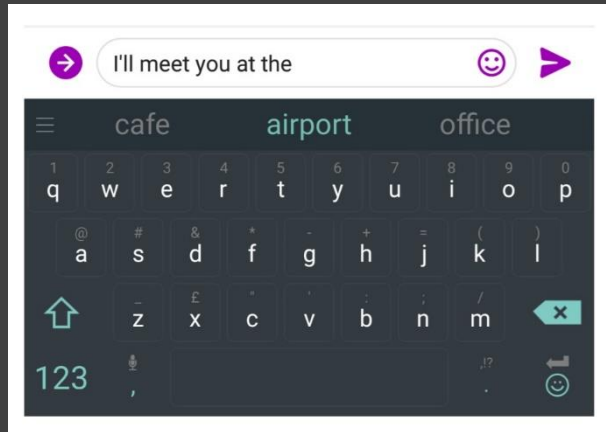
$$\begin{aligned} P(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}) &= P(\mathbf{x}^{(1)}) \times P(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}) \times \dots \times P(\mathbf{x}^{(T)} | \mathbf{x}^{(T-1)}, \dots, \mathbf{x}^{(1)}) \\ &= \prod_{t=1}^T P(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}) \end{aligned}$$

This is what our LM provides

Probability Assigning

각 단어에 확률을 부여하는 System

LANGUAGE MODELING



Example

Ex) 검색어 자동 완성 system

How To Learn Language Model?

n-gram LANGUAGE MODEL

- 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

n-gram : n개의 연속되는 단어 문치

n-gram LANGUAGE MODEL

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)}) = P(\mathbf{x}^{(t+1)} | \underbrace{\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}}_{n-1 \text{ words}})$$

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

1. Markov assumption

: 예측할 단어를 포함해 n개의 단어들만!

2. Probability

: Corpus에서 해당 단어들이 포함된 문장들의 빈도를 계산

n-gram LANGUAGE MODEL

~~as the proctor started the clock, the~~ students opened their _____
discard condition on this

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

Probability Assigning

Markov assumption으로 인해 문맥을 반영하지 못하는 문제 발생!

Ex) discard 'proctor' $\rightarrow P(\text{book} | \text{n-gram}) > P(\text{exam} | \text{n-gram})$

n-gram LANGUAGE MODEL

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

Sparsity Problem 1

Corpus에 'students opened their ω ' 라는 chunk가 없다면?
→ $P(\omega \mid \text{n-gram}) = 0$

Solution : 모든 단어에 작은 확률 δ 을 부여 → Smoothing

n-gram LANGUAGE MODEL

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

Sparsity Problem 2

Corpus에 'students opened their' 라는 chunk가 없다면?
→ Denominator = 0

Solution : 'opened their' chunk의 빈도수로 대체 → Backoff

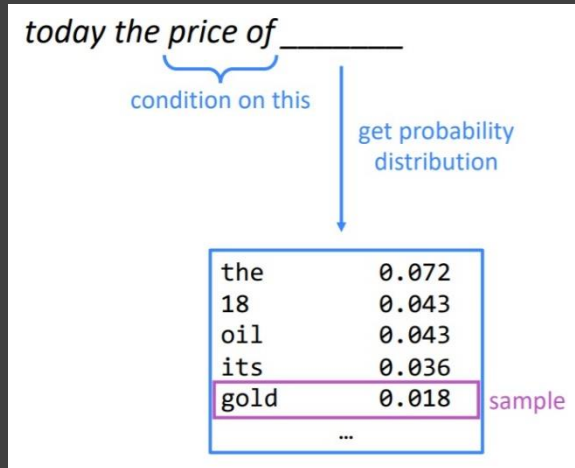
n-gram LANGUAGE MODEL

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

Storage problem

Corpus에 모든 chunk를 저장해야 한다!
→ Model size가 너무 커져버린다

n-gram LANGUAGE MODEL

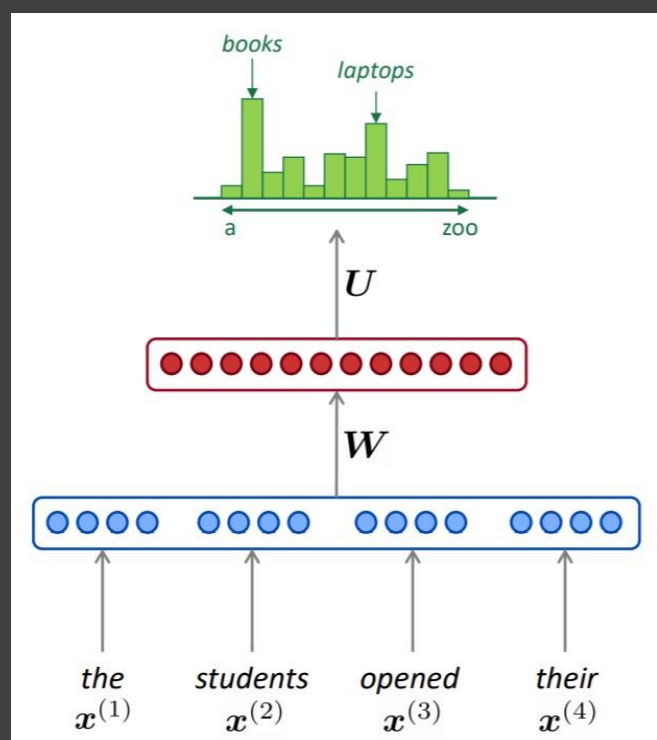


today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .

Generating text

Grammatical but Incoherent

NEURAL LANGUAGE MODEL



1. Window-based
: 'the students opened their' 을
fixed window로!

2. Neural model
: One-hot vectors \rightarrow Embedding
 \rightarrow Hidden Layer \rightarrow Output

3. Output
: $P(\omega \mid \text{n-gram})$

NEURAL LANGUAGE MODEL

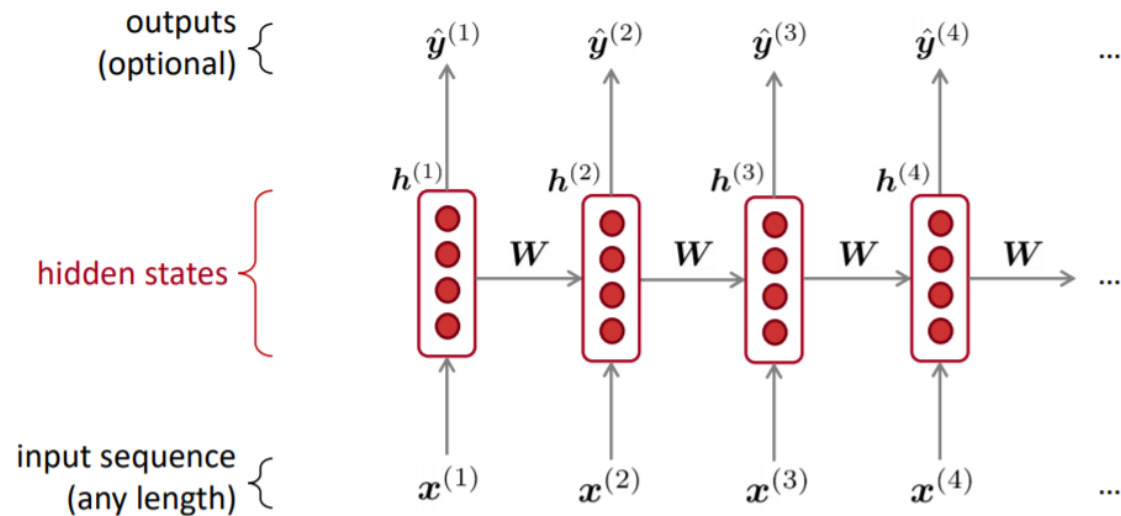
Improvement

1. No Sparsity problem
2. No Storage problem

Remaining problem

1. Too small fixed window
2. No symmetry

Recurrent Neural Networks (RNN)



기존의 NN모델: 모든 단어에 대응되는 행으로 구성된 W 사용

- 비경제적
- fixed window
- sparsity problem

Recurrent Neural Networks (RNN)

one-to-many

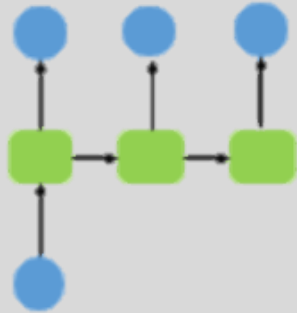


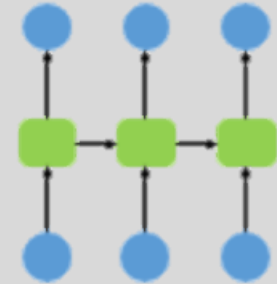
Image Captioning

many-to-one



Sentiment Classification
Document Classification

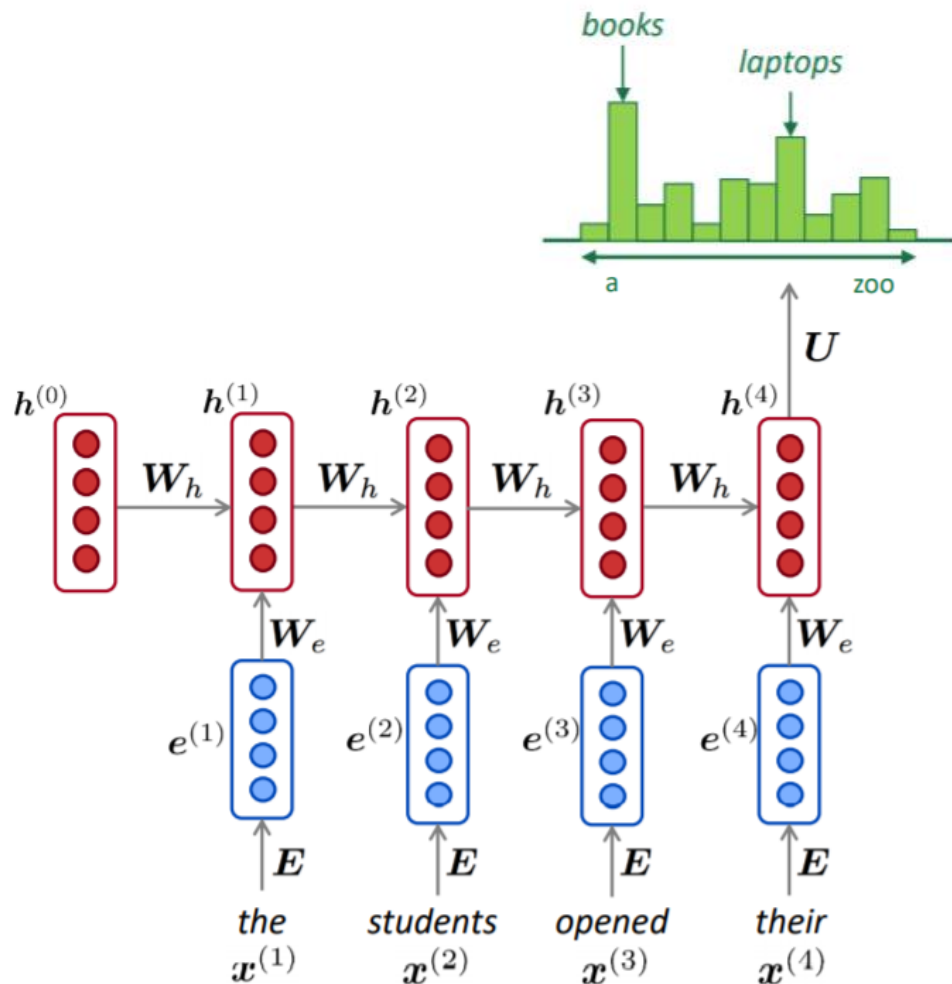
many-to-many



챗봇, 번역기
NER
품사태깅

Recurrent Neural Networks (RNN)

$$\hat{y}^{(4)} = P(x^{(5)} | \text{the students opened their})$$



1. Input

word embeddings

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

E : embedding matrix
(pre-trained, word2vec, ...)

2. hidden state

2- 1 Linear transformation on previous hidden state & current input

$h^{(0)}$ is the initial hidden state

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

2- 2 non-linearity function

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

3. output

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

Recurrent Neural Networks (RNN)

The diagram illustrates the computation of the hidden state $h^{(t)}$ at time step t . It shows the following components and operations:

- A green grid representing the hidden-to-hidden weight matrix W_h with dimensions $D_h \times D_h$.
- A red vertical vector representing the previous hidden state $h^{(t-1)}$ with dimensions $D_h \times 1$.
- A green grid representing the input-to-hidden weight matrix W_e with dimensions $D_h \times d$.
- A blue vertical vector representing the input vector $e^{(t)}$ with dimensions $d \times 1$.
- A grey vertical vector representing the bias b_1 with dimensions $D_h \times 1$.
- A red vertical vector representing the current hidden state $h^{(t)}$ with dimensions $D_h \times 1$.

The equation is: $\sigma \left(W_h \times h^{(t-1)} + W_e \times e^{(t)} + b_1 \right) = h^{(t)}$

d : 단어벡터의 차원

Advantages

- Can process **any length** input
- Model size doesn't increase for longer input (size of model is fixed)
- Same weights applied on every timestep

Disadvantages

- Slow
- In practice, difficult to access information from many steps back

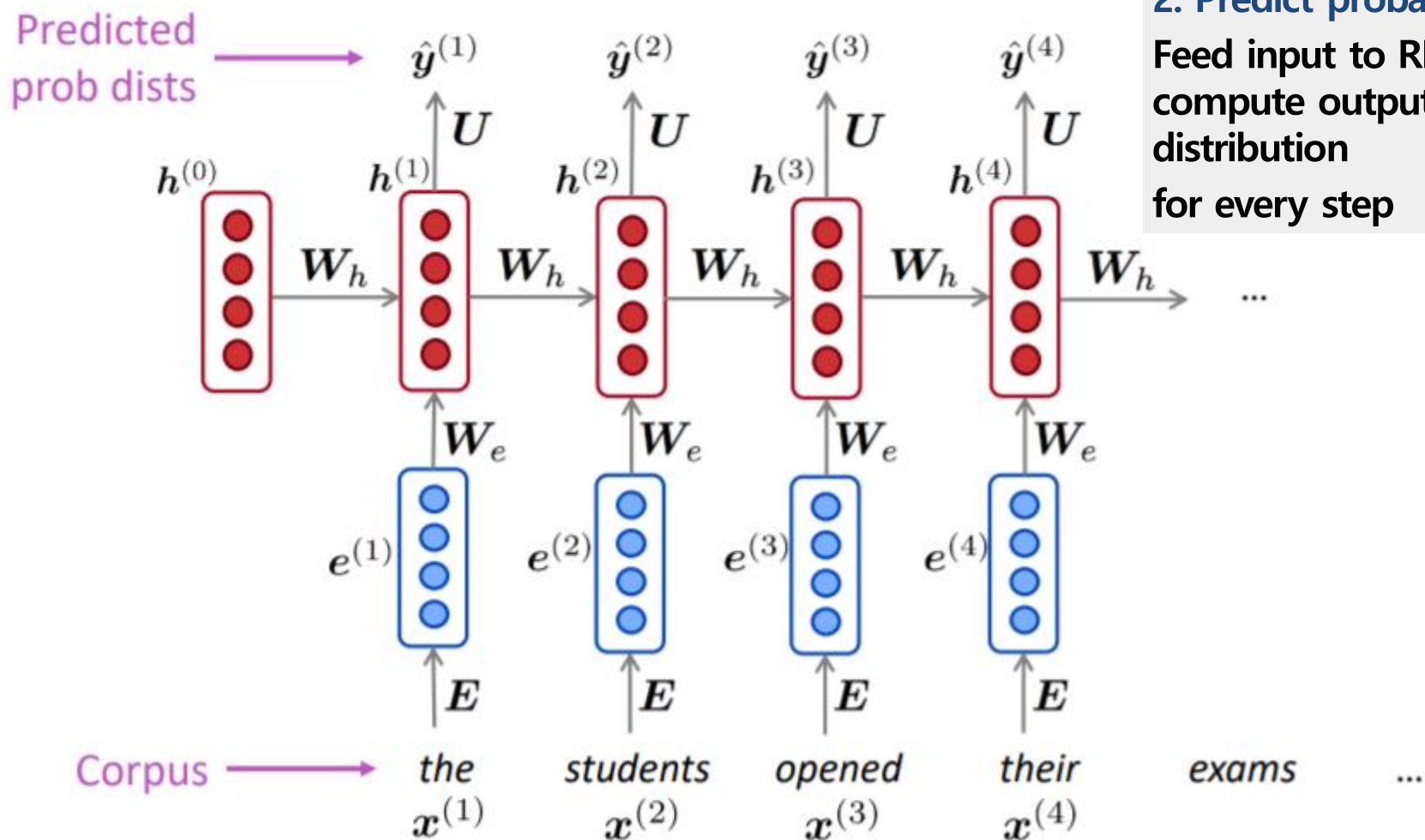
Training a RNN Language Model

1. Input

Get a big corpus of text
which is a sequence of words

Corpus \longrightarrow *the* *students* *opened* *their* *exams* ...
 $x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$

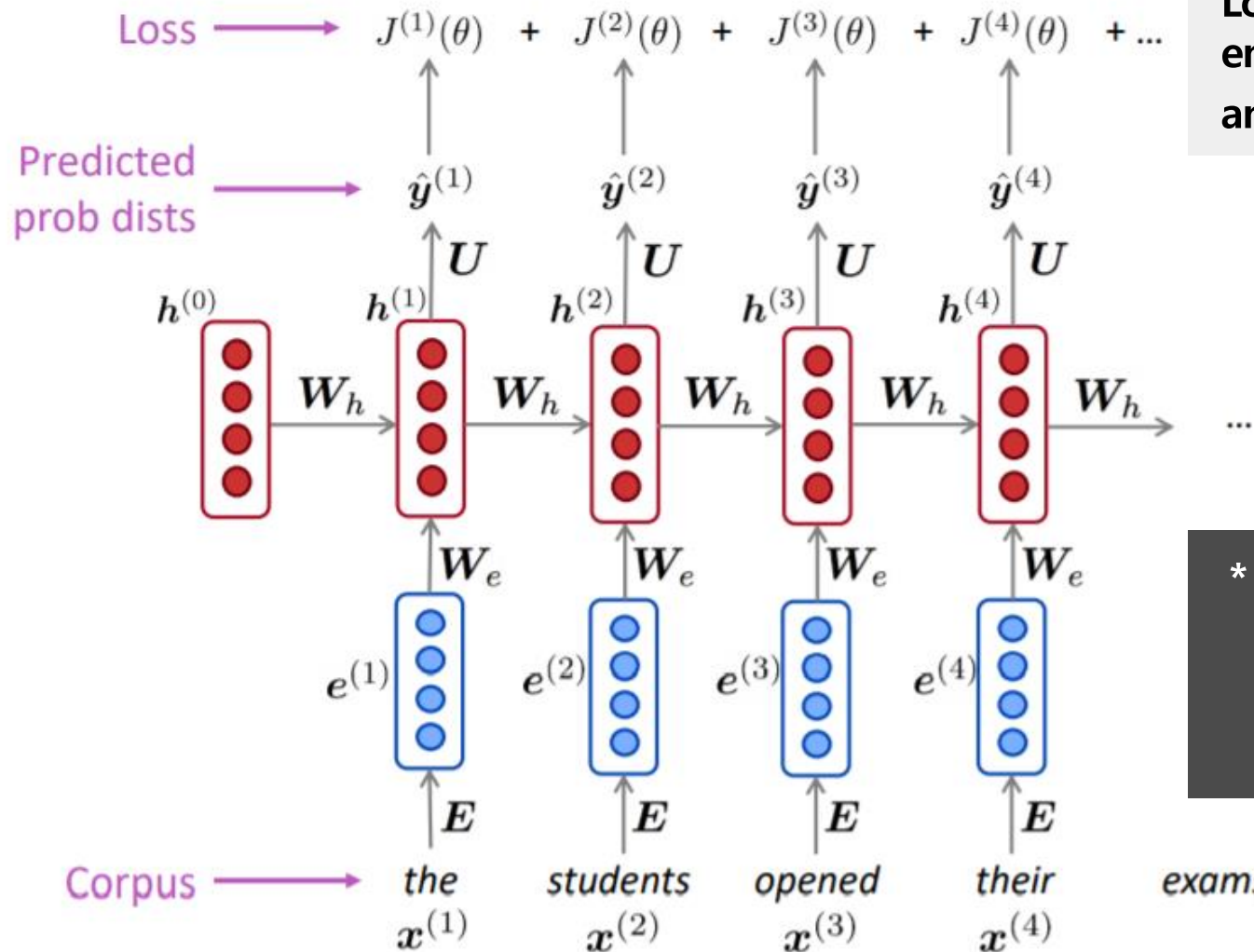
Training a RNN Language Model



2. Predict probability dist

Feed input to RNN,
compute output
distribution
for every step

Training a RNN Language Model



3. Loss

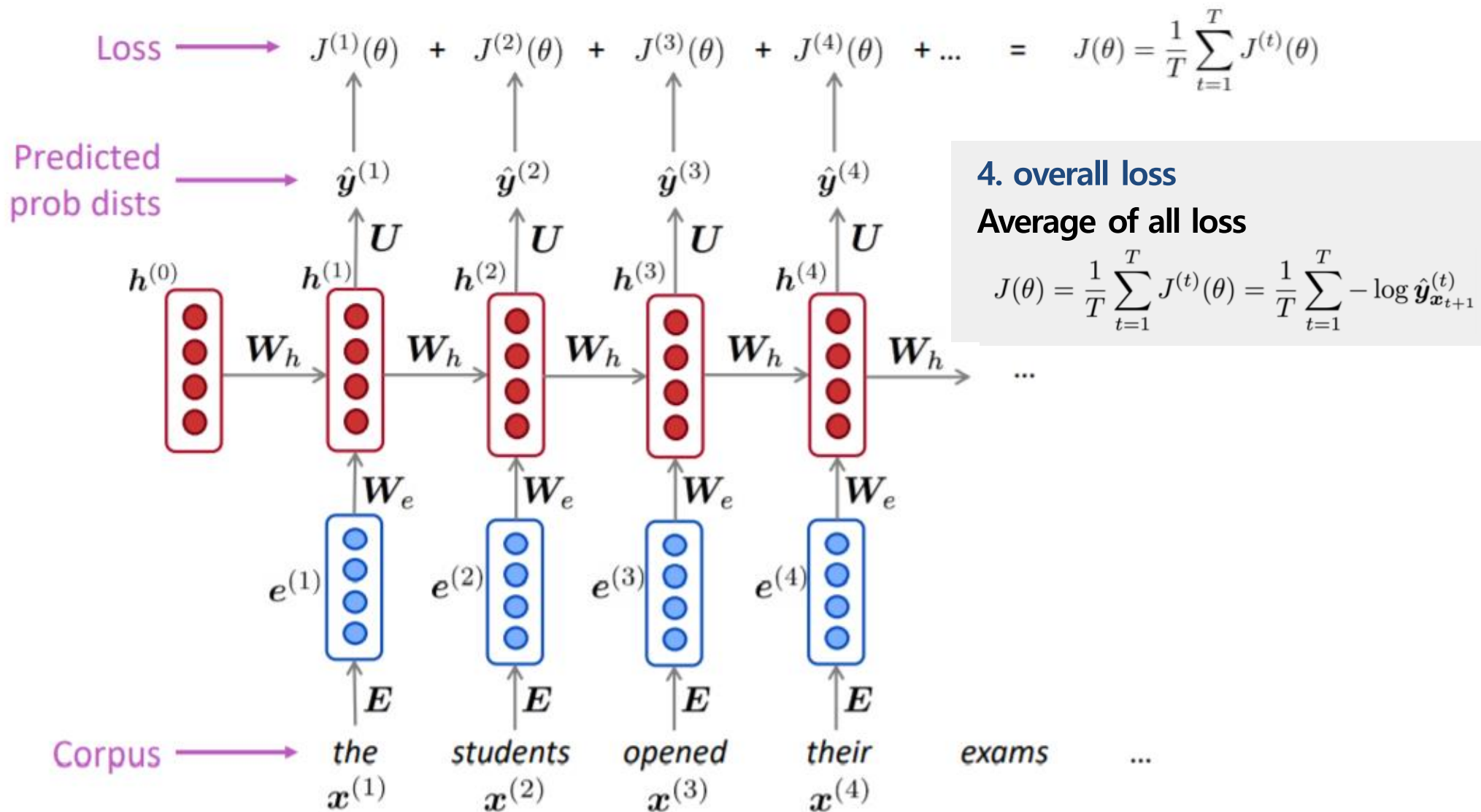
Loss of step t is cross-entropy between $\hat{y}^{(t)}$ and the true next word $y^{(t)}$

$$\begin{aligned} J^{(t)}(\theta) &= CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) \\ &= - \sum_{w \in V} \mathbf{y}_w^{(t)} \log \hat{\mathbf{y}}_w^{(t)} \\ &= - \log \hat{\mathbf{y}}_{x_{t+1}}^{(t)} \end{aligned}$$

* cross-entropy

$$H_p(q) = - \sum_{c=1}^C q(y_c) \log(p(y_c))$$

Training a RNN Language Model



Evaluating Language Models: Perplexity

Perplexity: standard evaluation metric for Language Models → **Lower**, the better

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

Normalized by number of words

Inverse probability of corpus, according to Language Model

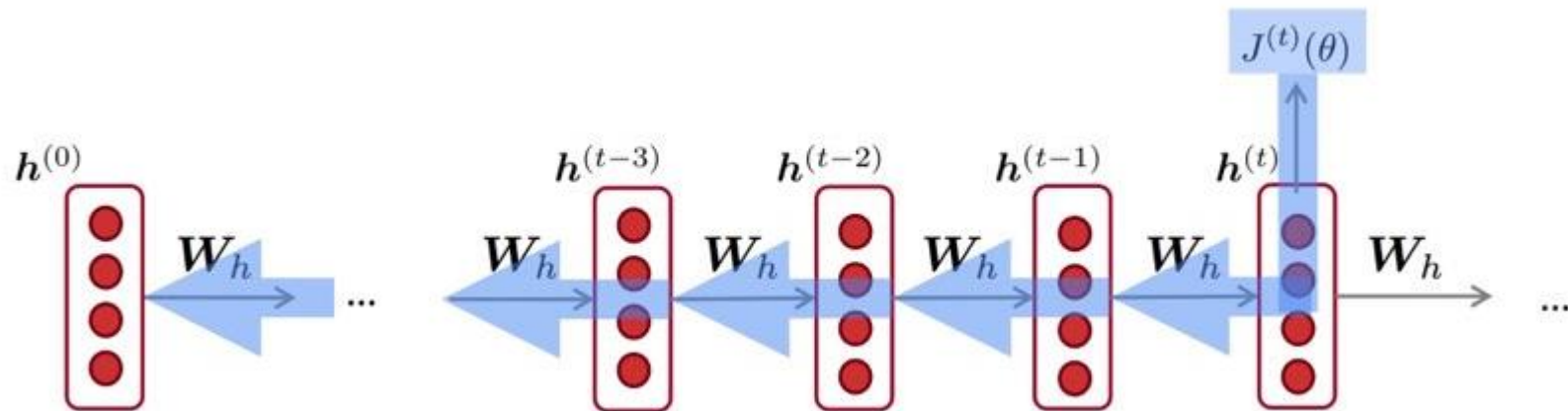
간단히 말해, **혼란한 정도**를 나타내는 지표
특정 확률 모델이 실제로 관측되는 값을 얼마나 잘 예측하는지를 나타내는 지표

Perplexity = exponential of the cross-entropy loss

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

즉, loss를 줄이면 자연스럽게 perplexity도 줄어들게 된다
따라서 RNN은 perplexity를 줄이는 데에 매우 효과적!

Backpropagation for RNNs



$$\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^t \left. \frac{\partial J^{(t)}}{\partial W_h} \right|_{(i)}$$

→ How do we calculate this?

Backpropagate over timesteps $l = t, \dots, 0$, summing gradients as you go.

This algorithm is called **“backpropagation through time”**

Generating text with RNN

Title: CHOCOLATE RANCH BARBECUE

Categories: Game, Casseroles, Cookies, Cookies

Yield: 6 Servings

2 tb Parmesan cheese -- chopped

1 c Coconut milk

3 Eggs, beaten

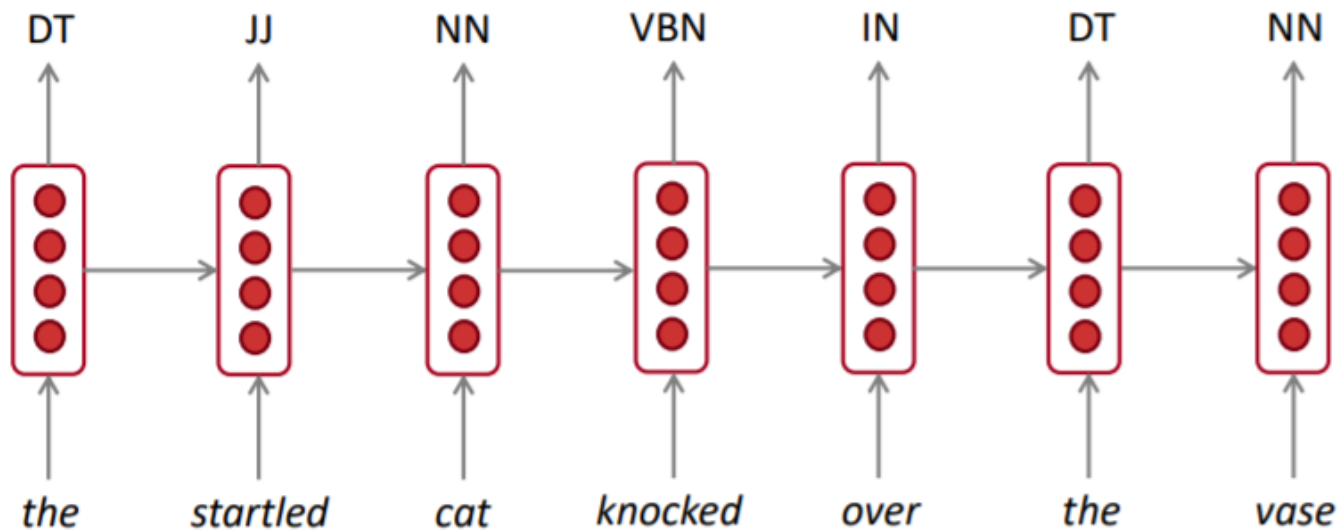
Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer until firm. Serve hot in bodied fresh, mustard, orange and cheese.

Combine the cheese and salt together the dough in a large skillet; add the ingredients and stir in the chocolate and pepper.

Problem

- " Inability to remember what is happening overall "
- 레시피 예시의 경우 'title'을 기억하며 의미적으로 유사한 재료나 요리 방법에 해당하는 단어를 선택하는 것이 불가능하므로 어색한 text가 생성된다

Tagging with RNN

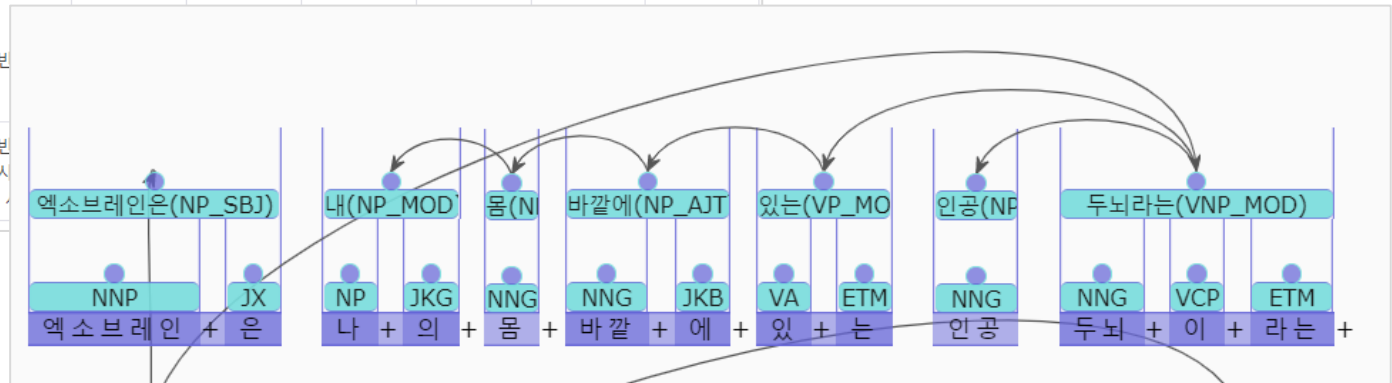


Tagging example: 엑소브레인

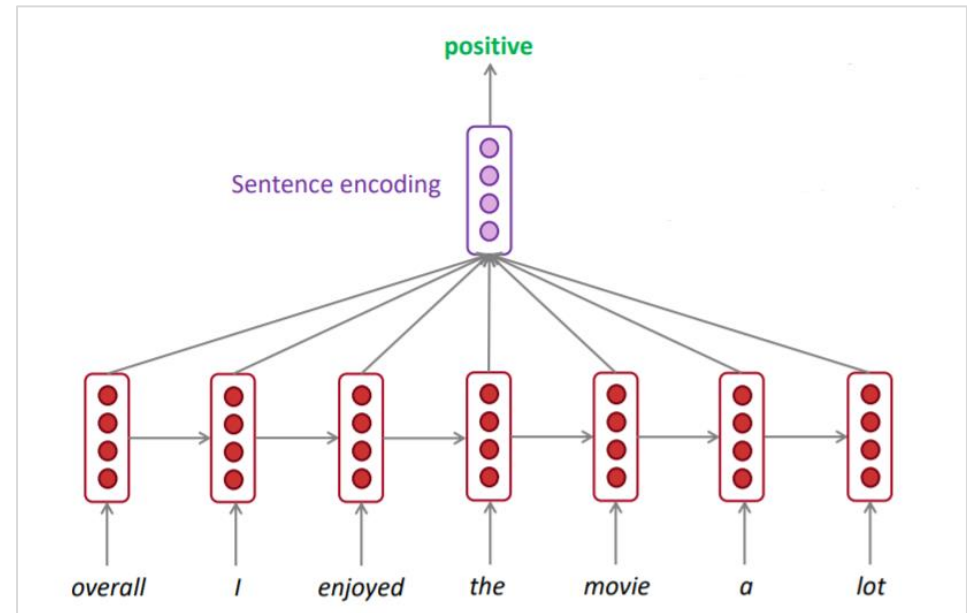
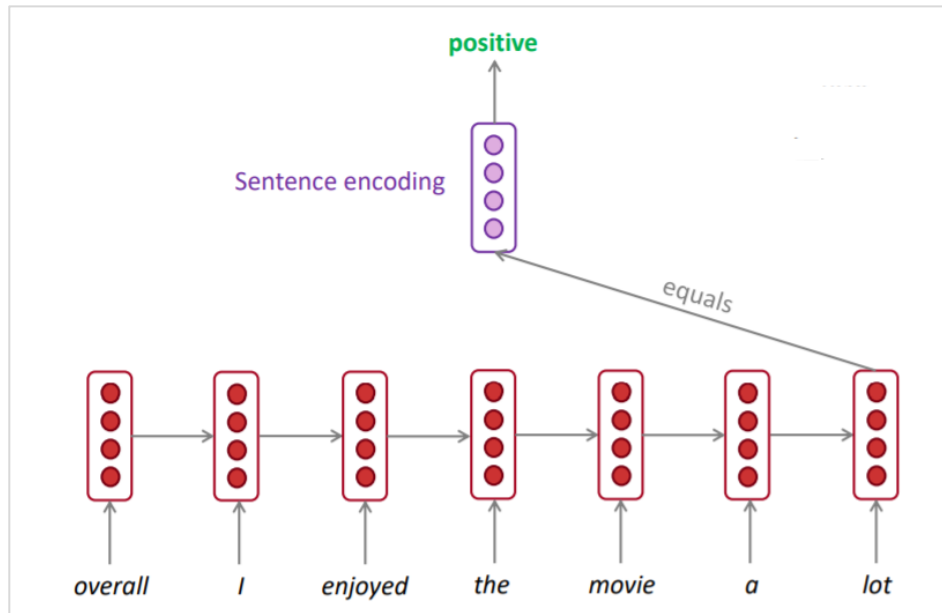
언어 분석 API(문어)

엑소브레인은 내 몸 바깥에 있는 인공 두뇌라는 뜻으로, 세계 최고인공지능 기술 선도라는 비전을 달성하기 위한 과학기술정보통신부 소프트웨어 분야의 국가 혁신기술 개발형 연구개발 과제이다.

No	Word	형태소			등음이의어			개체명		
		단어	태그	설명	단어	형태소 태그	의미번호	단어	태그	설명
0	엑소브레인은	엑소브레인 은	NNP JX	고유명사 보격조사	엑소브레인 은	NNP JX	00 00	엑소브레인	TMI_PROJECT	프로젝트 명칭
1	내	나의	NP JKG	대명사 관형격조사	나의	NP JKG	03 00			
2	몸	몸	NNG	일반						
3	바깥에	바깥 에	NNG JKB	일반						



Sentence Classification with RNN



방법

- 첫번째: 가장 마지막 hidden state를 sentence encoding 값으로 사용
- 두번째: 모든 hidden state의 element-wise max값이나 평균을 사용 (더 나은 성능을 보임)

감성분석 example: ADAMS

영화 '물란' 네이버 리  4

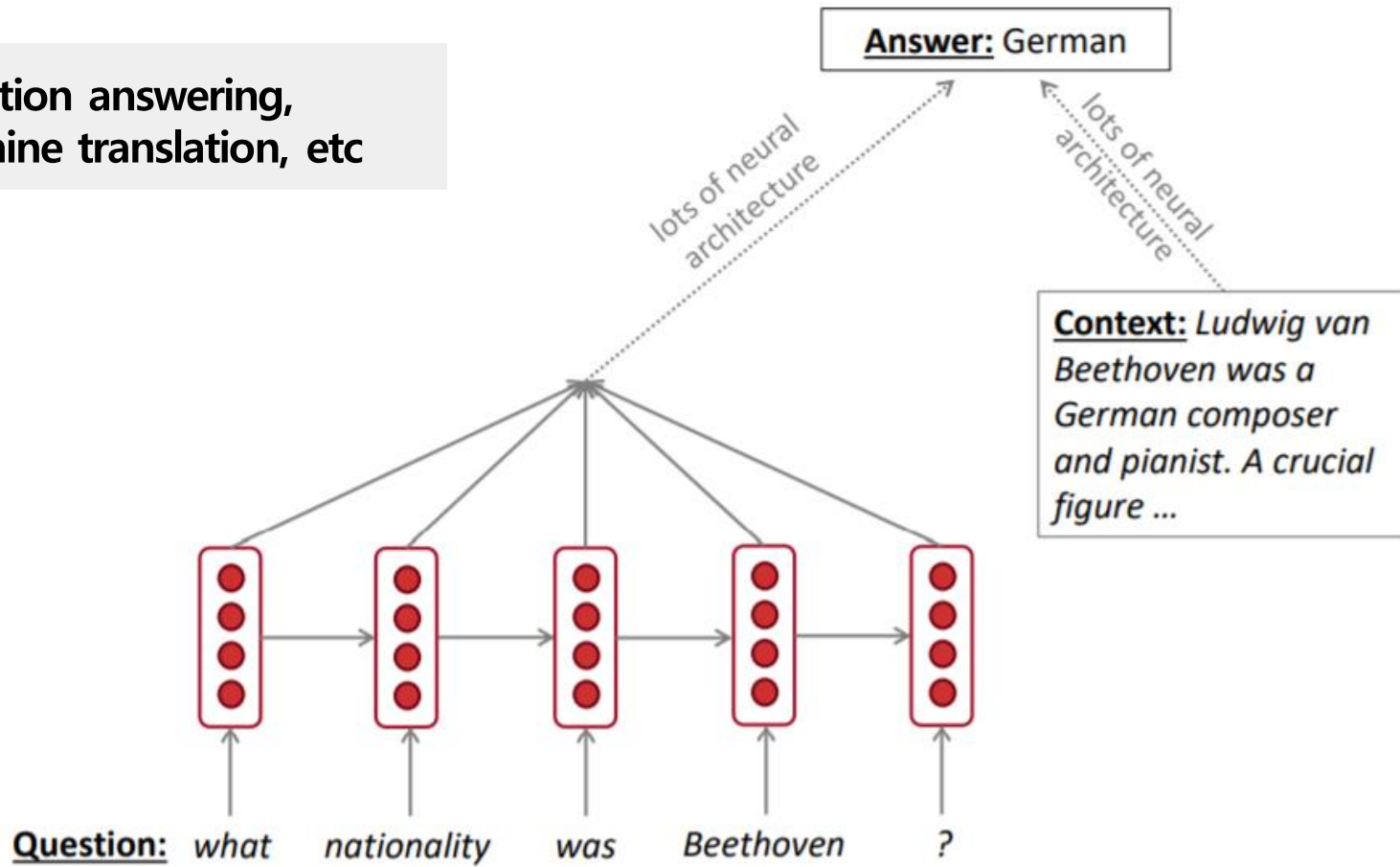
```
return_object: {  
  query: "킬링타임용 영상미는 좋으나 스토리등 유치하고 별로 원작보다노잼임",  
  type: "감성분석",  
  score: 0.9994524121284485,  
  label: "부정"  
},
```

영화 '오! 문희' 네이버  10
리뷰

```
return_object: {  
  query: "중후반부터 나문희 선생님 얼굴에서 울 어머니 얼굴이 보이는 매직이 펼쳐짐",  
  type: "감성분석",  
  score: 0.9994671940803528,  
  label: "긍정"  
},
```

Encoder module with RNN

Question answering,
machine translation, etc



데모

질의응답 example: 엑소브레인

위키백과 QA API

질문 선택

김구가 누구야?



원질문	김구가 누구야?	
질의응답 방식	hybridqa	
정답 순위	신뢰도	정답
1	0	김구(金九, 1876년 8월 29일 (1876년 음력 7월 11일) - 1949년 6월 26일)는 일제강점기 독립운동가이자 대한민국의 통일운동가, 정치인이다. 의열단체 한민애국단을 이끌었고 대한민국 임시 정부 주석을 역임하였으며 1962년 '건국훈장 대한민국장'이 추서되었다.
검색정보	위키피디아 문서제목	검색 문장
1		

출처:

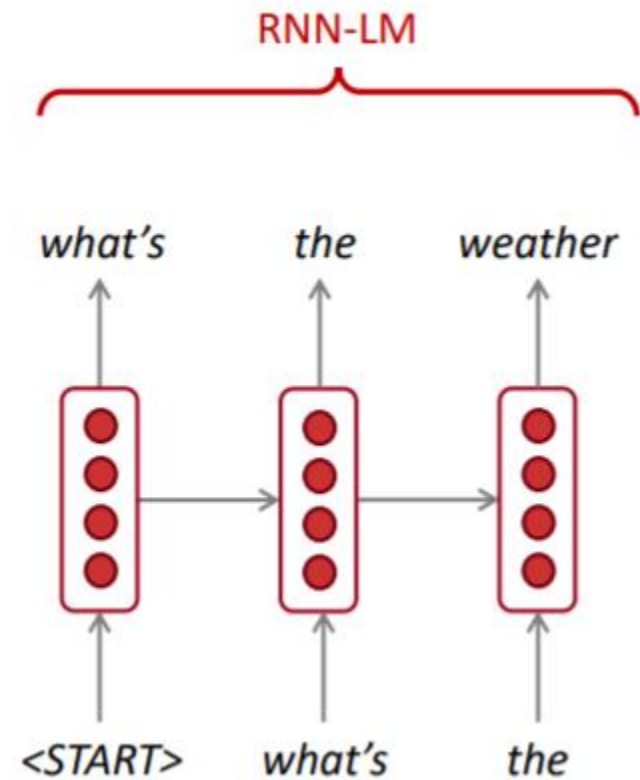
http://aiopen.etri.re.kr/guide_wikiQA.php#group05

Generate text with RNN


speech recognition, machine translation, summarization, etc



conditioning
.....>



STT(Speech to Text) example: 빅스비, Siri, ...

Google Cloud

Google을 선택해야 하는 이유 솔루션 제품 가격 책정 시작하기

AI 및 머신러닝 제품

Speech-to-Text

Google AI 기술로 지원되는 API를 사용하여 음성을 텍스트로 정확하게 변환할 수 있습니다.

[무료 체험하기](#)

- ✓ 정확한 캡션으로 콘텐츠 텍스트 변환
- ✓ 음성 명령어를 통해 제품에 더 나은 사용자 환경 제공
- ✓ 서비스 향상을 위해 고객 상호작용에 대한 유용한 정보 도출

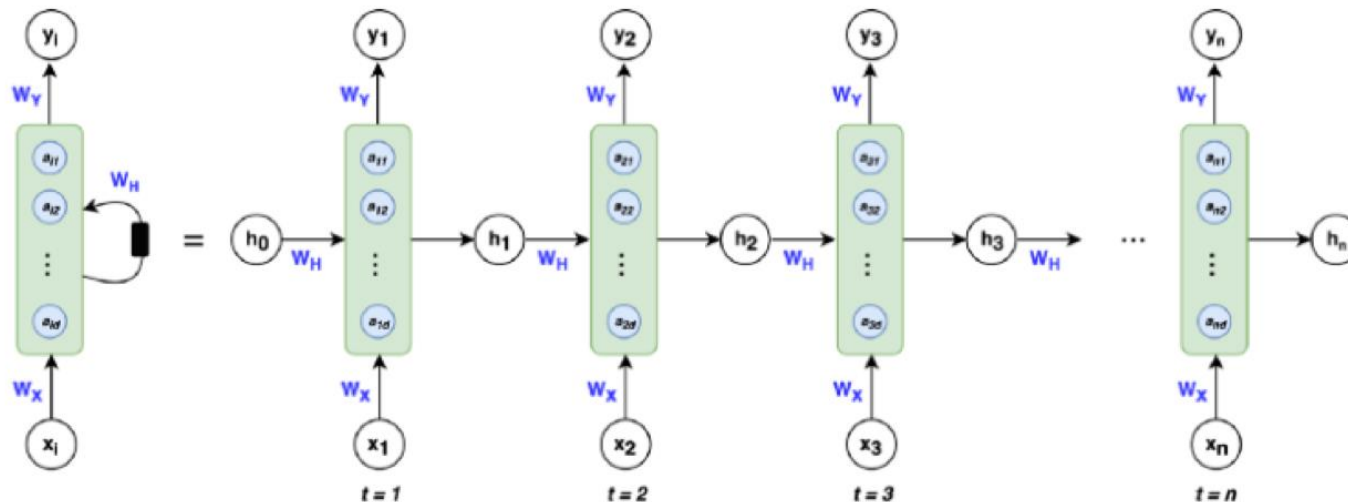
Speech-to-Text

- 이점
- 데모
- 주요 특징
- 고객
- 새로운 소식

문서

Understanding LSTM Networks

RNN Review



Left: Shorthand notation often used for RNNs, Right: Unfolded notation for RNNs

이론적으로는



다룰 수 있어야 하지만

실제로 RNN은 **Vanishing/exploding gradient problem** 으로 이를 학습하지 못한다.

Understanding LSTM Networks

RNN Review

Vanishing gradient problem

$$\frac{\partial J^{(i)}(\theta)}{\partial h^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial h^{(i)}} \prod_{j < t \leq i} \frac{\partial h^{(t)}}{\partial h^{(t-1)}}$$

Exploding gradient problem

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

Understanding LSTM Networks

Language task에서 vanishing gradient가 보여주는 문제

The writer of books _____

correct answer : 'is' (Syntactic recency)

wrong answer : 'are' (Sequential recency)

Vanishing gradient는 긴 문맥을 학습하지 못하기에 'are'과 같이 잘못된 품사를 뱉을 수 있다.


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 _____

마지막 단어를 예측하기 위해서는 첫 문장의 'tickets'에 대한 학습이 필요하다. 그렇지만 vanishing gradient문제가 발생하는 경우에는 예전 데이터를 학습하지 못하기 때문에 예측이 잘못될 확률이 높다.

Understanding LSTM Networks

Vanishing gradient problem


```
iteration 0: loss 1.156405
iteration 1000: loss 1.100737
iteration 2000: loss 0.999698
iteration 3000: loss 0.855495
iteration 4000: loss 0.819427
iteration 5000: loss 0.814825
iteration 6000: loss 0.810526
iteration 7000: loss 0.805943
iteration 8000: loss 0.800688
iteration 9000: loss 0.793976
iteration 10000: loss 0.783201
iteration 11000: loss 0.759909
iteration 12000: loss 0.719792
iteration 13000: loss 0.683194
iteration 14000: loss 0.655847
```



Sigmoid

Vanishing gradient에 취약

```
iteration 0: loss 1.116188
iteration 1000: loss 0.275047
iteration 2000: loss 0.152297
iteration 3000: loss 0.136370
iteration 4000: loss 0.130853
iteration 5000: loss 0.127878
iteration 6000: loss 0.125951
iteration 7000: loss 0.124599
iteration 8000: loss 0.123502
iteration 9000: loss 0.122594
iteration 10000: loss 0.121833
iteration 11000: loss 0.121202
iteration 12000: loss 0.120650
iteration 13000: loss 0.120165
iteration 14000: loss 0.119734
```



ReLU

Vanishing gradient에 강건

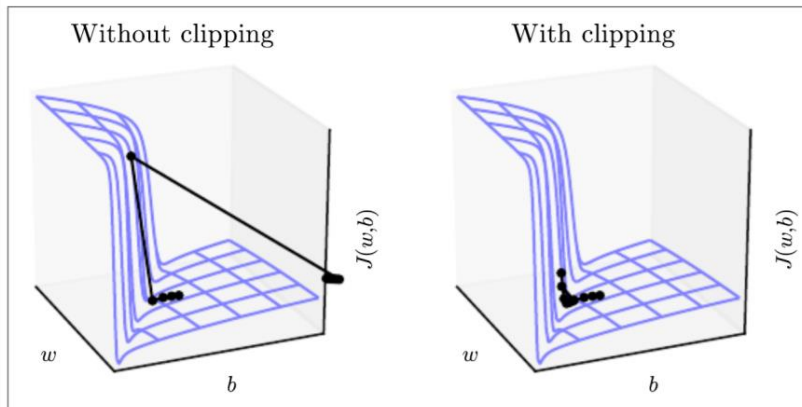
Understanding LSTM Networks

How to fix exploding / vanishing gradient problem?

Exploding problem

Gradient clipping

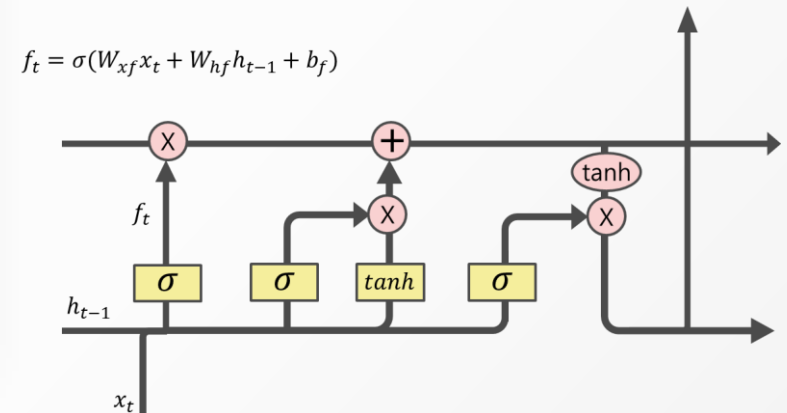
Gradient의 norm이 threshold를 넘어간다면 scaling down
(방향은 여전히 같다는 것이 중요)



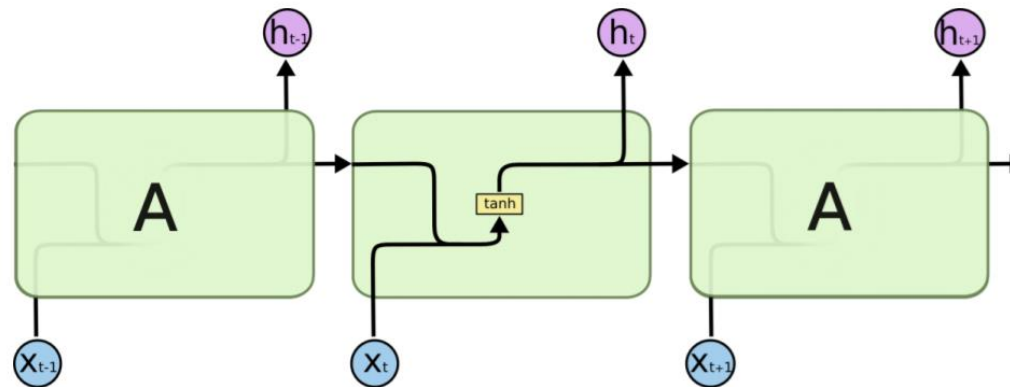
Vanishing problem

LSTM

Make a separate memory

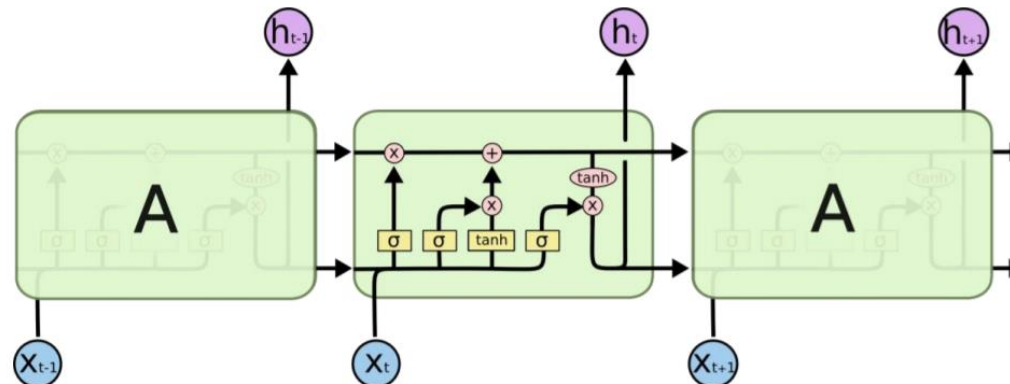


Understanding LSTM Networks



The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

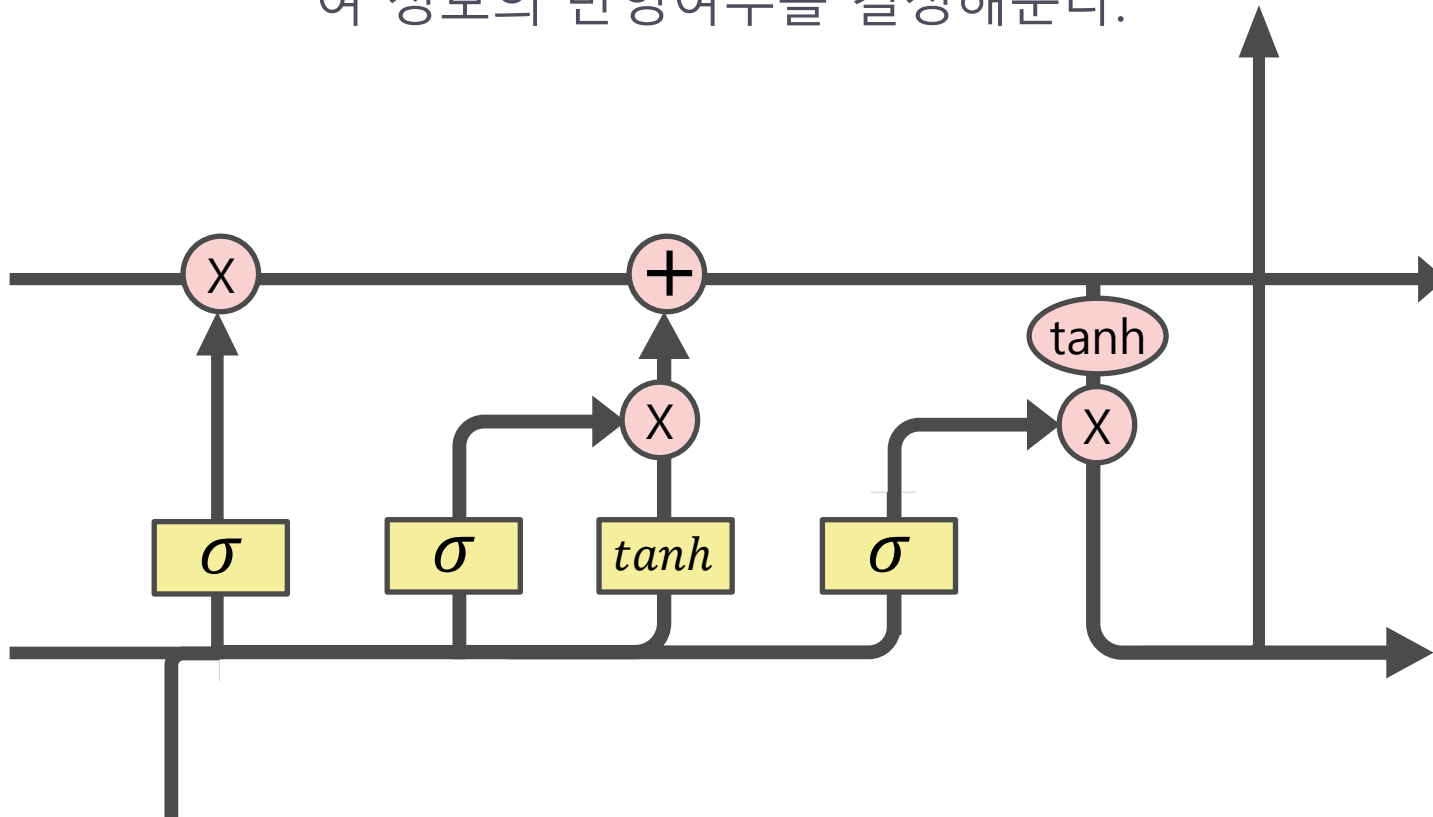


The repeating module in an LSTM contains four interacting layers.

Understanding LSTM Networks

Structure

Cell State(memory)가 forget, input, output 세 개의 게이트를 이용하여 정보의 반영여부를 결정해준다.

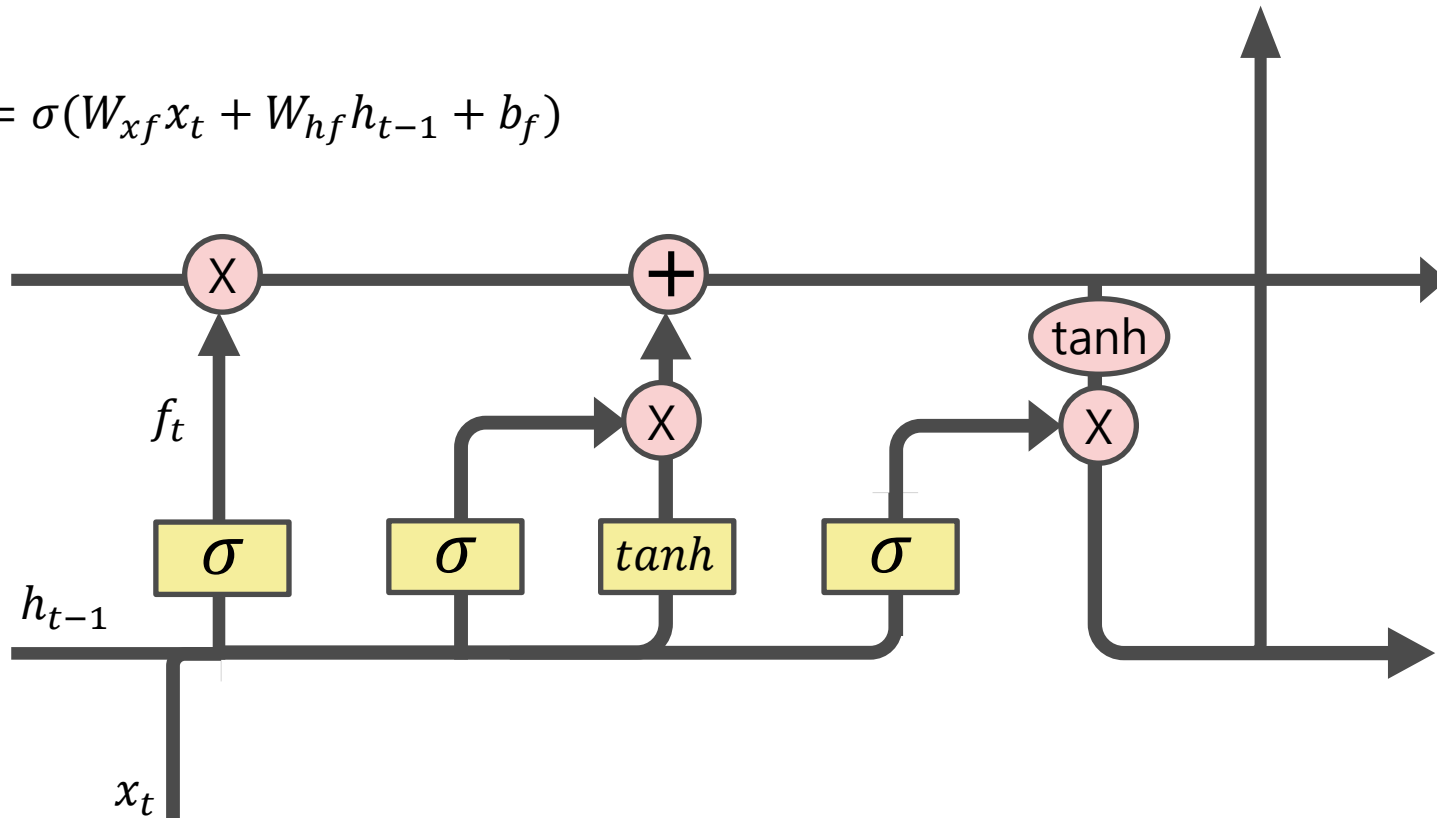


Understanding LSTM Networks

Forget gate layer

0인 경우엔 없애고 1인 경우엔 정보를 보존한다.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$



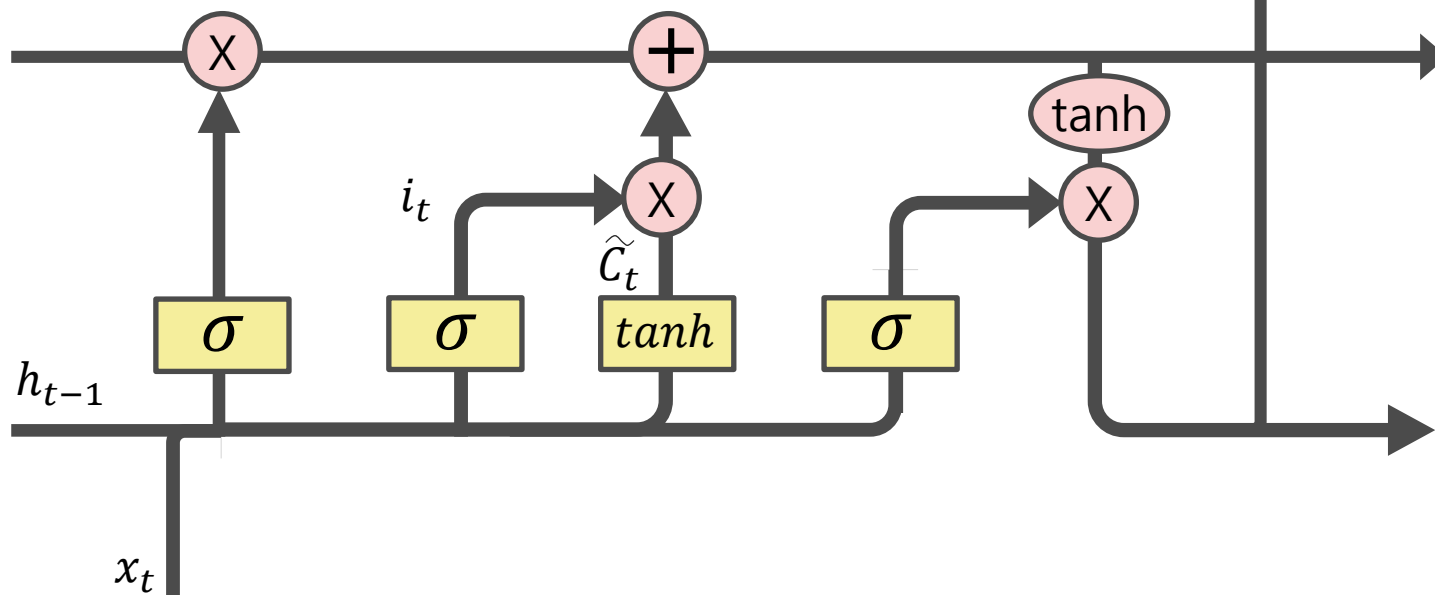
Understanding LSTM Networks

업데이트 하려는 변수를 결정한다.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$\tilde{C}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

tanh layer
A vector of candidate values를 만들어준다
(말그대로 후보군)

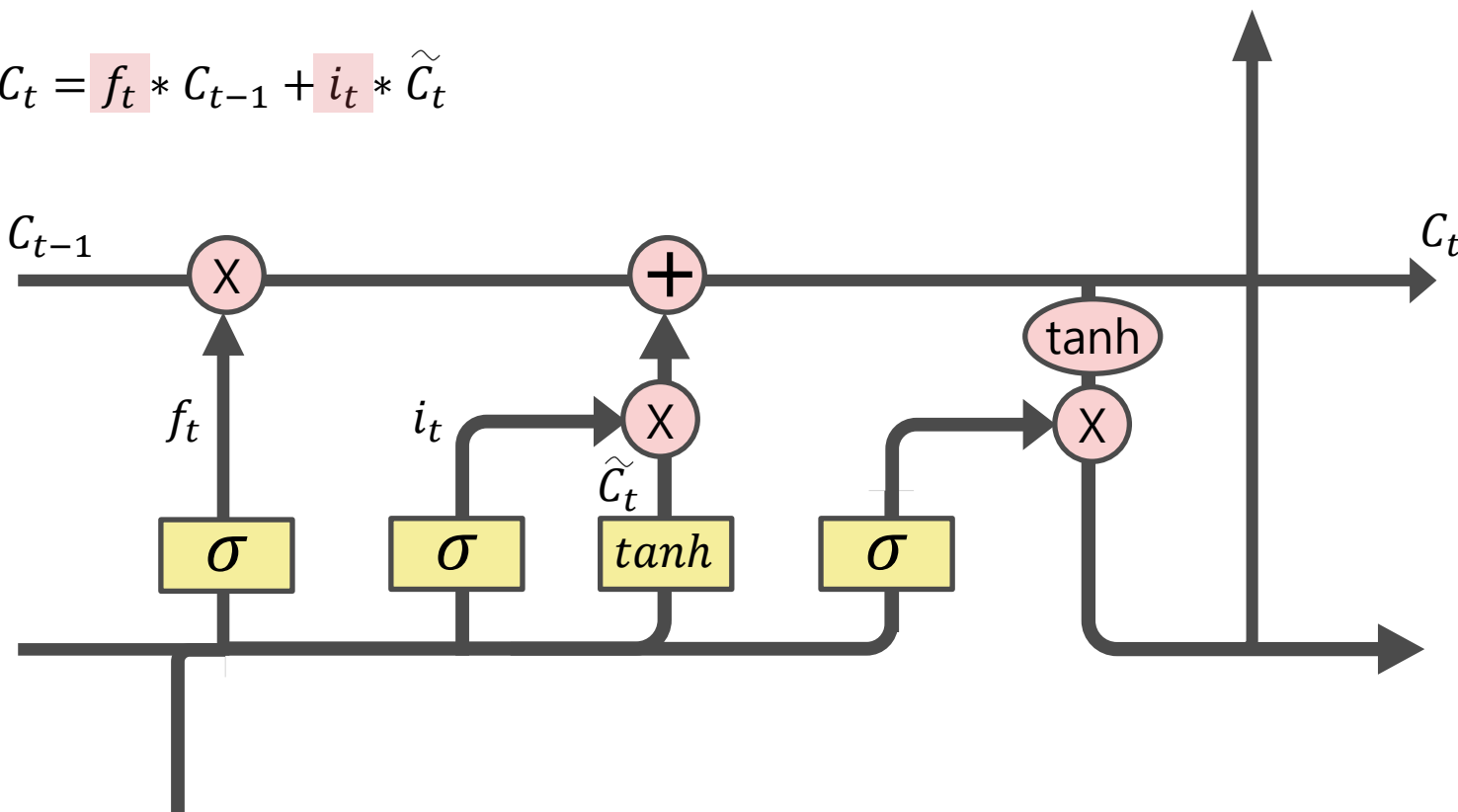


Understanding LSTM Networks

Updating cell state

Forget gate와 input gate에서 출력된 값들을 이용해 cell state를 업데이트 해준다.
실제 필요없는 정보의 drop과 필요한 새 정보의 추가가 이뤄지는 단계이다.

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



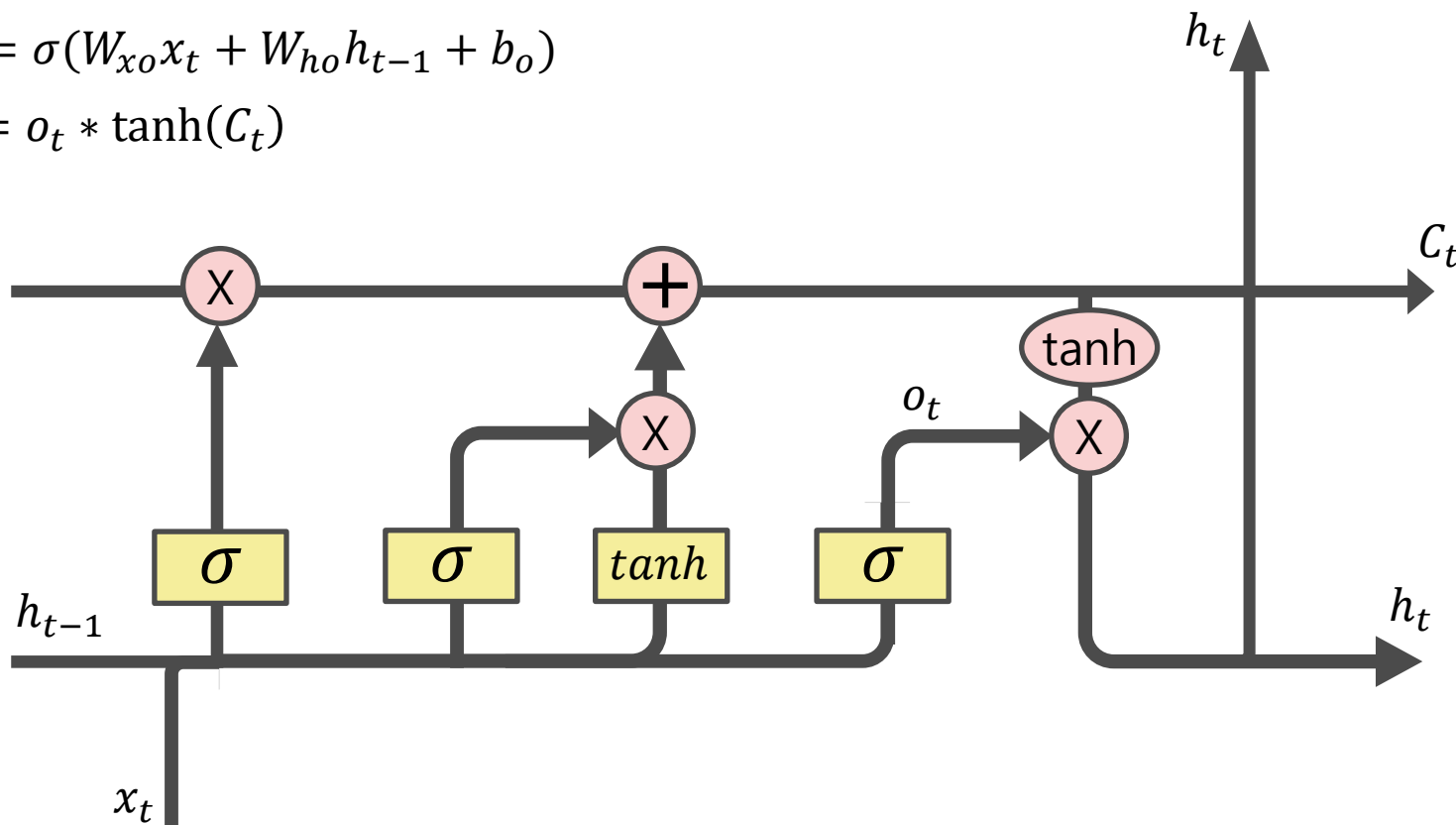
Understanding LSTM Networks

Output gate layer

출력값을 결정하는 단계이며 Cell state에 activation function을 씌운 값이다.
 o_t 를 통해 cell state를 얼마만큼 다음단계로 보낼지 계산한다.

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

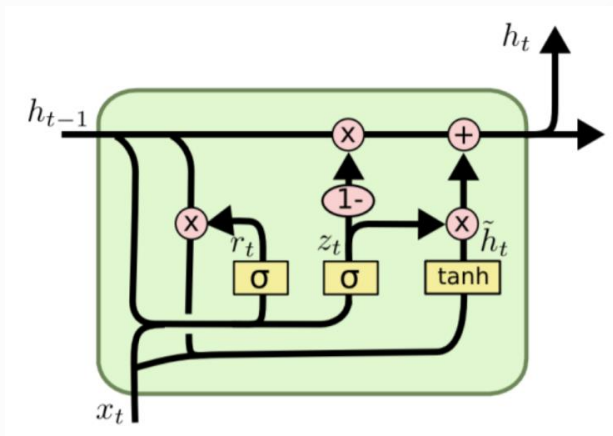
$$h_t = o_t * \tanh(C_t)$$



GRU Model

Structure

01



Reset gate(r) (과거 정보에 대한 reset 담당)
Update gate(z)

02

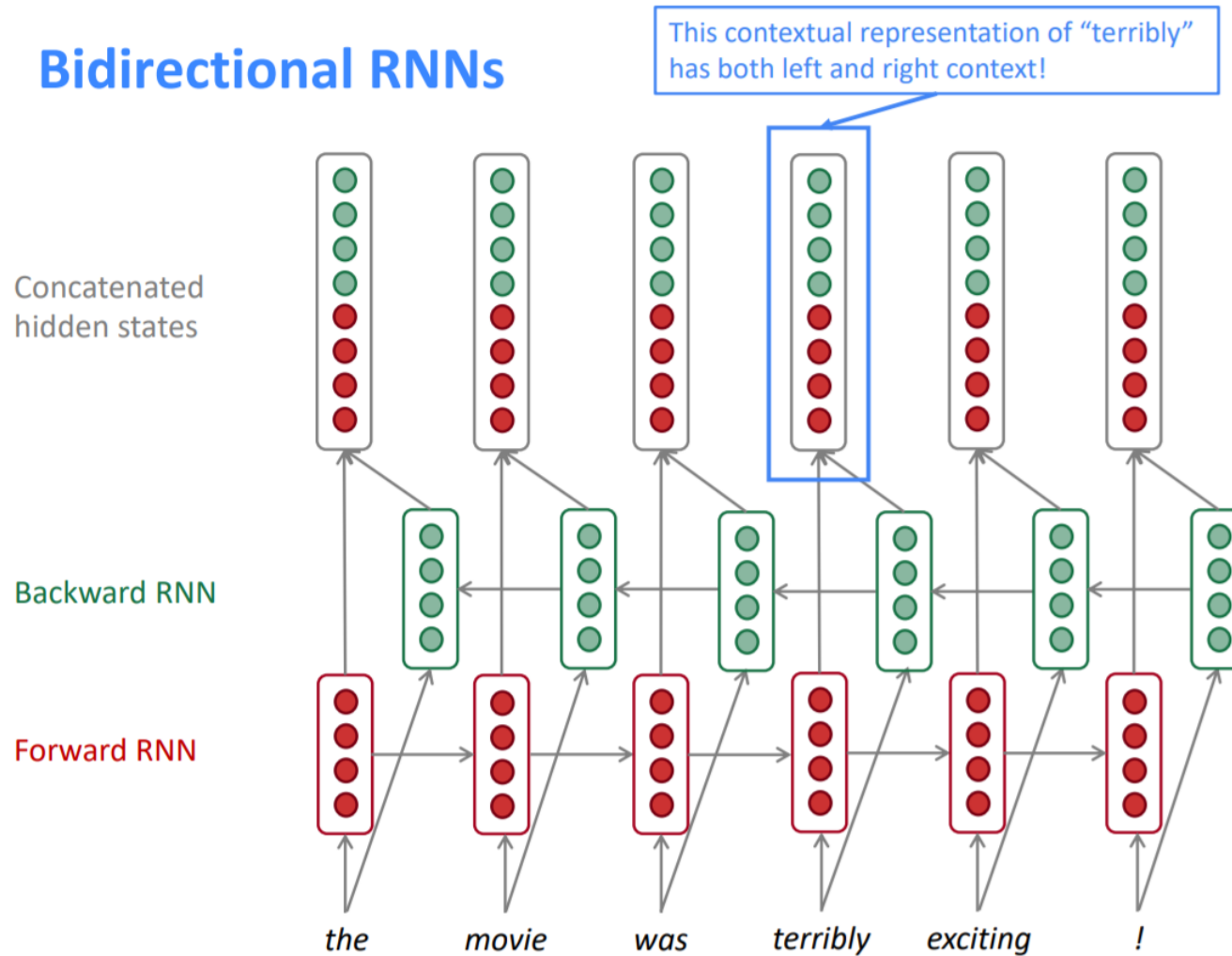
GRU

Forget gate와 input gate를 update gate(z)로 합침.
cell state와 hidden state를 합침.
그 외 몇가지 수정

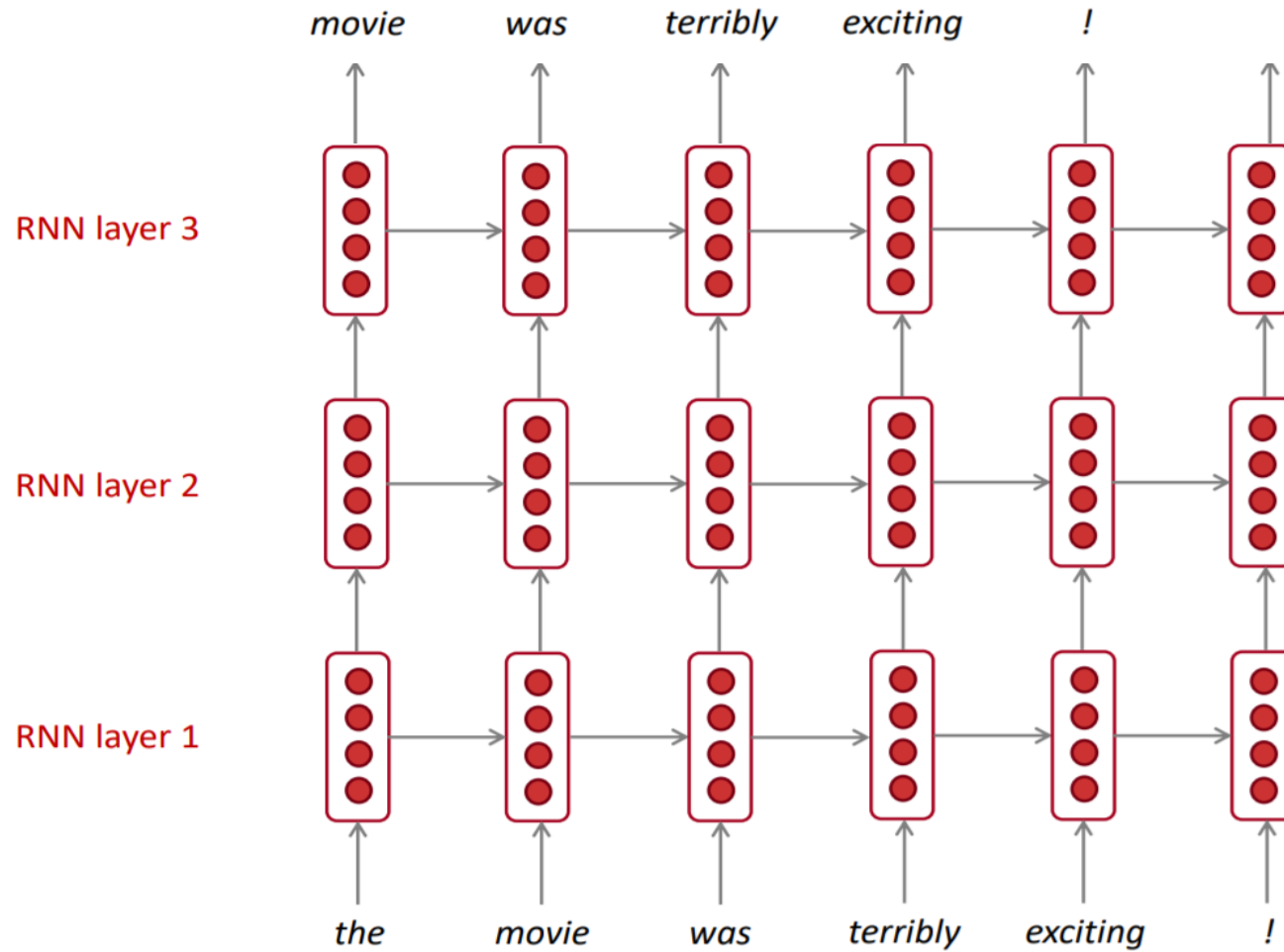
$$\begin{aligned} r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\ z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\ \tilde{h}_t &= \tanh(W_{xh}x_t + W_{hh}(r_t * h_{t-1}) + b_h) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned}$$

Bidirectional RNN

Bidirectional RNNs



Multi-layer RNN



각 모델의 장점

Bidirectional / Multi-layer RNN model

Bidirectional model

단어 예측에 있어서 해당 단어를 기준으로 좌/우 문맥을 모두 고려할 수 있다.

Ex) The movie was **terribly** exciting!
terribly는 우측의 **exciting**의 문맥 하에서만 제대로 된 의미를 가질 수 있다.



Multi-layer model

Higher RNN은 higher-level features를 계산할 수 있다.

(복잡한 태스크에 적용하면 좋다.)

실전에서는 RNN 모형보다 Multi-layer RNN 모형이 성능을 잘 내고 있다.



참조사이트

2020-1학기 Yonsei Univ Applied Statistics 박재우 교수님 딥러닝 강의자료 Recurrent Neural Networks

<https://aikorea.org/blog/rnn-tutorial-3/>

<https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/>

<https://excelsior-cjh.tistory.com/89>

<https://excelsior-cjh.tistory.com/185>

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

<https://arxiv.org/pdf/1412.3555.pdf>

<https://brightwon.tistory.com/10>

<https://curt-park.github.io/2017-04-03/why-is-lstm-strong-on-gradient-vanishing/>

감사합니다.