

텍스트 세미나

ToBig's 13기 김민정

Chapter 6

# Language Models and Recurrent Neural Networks

## Unit 00 | Overview

오늘 배울 것들 중 중요한 두 가지 키워드!

- **Language Modeling**  
: 문장이 주어졌을 때 지금까지의 문장 이후에 나올 단어를 예측하는 작업
- **Recurrent Neural Networks (RNNs)**  
: 해당 작업을 효과적으로 수행하기 위해 도입한 Neural Network의 일종

# Contents

---

**Unit 01 | Language Modeling**

---

**Unit 02 | N-gram Language Model**

---

**Unit 03 | Neural Language Model**

---

**Unit 04 | RNN Language Model**

---

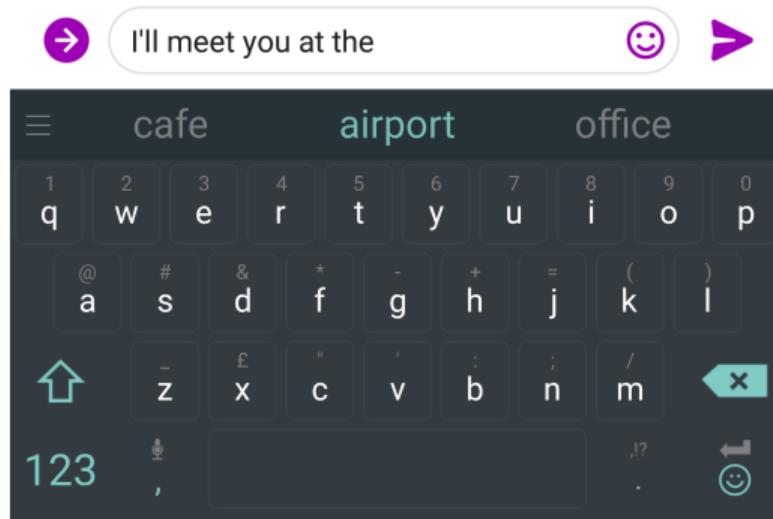
**Unit 05 | Perplexity**

---

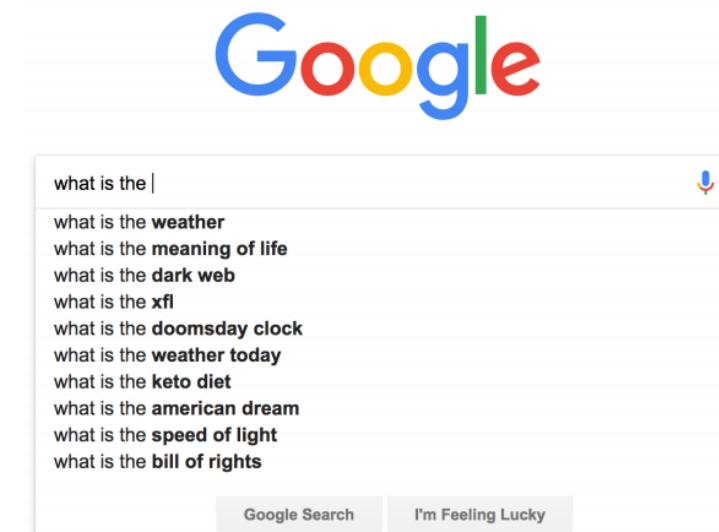
# Unit 01 | Language Modeling

We use Language Models every day!

자동완성



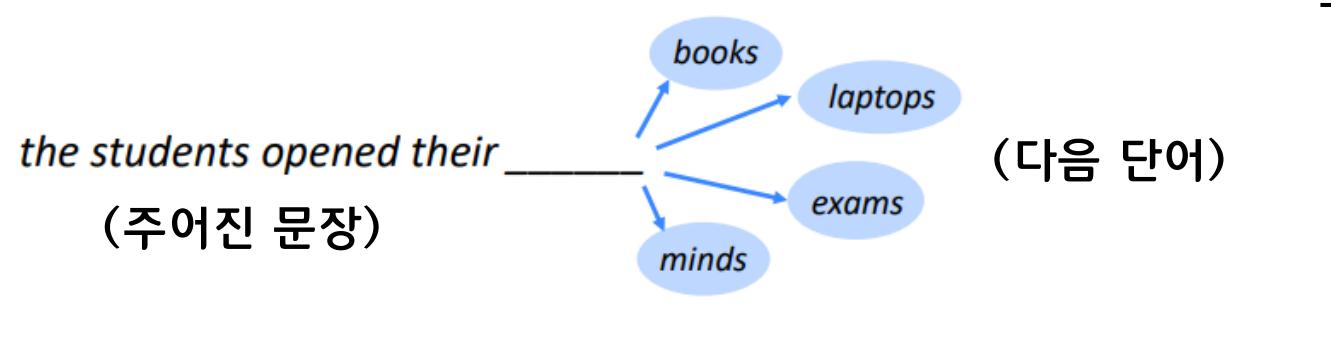
구글 검색



# Unit 01 | Language Modeling

## Language Modeling

: 현재까지 주어진 문장의 다음 단어를 예측하는 것



Language Model  
: 이러한 일을 수행하는 모델

# Unit 01 | Language Modeling

조금 더 formal하게 표현하자면!

## Language Modeling

: 현재까지 주어진 문장의 다음 단어를 예측하는 것

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

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

where  $x^{(t+1)}$  can be any word in the vocabulary  $V = \{w_1, \dots, w_{|V|}\}$

특정 문장에 확률을 할당할 수 있음

문장의 단어  $x^{(1)}, x^{(2)}, \dots, x^{(t)}$ 가 주어졌을 때,  
다음에 올 단어  $x^{(t+1)}$ 의 확률

# Unit 01 | Language Modeling

조금 더 formal하게 표현하자면!

## Language Modeling

: 현재까지 주어진 문장의 다음 단어를 예측하는 것

확률을 차례로 곱해 나감

$$P(x^{(1)}, \dots, x^{(T)}) = P(x^{(1)}) \times P(x^{(2)} | x^{(1)}) \times \dots \times P(x^{(T)} | x^{(T-1)}, \dots, x^{(1)})$$

piece of text

$$= \prod_{t=1}^T P(x^{(t)} | x^{(t-1)}, \dots, x^{(1)})$$



This is what our LM provides

특정 문장의 확률(좌변)을 식의 우변과 같이 연속된 조건부확률로 풀어 쓴 이후에,

LM을 통해 알아낼 수 있는 값들(두번째 줄)을 통해 확률을 계산할 수 있음

\* conditional probability :  $P(A|B) = \frac{P(A \cap B)}{P(B)} \rightarrow P(A \cap B) = P(B) \cdot P(A|B)$

"multiplication rule" :  $P(A_1 \cap \dots \cap A_n) = P(A_1) \cdot P(A_2 | A_1) \cdot P(A_3 | A_1 \cap A_2) \dots \cdot P(A_n | A_1 \cap \dots \cap A_{n-1})$

(ex)  $P(A \cap B \cap C) = P(A \cap B) \cdot P(C | A \cap B) = P(A) \cdot P(B | A) \cdot P(C | A \cap B)$

# Contents

---

Unit 01 | Language Modeling

---

Unit 02 | N-gram Language Model

---

Unit 03 | Neural Language Model

---

Unit 04 | RNN Language Model

---

Unit 05 | Perplexity

---

## Unit 02 | N-gram Language Model

### N-gram Language Model

Question) Language Model은 어떻게 학습하는 거야?

Answer) Deep Learning을 도입하기 전에 주로 사용된 방법은 N-gram Language Model !

#### Definition of N-gram

: 연속된 N개의 단어 덩어리

ex)

- 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" 4개의 연속된 단어 -> 4-gram

## Unit 02 | N-gram Language Model

### N-gram Language Model

Question) 그러면 다음에 어떤 단어가 올 확률은 어떻게 구하는데?

Answer) large corpus에서 counting을 할거야!

근데 그 전에, **가정** 하나만 하겠습니다!

가정) “다음 단어는 오직 직전의  $n-1$ 개의 단어에만 영향을 받는다”

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

즉,  $n-2$ 번째나 그 이전의 단어는 다음에 오게 될 단어와 전혀 상관이 없다고 생각

## Unit 02 | N-gram Language Model

### N-gram Language Model

아까 정의한 “문장에 확률을 할당할 수 있다”라는 개념을 통해

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

N-gram 문장이 나타날 확률과, (N-1)-gram이 나타날 확률을 이용하면,  
현재 문장이 주어졌을 때 다음 단어가 올 확률을 계산할 수 있음

이때, 계산에 필요한 N-gram과 (N-1)-gram의 확률은 큰 corpus에서 출현 빈도를 세서 얻을 수 있음

# Unit 02 | N-gram Language Model

## N-gram Language Model

ex)

Suppose we are learning a **4-gram** Language Model.

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

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

For example, suppose that in the corpus:

- “students opened their” occurred 1000 times
- “students opened their books” occurred 400 times
  - $P(\text{books} | \text{students opened their}) = 0.4$
- “students opened their exams” occurred 100 times
  - $P(\text{exams} | \text{students opened their}) = 0.1$

Should we have  
discarded the  
“proctor” context?

- 우리는 4-gram model을 사용하므로 가정에 따라 ‘students’ 이전의 단어들은 전혀 고려하지 않음
- ‘students opened their’ 다음에 단어 w가 나올 확률은 4-gram을 3-gram으로 나눈 값
- ‘books’: 40% / ‘exams’: 10%

# Unit 02 | N-gram Language Model

## N-gram Language Model

ex)

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the students opened their~~  
discard  
그럼 N을 늘리면 안되려나?  
condition on this

Nooooo..

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

For example, suppose that in the corpus:

- “students opened their” occurred 1000 times
- “students opened their books” occurred 400 times
  - $P(\text{books} | \text{students opened their}) = 0.4$
- “students opened their exams” occurred 100 times
  - $P(\text{exams} | \text{students opened their}) = 0.1$

Should we have  
discarded the  
“proctor” context?

- 우리는 4-gram model을 사용하므로 가정에 따라  
‘students’ 이전의 단어들은 전혀 고려하지 않음

- ‘students opened their’ 다음에 단어  $w$ 가 나올  
확률은 4-gram을 3-gram으로 나눈 값

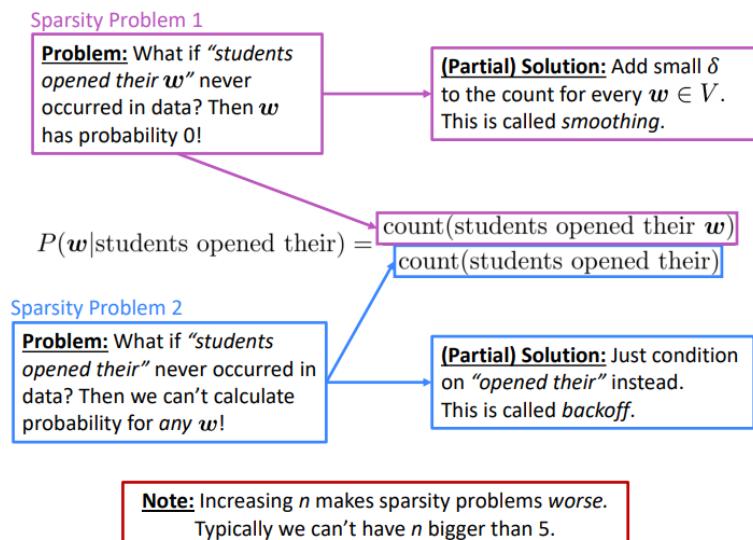
- ‘books’: 40% / ‘exams’: 10%

# Unit 02 | N-gram Language Model

## N-gram Language Model의 문제점

### 1) Sparsity problems

특정 N-gram의 corpus 내 출현 빈도가 낮을 때 발생  
일반적으로  $N < 5$ 로 설정한다고 함



한번도 등장하지 않은 단어에 확률 '0'을 줘버림

#### Problem 1)

- 분자 = 0
- "students opened their W" 등장 x
- => smoothing: everything has at least small prob!

#### Problem 2)

- 분모 = 0
- "students opened their" 등장 x
- => backoff: N-gram 대신 (N-1)-gram이라도 보자!

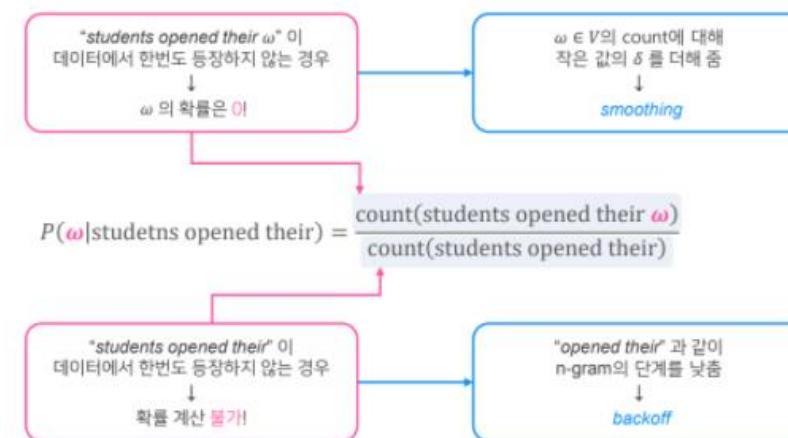
# Unit 02 | N-gram Language Model

<https://heiwa25.github.io/nlp/2019/10/06/Language-model-2/>

질문!

n-gram LM에서 sparsity problem을 해결하기 할 때 분모에도 문자처럼 작은 수를 더해주지 않고 왜 backoff 방식을 사용하는건가요? backoff를 사용하면 의도하는 것과 다르게 계산되지 않나요 ㅠ

답변



분모가 0인 경우가 더 심각한 경우임

## Unit 02 | N-gram Language Model

### N-gram Language Model의 문제점

#### 2) Storage problems

N-gram의 count 정보를 저장하기 위해 model의 크기가 지나치게 커지는 문제

**Storage:** Need to store count for all  $n$ -grams you saw in the corpus.

Increasing  $n$  or increasing corpus increases model size!

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

## Unit 02 | N-gram Language Model

### N-gram Language Model의 문제점

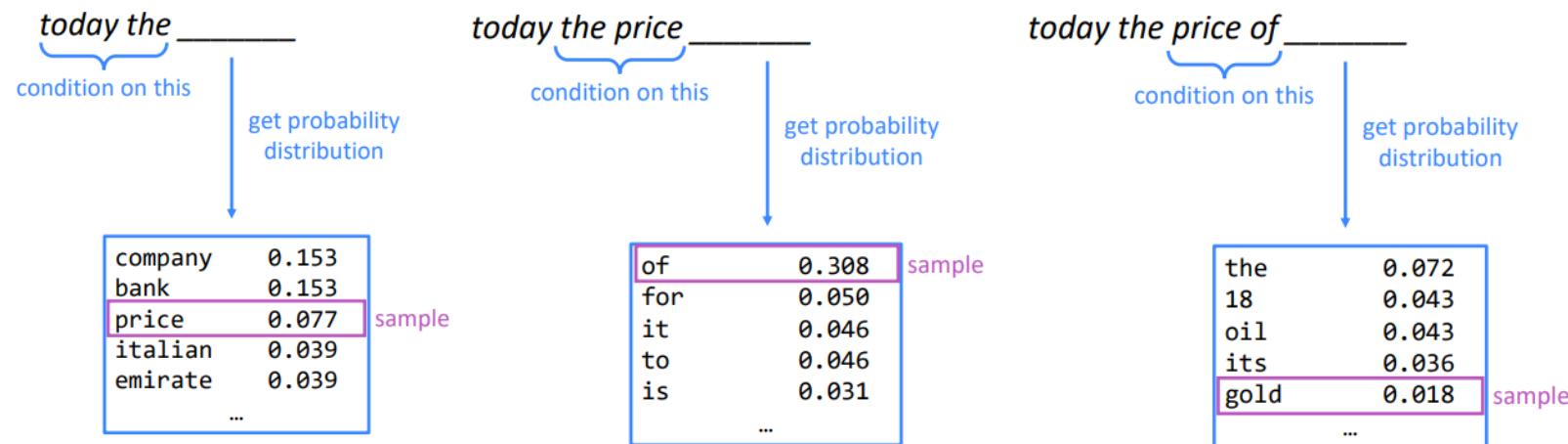
#### 3) Incoherence problems

N-gram model이 문맥을 충분히 반영하지 못하는 문제

- 앞서 했던 “다음 단어는 앞의  $n-1$ 개 단어에만 영향을 받는다”라는 가정 때문에 이전 문맥을 충분히 반영하지 못함
- 즉, 다음 단어를 예측하는데 아주 중요한 정보가 있다고 해도 그 정보를 놓치는 경우가 생김
- **N의 크기를 늘리면 이러한 문제를 어느정도 해결할 수 있지만, 동시에 sparsity problem이 심해짐**

# Unit 02 | N-gram Language Model

## Generating text with a N-gram Language Model



N-gram Language Model로 generation 할 수 있음!

## Unit 02 | N-gram Language Model

### Generating text with a N-gram Language model

#### 3-gram으로 얻은 text

*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 .*

- 쉼표(,)도 단어로 취급
- 놀랍게도 나름 문법적임
- 그러나 **incoherent...**

그러면 N의 크기를 늘려야 겠네!  
-> 그럼 또 sparsity 문제와 함께 모델의 크기가 커져버리는 문제 발생

## Unit 02 | N-gram Language Model

### Generating text with a N-gram Language model

이러한 N-gram model의 고질적인 문제를 해결하기 위해 도입한 것이

3-gram으로 얻은 text

*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 .*

Neural Network

- 쉼표(,)도 단어로 취급
- 놀랍게도 나름 문법적임
- 그러나 **incoherent...**

그러면 N의 크기를 늘려야 겠네!  
-> 그럼 또 sparsity 문제와 함께 모델의 크기가 커져버리는 문제 발생

# Contents

---

Unit 01 | Language Modeling

---

Unit 02 | N-gram Language Model

---

Unit 03 | Neural Language Model

---

Unit 04 | RNN Language Model

---

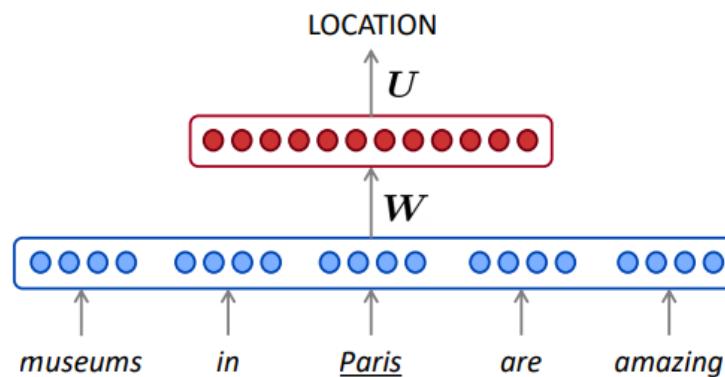
Unit 05 | Perplexity

---

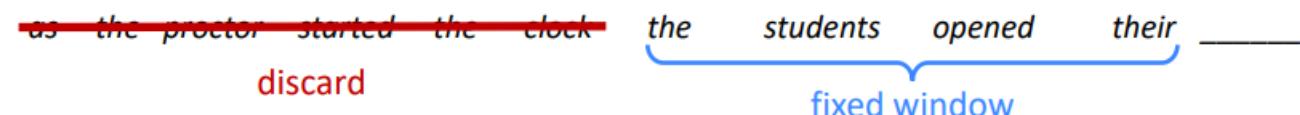
# Unit 03 | Neural Language Model

## Neural Language Model

Neural Network를 이용한 가장 간단한 Neural Language model



Lecture 3에서 NER에 적용했던  
Window-based neural network는  
Center를 기준으로 앞뒤의 window를 정했다면



Lecture 6에서는 예측할 단어의  
이전에 window를 고정

# Unit 03 | Neural Language Model

## A fixed-window Neural Language Model

output distribution

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

hidden layer

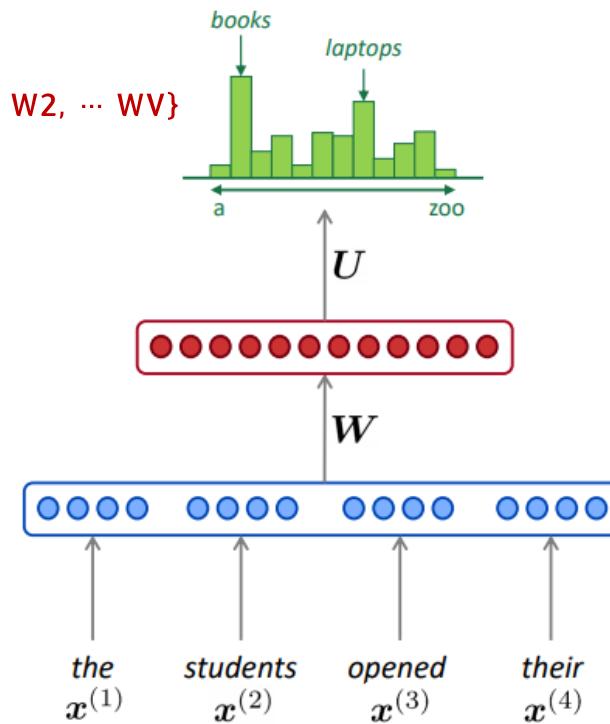
$$h = f(We + b_1)$$

concatenated word embeddings

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

words / one-hot vectors

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



2) Neural Network에 넣음

1) 특정 개수의 단어를 입력으로 받은 후 임베딩

# Unit 03 | Neural Language Model

## A fixed-window Neural Language Model

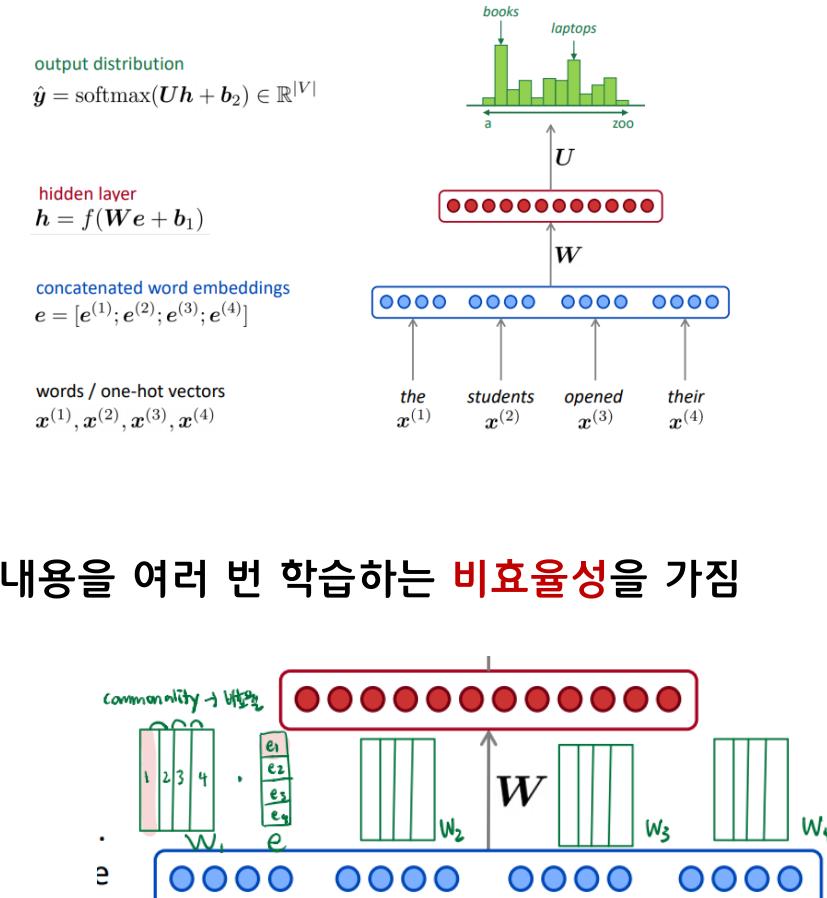
### 장점

- sparsity 문제가 사라짐 (Softmax 덕분에)
- Don't need to store all observed

### 단점

- Window size(관찰하는 단어의 개수)가 작음 (문맥을 반영하지 못함)  
또한, window size를 늘리더라도 중요한 context를 담지 못할 수도 있음
- 단어의 위치에 따라 곱해지는 가중치가 다르기 때문에 neural model이 비슷한 내용을 여러 번 학습하는 **비효율성**을 가짐

즉, input의 길이가 어떻더라도 처리할 수 있는 neural architecture가 필요!



# Contents

---

Unit 01 | Language Modeling

---

Unit 02 | N-gram Language Model

---

Unit 03 | Neural Language Model

---

Unit 04 | RNN Language Model

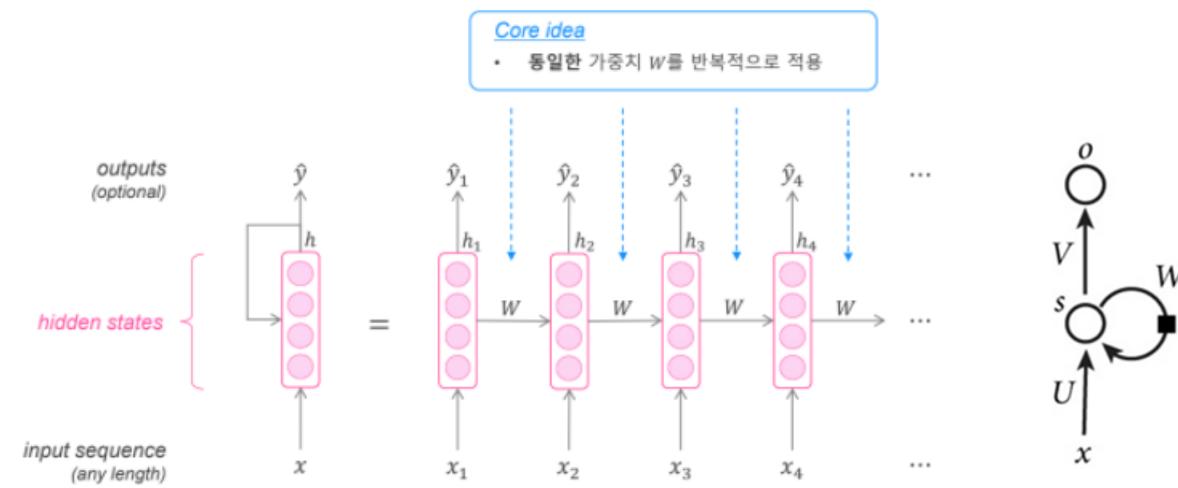
---

Unit 05 | Perplexity

---

# Unit 04 | RNN Language Model

## Core idea



동일한 가중치를 반복적으로 적용하자!

# Unit 04 | RNN Language Model

## 4-1. RNN (Recurrent Neural Network)

기본적인 아이디어: 순차적인 정보를 처리하는 것

This is a very long sentence explaining about a long sentence.

rnn은 다 본다!      trigram bigram Target

기존의 신경망 구조 -> 모든 입력이 각각 독립적이라고 가정

RNN -> 이전의 계산 결과에 영향을 받음

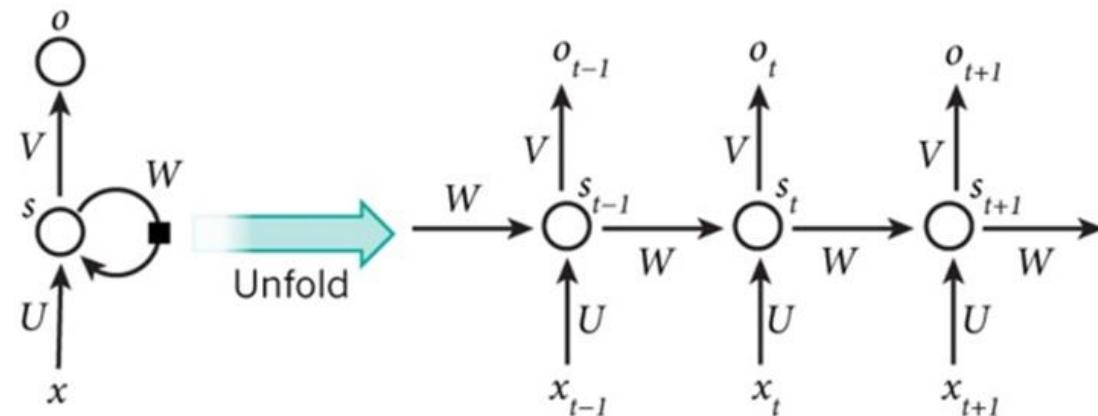
(현재까지 계산된 결과에 대한 메모리 정보를 갖고 있다고 볼 수도 있음)

RNN은 임의의 길이의 sequence 정보를 처리할 수 있다고 하지만, 실제로는 비교적 짧은 sequence만 효과적으로 처리할 수 있다고 함

# Unit 04 | RNN Language Model

## 4-1. RNN (Recurrent Neural Network)

output: 추측된 단어들의 sequence

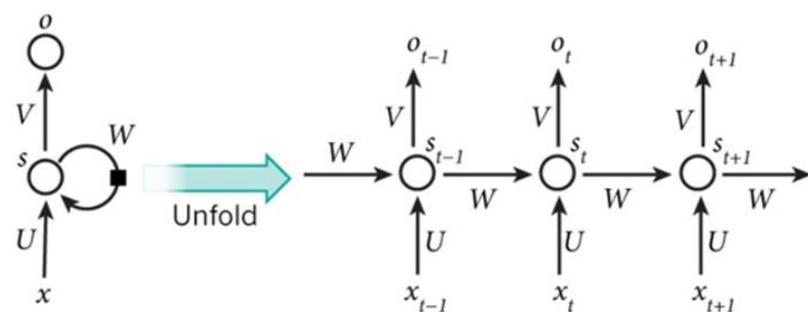


input: 단어들의 sequence

네트워크를 학습할 때에는 시간 스텝에서의  
출력 값이 실제로 다음 입력 단어가 되도록  
 $o_t = x_{t+1}$ 로 정해줌

# Unit 04 | RNN Language Model

## 4-1. RNN (Recurrent Neural Network)



$x_t$ : 시간 스텝(time step)  $t$ 에서의 입력값

$s_t$ : 시간 스텝  $t$ 에서의 hidden state

$$S_t = f(U_{x_t} + W_{s_{t-1}})$$

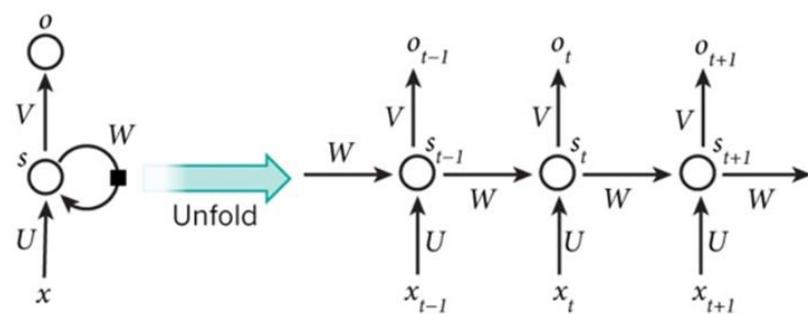
비선형 함수  $f$ 는 보통 tanh이나 ReLU가 사용됨

첫 hidden state를 계산하기 위한 hidden state는 보통 0으로 초기화 시킨다고 함

$o_t$ : 시간 스텝  $t$ 에서의 출력값

# Unit 04 | RNN Language Model

## 4-1. RNN (Recurrent Neural Network)



- $s_t$  는 네트워크의 메모리라고 생각할 수 있음
- RNN은 학습해야 하는 파라미터 수를 많이 줄여 줌
- 매 시간 스텝마다 출력값을 낼 필요는 없음
- 매 시간 스텝마다 입력값을 낼 필요도 없음

# Unit 04 | RNN Language Model

강의 자료에 있던 그림으로 한번 더 살펴봅시다!

## 4-1. RNN (Recurrent Neural Network)

### A RNN Language Model

output distribution

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

hidden states

$$\mathbf{h}^{(t)} = \sigma(W_h \mathbf{h}^{(t-1)} + W_e e^{(t)} + \mathbf{b}_1)$$

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

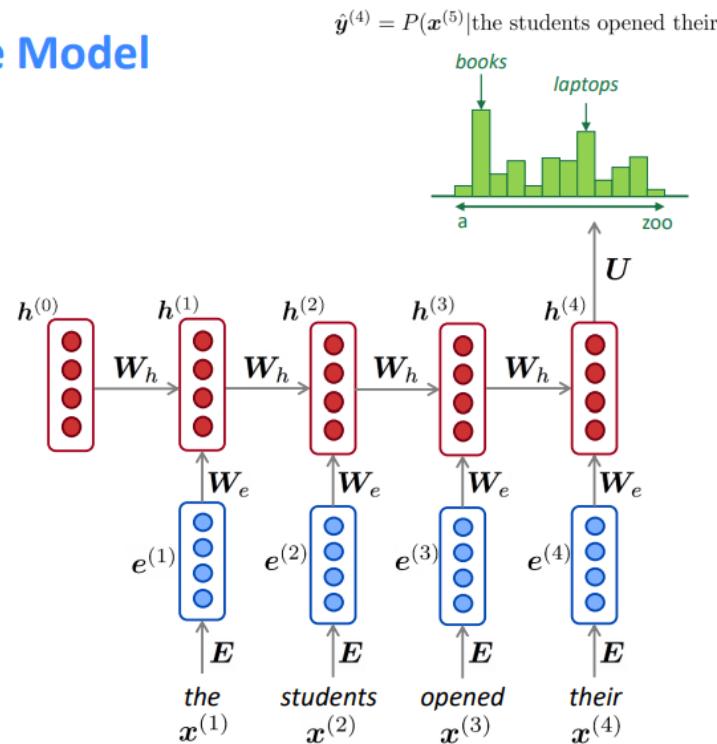
word embeddings

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

words / one-hot vectors

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

*Note:* this input sequence could be much longer, but this slide doesn't have space!



## Unit 04 | RNN Language Model

### 질문!

강의 질문 중에서 embedding값을 모델에서 학습하는 건 선택?이라고 했는데,  
대부분 어떤 방법을 사용하나요? pre-trained? fine-tuning? 아님 random 값에서 파라미터로 학습?

### 답변

지난 텍스트 세미나 때 비슷한 질문이 있어서 가져와봤습니다.

임베딩값이 모델에서 학습하는 값은 아닌거죠? 초기 intial 값을 사용하고, weight만 학습하는 것인가?

선택의 문제입니다라고 합니다. word2vec이나 glove와 같이 pretrained된 아이들을 활용할 수 있고 거기서 또 fine-tuning(미세조정)을 하면서 학습시킬 수도 있고 아니면 아예 랜덤값을 부여해서 처음부터 학습시킬 수도 있다고 ...

## Unit 04 | RNN Language Model

### 4-1. RNN (Recurrent Neural Network)

#### 장점

- Input length에 상관없이 다음 단어를 예측할 수 있음 (이론상)
- 먼 곳에 위치한 단어도 고려할 수 있어 context를 반영할 수 있음
- Input이 길어져도 model size가 증가하지 않음

#### 단점

- 다음 단계로 진행하기 위해서는 이전 단계의 계산이 완료되어야 하므로 계산이 병렬적으로 진행되지 않아 느림
- 이론적으로는 먼 곳의 단어를 반영할 수 있지만 실제로는 vanishing gradient problem등의 문제가 있어 context가 반영되지 않는 경우도 있다는 점

## Unit 04 | RNN Language Model

### 4-2. Training RNN

- 1) 각 단어를 RNN model에 input으로 주고, 모든 단계에서 예상되는 다음 단어를 계산
- 2) 모든 단계에서 예상되는 다음 단어와 실제 다음 단어의 차이의 cross-entropy를 통해 loss를 구함

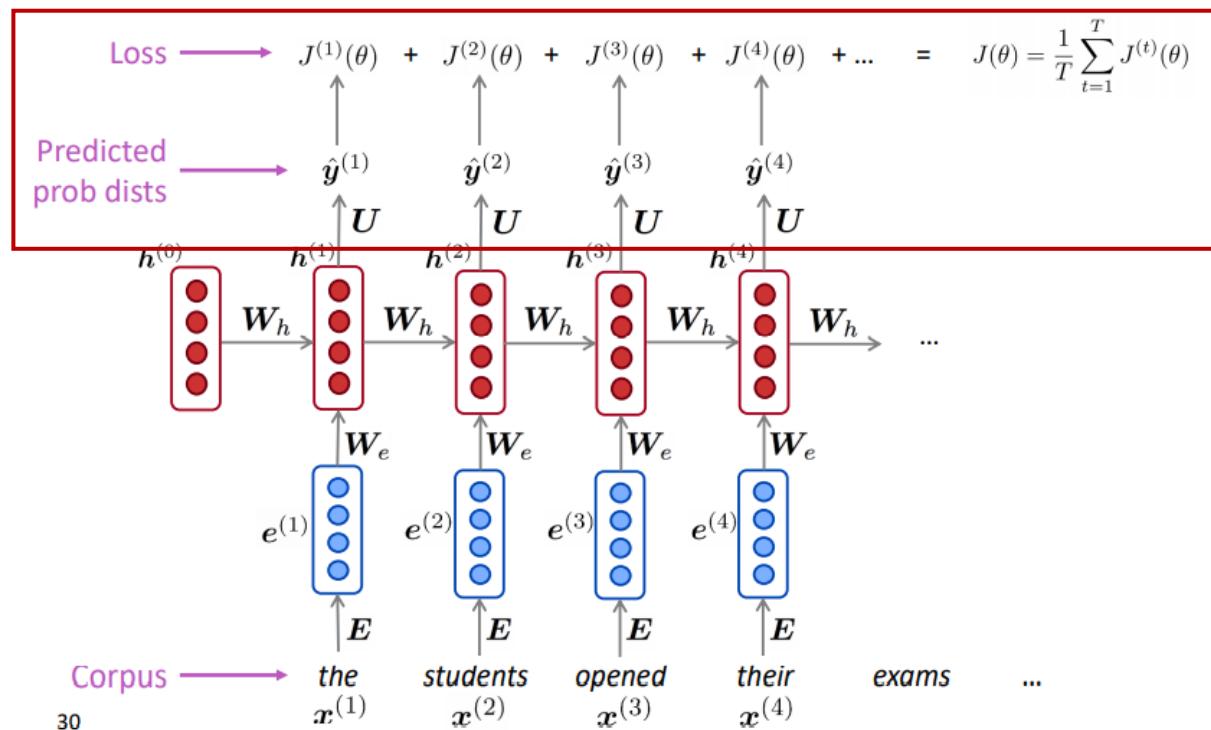
$$J^{(t)}(\theta) = CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = - \sum_{w \in V} \mathbf{y}_w^{(t)} \log \hat{\mathbf{y}}_w^{(t)} = -\log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

- 3) 모든 단계에서의 loss의 평균을 통해 전체 loss를 구함

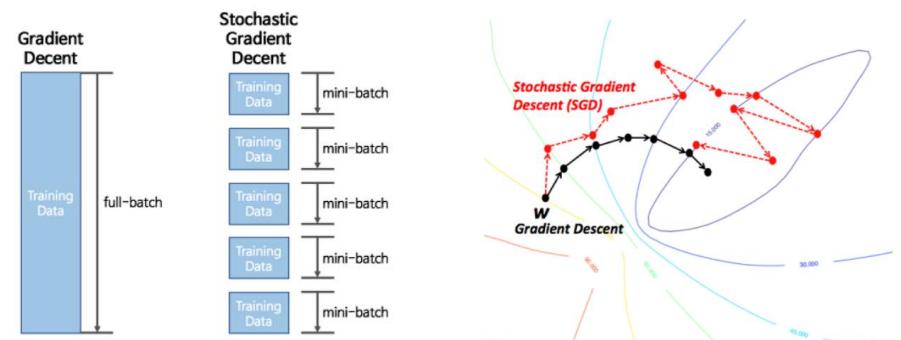
$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^T -\log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

# Unit 04 | RNN Language Model

## 4-2. Training RNN



- 다만, 실제 RNN-model을 학습할 때 이와 같이 학습하면 지나치게 많은 계산이 필요하므로, 실제로는 단어 단위가 아니라 문장 혹은 문서 단위로 입력을 줌
- 또, Stochastic Gradient Descent를 통해 optimize하는 것도 좋은 방법



## Unit 04 | RNN Language Model

질문!

보통  $x(1), \dots, x(t)$ 에 단어 하나가 들어가서  $x(t+1)$ 을 단어로 예측하는데,  
 $x$  가 sentence 나 document가 되면 output이 어떻게 나오는지 궁금해요!

답변

Task를 “분류” 라고 가정을 해보겠습니다!

제가 이해한 바로는

$\overset{\uparrow}{E}$        $\overset{\uparrow}{E}$        $\overset{\uparrow}{E}$        $\overset{\uparrow}{E}$        $\overset{\uparrow}{E}$   
the      students      opened      their      exams      ...  
 $x^{(1)}$        $x^{(2)}$        $x^{(3)}$        $x^{(4)}$

문장

단어

문서

문장

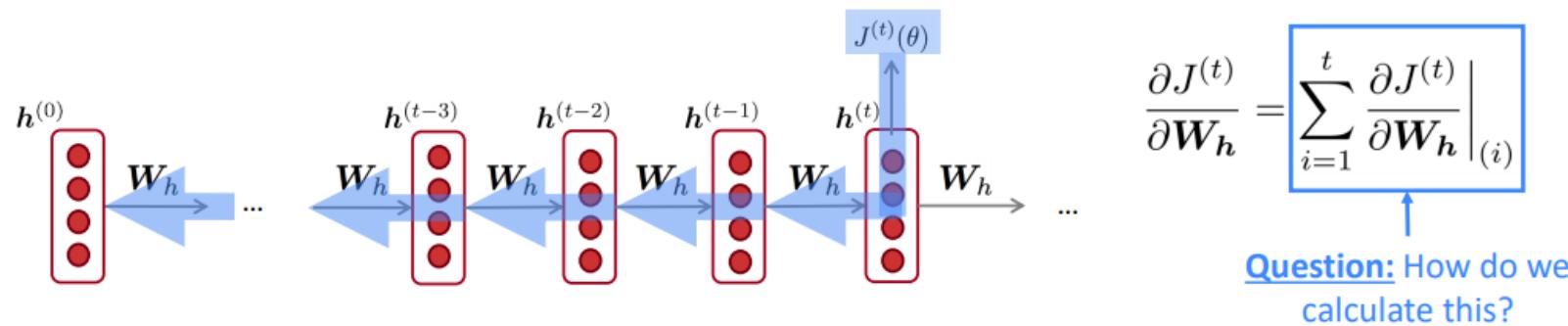
책

문서

# Unit 04 | RNN Language Model

## 4-3. Backpropagation for RNNs

- RNN 네트워크를 학습하는 것은 기존의 신경망 모델을 학습하는 것과 매우 유사
- 단, 기존의 backpropagation을 그대로 사용하진 못하고 **BPTT(Backpropagation Through Time)**라고 하는 약간 변형된 알고리즘을 사용
- 그러나 vanishing/exploding gradient 문제로 단순한 RNN을 BPTT로 학습시키는 것은 어려움



**Answer:** Backpropagate over timesteps  $i=t, \dots, 0$ , summing gradients as you go.  
This algorithm is called "backpropagation through time"

## Unit 04 | RNN Language Model

### 4-4. Generating text with a RNN

#### 1) Obama speeches

오바마 대통령 연설 생성



*The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.*

<https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0>

#### 2) Harry potter

해리포터 소설 생성



“Sorry,” Harry shouted, panicking—“I’ll leave those brooms in London, are they?”

“No idea,” said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry’s shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn’t felt it seemed. He reached the teams too.

<https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6>

# Unit 04 | RNN Language Model

## 4-4. Generating text with a RNN

### 3) recipes

#### 레시피 생성



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.

<https://gist.github.com/nylki/1efbaa36635956d35bcc>

### 4) Paint color names

#### 페인트 색 이름 생성

Ghasty Pink	231	137	165
Power Gray	151	124	112
Navel Tan	199	173	140
Bock Coe White	221	215	236
Horble Gray	178	181	196
Homestar Brown	133	104	85
Snader Brown	144	106	74
Golder Craam	237	217	177
Hurky White	232	223	215
Burf Pink	223	173	179
Rose Hork	230	215	198

Sand Dan	201	172	143
Grade Bat	48	94	83
Light Of Blast	175	150	147
Grass Bat	176	99	108
Sindis Poop	204	205	194
Dope	219	209	179
Testing	156	101	106
Stoner Blue	152	165	159
Burble Simp	226	181	132
Stanky Bean	197	162	171
Turdly	190	164	116

<https://aiweirdness.com/post/160776374467/new-paint-colors-invented-by-neural-network>

# Contents

---

Unit 01 | Language Modeling

---

Unit 02 | N-gram Language Model

---

Unit 03 | Neural Language Model

---

Unit 04 | RNN Language Model

---

Unit 05 | Perplexity

---

# Unit 05 | Perplexity

## Evaluating Language Models: Perplexity

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

Normalized by number of words

Inverse probability of corpus, according to Language Model

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

LM을 통해 예측한 corpus의 invers를  
corpus 길이로 normalize 해준 값

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

1.  $\log$   $\rightarrow$  exp  
2.  $\exp$   $\rightarrow$  exp  
3.  $\frac{1}{T} \sum$   $\rightarrow$   $J(\theta)$

*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

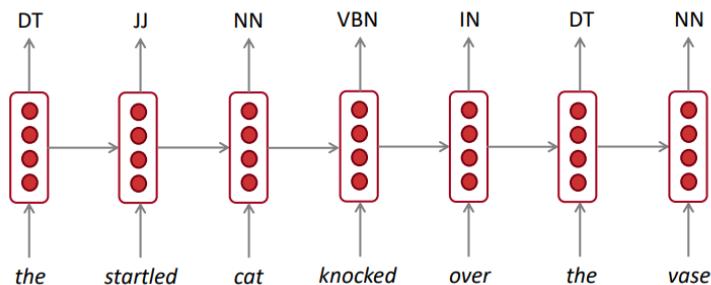
Perplexity improves  
(lower is better)

Perplexity가 낮을 수록 좋은 Language Model

# RNN은 이런 거에도 쓰일 수 있어요 !

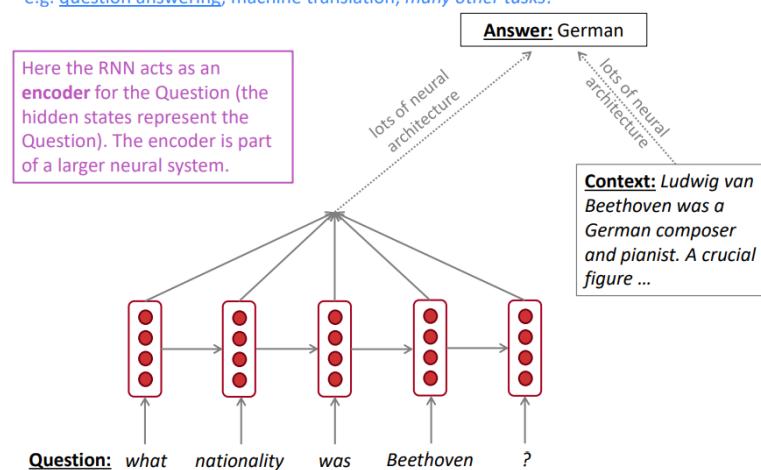
## RNNs can be used for tagging

e.g. part-of-speech tagging, named entity recognition



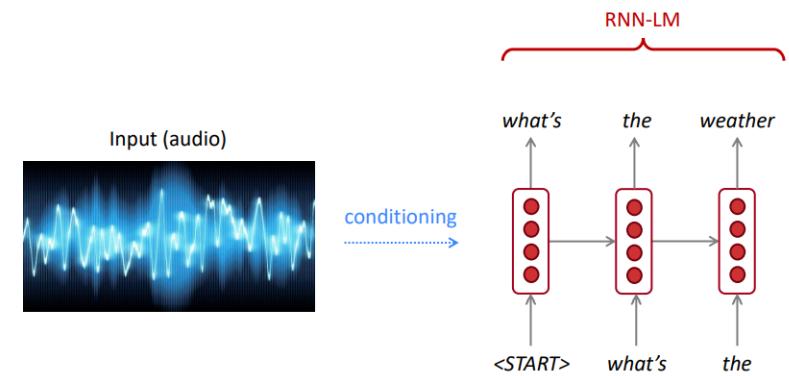
## RNNs can be used as an encoder module

e.g. question answering, machine translation, many other tasks!



## RNN-LMs can be used to generate text

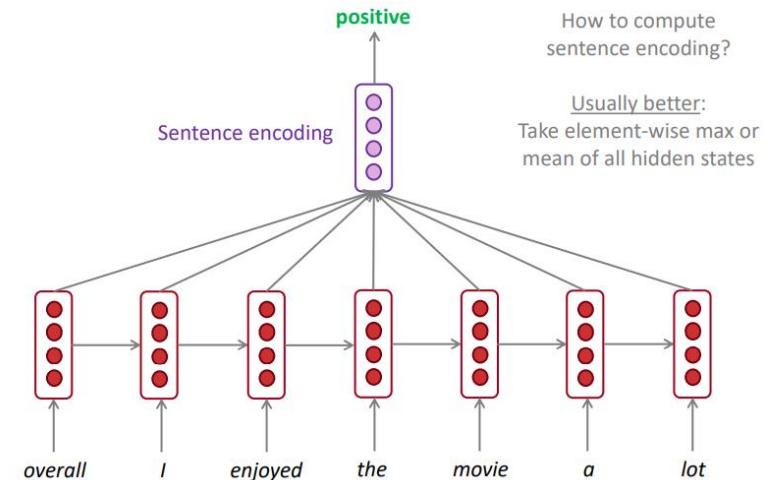
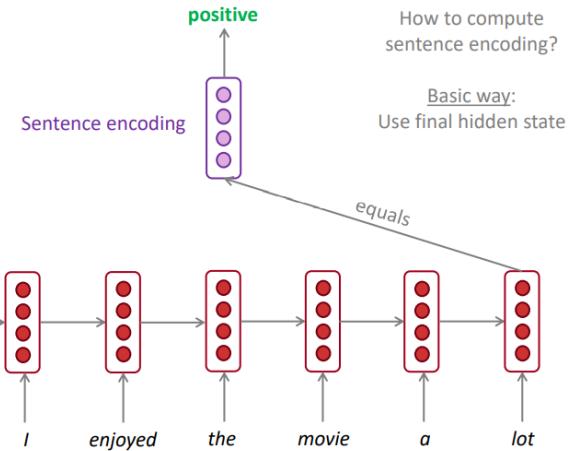
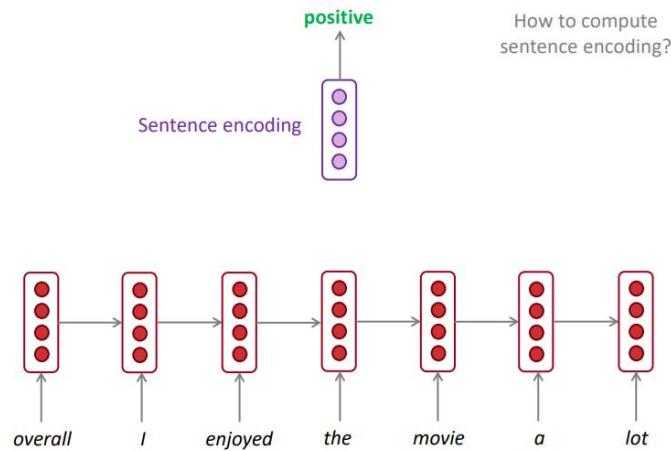
e.g. speech recognition, machine translation, summarization



# RNN은 이런 거에도 쓰일 수 있어요 !

RNNs can be used for sentence classification

e.g. sentiment classification



## Recap

### 질문!

기본적인 n gram 부터 랭기지모델, rnn,(lstm) 까지 비교해서 한번 설명해주실 수 있나요?

구지식과 신지식이 머릿속에서 싸우고 있어요,,,@

답변

N-gram 언어 모델은 카운트에 기반해서, 통계적으로 접근한 모델입니다.

예측하려는 단어 이전의 N개의 단어들만 보는데, Sparsity, Storage, Incoherence 문제가 있어 해결하고자 등장한 것이 Fixed-Window neural 모델이고, 이는 신경망을 이용하는데 Sparsity 문제는 해결하지만 여전히 문맥을 반영하지 못하는 Incoherence 문제 가 남아있습니다.

이 두가지 모델은 이전의 N개의 단어만 봤다면 다음으로 등장하는 RNN은 문장 전체를 순차적으로 보는 특징이 있습니다.

그리고 다음시간에 배울 예정인 LSTM은 Singe layer인 RNN과 다르게 더 복잡한 구조를 가진다! 실제로 RNN의 성과들은 대부분 LSTM을 통해서 이루어 졌다! 정도만 언급하겠습니다. ㅎ.ㅎ

# RNN이란... & 다음 시간에는 이런 것을 배울 거예요 !

## A note on terminology

RNN described in this lecture = “vanilla RNN”  
(*simple RNN*)



Next lecture: You will learn about other RNN flavors



and multi-layer RNNs



By the end of the course: You will understand phrases like  
“stacked bidirectional LSTM with residual connections and self-attention”



트핑일봉~

- Problems with RNNs!

- Vanishing gradients

motivates

- Fancy RNN variants!

- LSTM
- GRU
- multi-layer
- bidirectional

## 출처

- CS224n Lecture 6 강의 & 강의 자료
- 기타 블로그들

<https://misconstructed.tistory.com/36>

<https://aikorea.org/blog/rnn-tutorial-1/>

<http://dsba.korea.ac.kr/seminar/?category1=Lecture%20Review&mod=document&pageid=1&uid=42>

<https://heiwais25.github.io/nlp/2019/10/06/Language-model-2/>

<https://jiho-ml.com/weekly-nlp-18/>

<https://pakalguksu.github.io/2020/03/01/CS224n-6%EA%B0%95-Language-Models-and-Recurrent-Neural-Networks/>