

CS224n 스터디

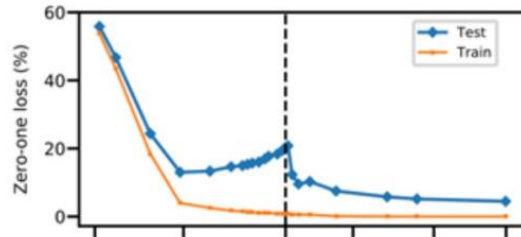
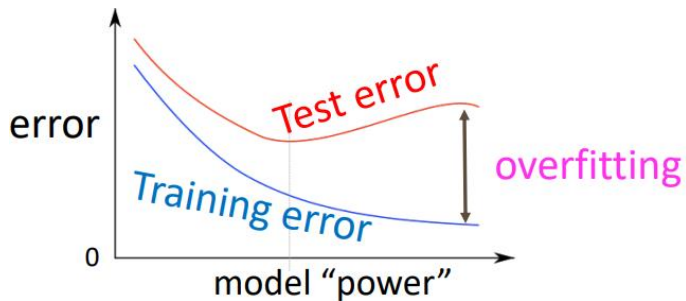
Lecture 5: Language Models and Recurrent Neural Networks

2025.03.31

발표자 : 김지호

내용

Regularization



과거의 관점: Train 데이터 overfit되면 Test error는 증가

현재의 관점: Train 데이터에 overfit 되더라도 Regularization 과정을 잘 거치면 일반적인 데이터에도 좋은 성능을 기대할 수 있음

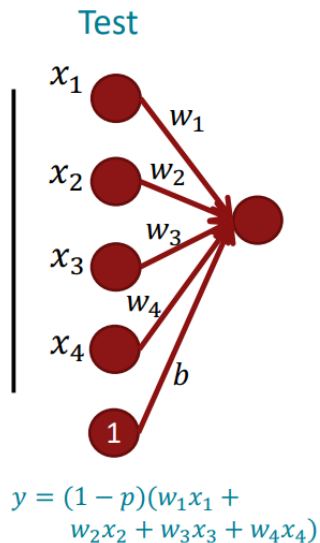
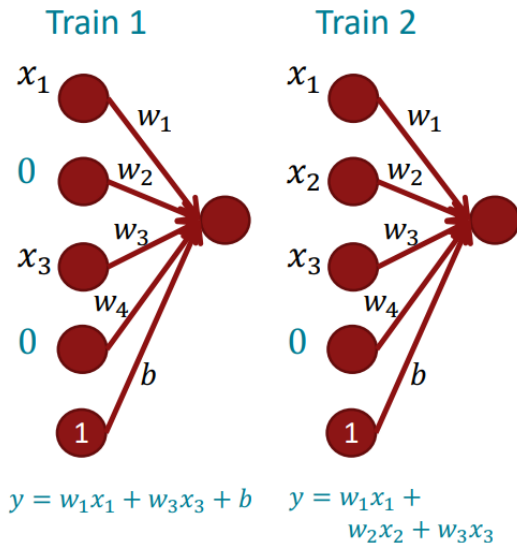
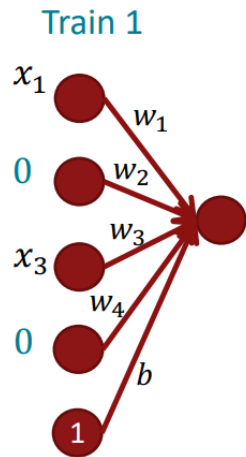
내용

Dropout

Overfit을 방지하기 위해
일부 뉴런을 0으로 만듦

이를 구현하기 위해 0,1로 이루어진
마스크 행렬을 hadamard product
수행하여 일정 비율로 0으로 만듦

Test 과정에서는 Dropout을 적용하지
않지만 Train과의 출력 스케일을 맞춰주기 위해
1-p를 곱하는 스케일링 과정이 추가됨



내용

Parameter initialization

파라미터 초기값 설정 시 모두 같은 값을 주면
모든 노드의 학습이 동일하게 일어남

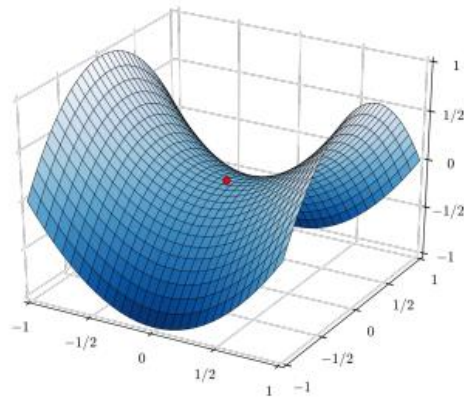
이를 방지하기 위해 랜덤하게 파라미터의 초기값을
임의의 값으로 설정해야 함

Xavier initialization

n_{in} : previous layer size

n_{out} : next layer size

$$\text{Var}(W_i) = \frac{2}{n_{in} + n_{out}}$$



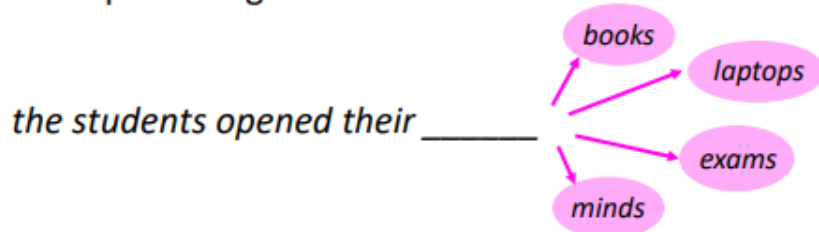
내용

Optimizer

- Adagrad ← Simplest member of family, but tends to “stall early”
- RMSprop
- Adam ← A fairly good, safe place to begin in many cases
- AdamW
- NAdamW ← Can be better with word vectors (W) and for speed (Nesterov acceleration)

내용

Language Modeling is the task of predicting what word comes next



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

Language Modeling

첫번째 시점부터 t번째 시점까지의 단어 시퀀스가 주어졌을 때,
t+1 시점에 올 단어의 확률 분포를 구하는 것

내용

우리는 이러한 Language model을 텍스트에 확률을 부여하는 시스템으로 이해할 수 있음

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

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



This is what our LM provides

내용

N-gram

딥러닝 이전에 Language Model을 학습시키기 위해 사용

정의: n개의 연속된 단어의 chunk

- **unigrams**: "the", "students", "opened", "their"
- **bigrams**: "the students", "students opened", "opened their"
- **trigrams**: "the students opened", "students opened their"
- **four-grams**: "the students opened their"

Markov Assumption : t+1 번째 단어는 오직 앞의 n-1개의 단어에만 의존한다

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

마코프 가정 적용

prob of a n-gram

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

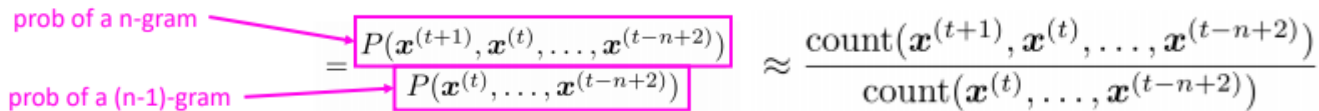
prob of a (n-1)-gram

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

위 가정을 조건부 확률 공식으로 표현
→ 어떻게 이 확률을 계산할까?

내용

코퍼스에서 해당 n-gram과 n-1 gram의 수를 Count


$$\begin{aligned} \text{prob of a n-gram} &\rightarrow P(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}) \\ &= P(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}) \\ \text{prob of a (n-1)-gram} &\rightarrow P(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}) \end{aligned} \approx \frac{\text{count}(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}{\text{count}(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}$$

예를 들어 4-gram 모델을 학습할 때 코퍼스에서
'students opened their' 이라는 문장이 1000번 등장하였고
'students opened their books' 라는 문장이 400번 등장하였으면
 $P(\text{books} \mid \text{students opened their}) = 400/1000 = 0.4$

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

내용

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

문제점 1: 분자가 0, 즉 'students opened their \mathbf{w} '가 코퍼스에 1번도 등장하지 않으면 그 확률은 0이 됨

→ 작은 델타값(0.25)를 모든 \mathbf{w} 에 추가해 줌: 확률이 0이 되는 것을 방지

문제점 2: 분모가 0, 즉 'students opened their' 이 코퍼스에 등장하지 않으면 확률을 계산할 수 없음

→ 더 작은 n-gram을 사용, 즉 'opened their'을 대신 사용함. 이를 **backoff**라 부름

문제점 3: n값이 증가하거나 코퍼스의 크기가 커지면 모델의 크기도 커짐

내용

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 .

문제점 4: 문법적으로는 올바르지만, 문장의 일관성이 떨어짐
이는 다음 단어를 예측할 때 이전 $n-1$ 개만의 단어만 참고하기 때문
 n 을 증가시키면 이러한 문제를 해결 할 수 있지만, 이로 인해 Sparsity 문제가 나타남

내용

Fixed-window Neural Language Model

Sparsity 문제 해결

모든 n-gram을 저장하지 않아도 됨

문제점

Fixed window의 크기가 너무 작음

Window의 크기가 늘어나면

W(가중치)의 크기도 늘어남

같은 단어라도 자리가 다르면 다른

가중치를 사용 – No symmetry

→ 어떠한 문장길이도 처리 가능한

신경망 구조가 필요함

output distribution

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

hidden layer

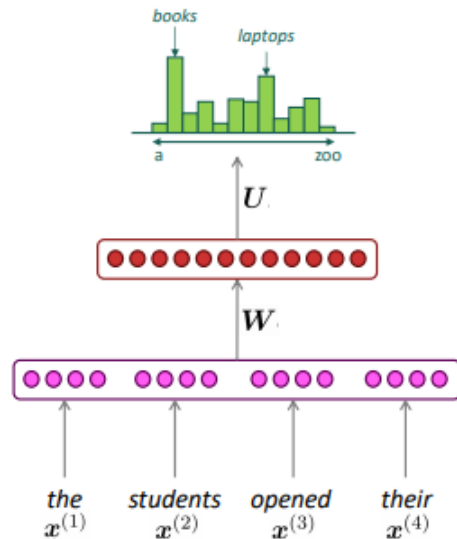
$$h = f(We + b_1)$$

concatenated word embeddings

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

words / one-hot vectors

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



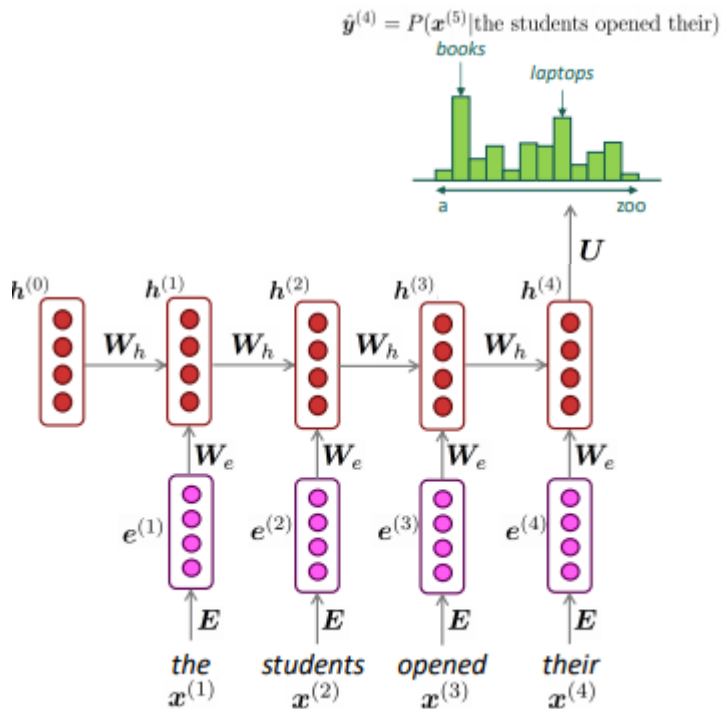
내용

Recurrent Neural Language Model

각 층마다 같은 가중치 행렬 적용
최종 은닉층을 기반으로 softmax를
통해 단어 확률 분포를 계산함

장점: 어떠한 input 길이가 들어와도
처리 가능, 같은 가중치 사용

단점: 반복 계산에 시간이 오래 소요,
현 시점과 멀리 떨어진 시점의 정보에
접근하는 것이 어려움



내용

RNN 학습

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

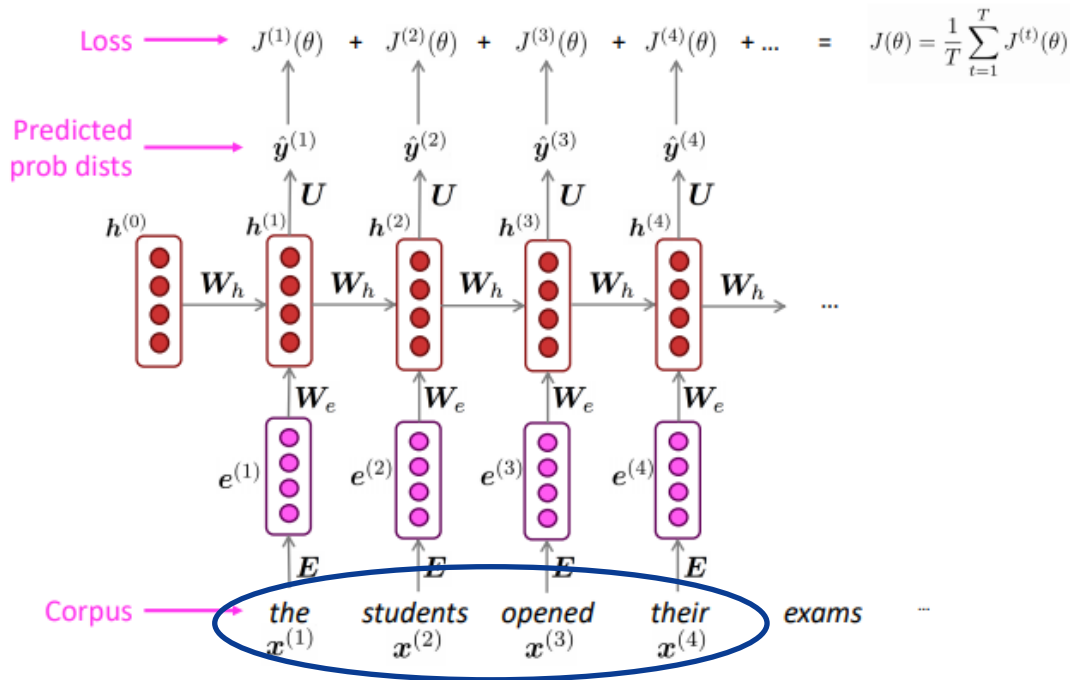
각 시점의 손실함수: 예측된 확률분포와 실제 다음 단어의 cross entropy

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

전체 손실함수: 각 시점의 loss의 평균을 계산

내용

RNN 학습



Teacher forcing: 다음 단어 예측 시 입력값으로 이전 층에서 예측한 값이 아닌 실제 정답을 입력함
잘못된 예측에 따른 오차가 누적되어 학습을 방해하는 것을 방지

내용

RNN 학습

모든 시점의 예측 확률 - 실제 정답 사이 cross entropy 구하고, 이를 평균 내는 손실을 구해야 함

이를 코퍼스 전체에 계산하려면 연산량과 메모리 소모가 증가

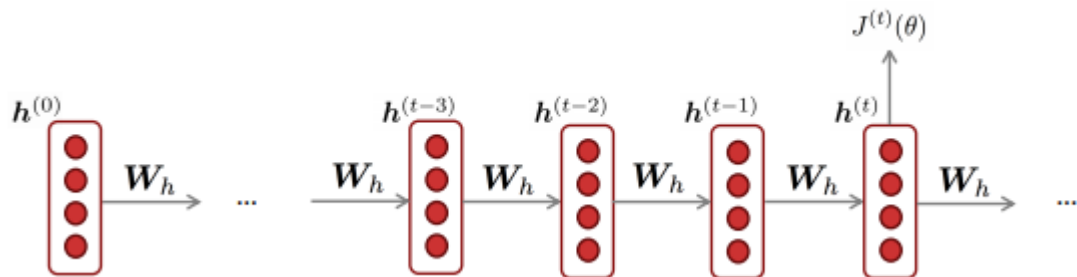
$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta)$$

이를 해결하기 위해 텍스트를 여러 개의 문장이나 배치로 나누어 학습
각각의 묶음에 대해 Loss와 Gradient를 계산하여 모델 업데이트

강의에서는 100개의 단어로 segment를 나누어 vectorized 연산을 수행할 수
있다고 함

내용

t시점 손실함수에 대한 가중치 W_h 의 편미분값 구하기



Question: What's the derivative of $J^{(t)}(\theta)$ w.r.t. the **repeated** weight matrix W_h ?

Answer:
$$\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^t \frac{\partial J^{(i)}}{\partial W_h} \Big|_{(i)}$$

t시점의 손실함수를 1~t시점 까지의 W_h 가중치 행렬로 편미분한 값을 모두 더함
왜 이렇게 구할 수 있는가?

내용

Multivariable Chain Rule

$$\underbrace{\frac{d}{dt} f(\mathbf{x}(t), \mathbf{y}(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt} + \frac{\partial f}{\partial \mathbf{y}} \frac{d\mathbf{y}}{dt}$$

Apply the multivariable chain rule:

= 1

$$\frac{\partial J^{(t)}}{\partial \mathbf{W}_h} = \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial \mathbf{W}_h} \bigg|_{(i)} \boxed{\frac{\partial \mathbf{W}_h|_{(i)}}{\partial \mathbf{W}_h}} = \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial \mathbf{W}_h} \bigg|_{(i)}$$

t시점의 손실함수: 1~t시점까지 \mathbf{W}_h 를 반복 적용하여 업데이트된 결과물

t시점의 손실함수를 \mathbf{W}_h 로 미분할 때, 다변수 chain rule에 의하여 t시점의 손실함수를 각 시점의 \mathbf{W}_h 로 미분한 값의 합으로 표현할 수 있음

이 때 각 시점의 \mathbf{W}_h 는 사실 모두 같은 값이므로 보라색 부분 값은 1이 됨

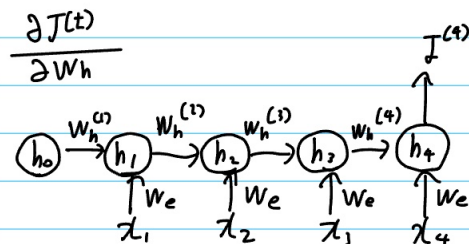
내용

Multivariable Chain Rule

Ex) $t = 4$ 일 때 예시 풀이

주의)

$W_h(i)$ 는 그 값은 모두 같지만,
손실함수를 각 시점의 $W_h(i)$ 로
미분한 결과는 모두 다름



$$\left. \begin{aligned} h^{(1)} &= f(w_h^{(1)} h^{(0)} \sim) \\ h^{(2)} &= f(w_h^{(2)} h^{(1)} \sim) \\ h^{(3)} &= f(w_h^{(3)} h^{(2)} \sim) \\ h^{(4)} &= f(w_h^{(4)} h^{(3)} \sim) \end{aligned} \right\}$$
$$\hat{y}^{(4)} = \text{Softmax}(\sim h^{(4)} \sim)$$
$$J^{(4)} = L(\hat{y}^{(4)}, y^{(4)})$$

즉, 4번째 시점의 Loss는 매 시점의 $W_h^{(i)}$ 에 의해 영향은 받음.
 \therefore multivariable chain rule에 의해

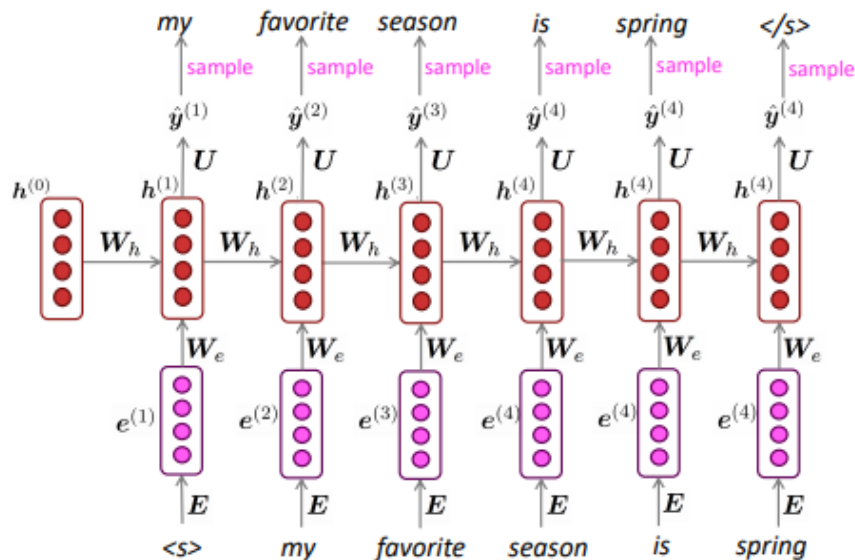
$$\frac{\partial J^{(4)}}{\partial w_h} = \sum_{i=1}^4 \frac{\partial J^{(4)}}{\partial w_h^{(i)}} \times \frac{\partial w_h^{(i)}}{\partial w_h} \quad \text{2 표현 가능.}$$

$$\text{이 때 } w_h^{(i)} = w_h \text{ 이므로}$$

$$= \sum_{i=1}^4 \frac{\partial J^{(4)}}{\partial w_h^{(i)}}$$

내용

RNN을 이용한 text generating



첫 입력으로 시퀀스의 시작을 알리는 $<s>$ 가 입력
마지막 출력은 시퀀스의 끝을 알리는 $</s>$ 가 출력