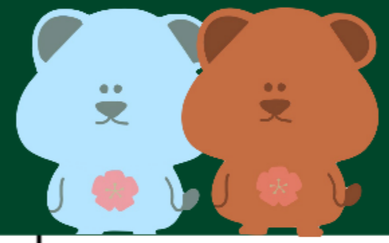
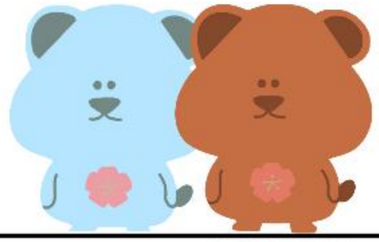


Recurrent Neural Networks

김경민, 김수한

EWHA EURON

목차



EWHA EURON

Dependency Parser

EWHA EURON

#01 Dependency Parser

Transition-based Dependency Parsing

두 단어의 의존 여부를 차례대로 결정해나가면서 Dependency Structure를 구성해나가는 방법

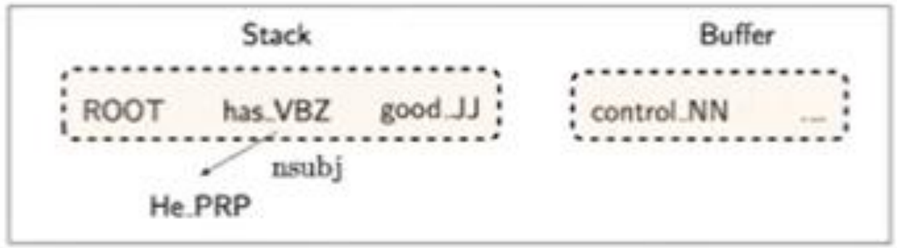
- 문제
  - Sparse
  - Incomplete
  - Expensive Computation

#01 Dependency Parser

Neural Dependency Parser

First win: Distributed Representations

- 각 단어를 d-dimensional dense vector로 표현 (비슷한 단어는 유사한 벡터, like word embedding)
- 품사태그나 dependency labels들도 d-dimensional vector로 표현



	word	POS	dep.
s_1	good	JJ	Ø
s_2	has	VBZ	Ø
b_1	control	NN	Ø
$lc(s_1)$	Ø	Ø	Ø
$rc(s_1)$	Ø	Ø	Ø
$lc(s_2)$	He	PRP	nsubj
$rc(s_2)$	Ø	Ø	Ø

A concatenation of the vector representation of all these is the neural representation of a configuration

#01 Dependency Parser

Neural Dependency Parser

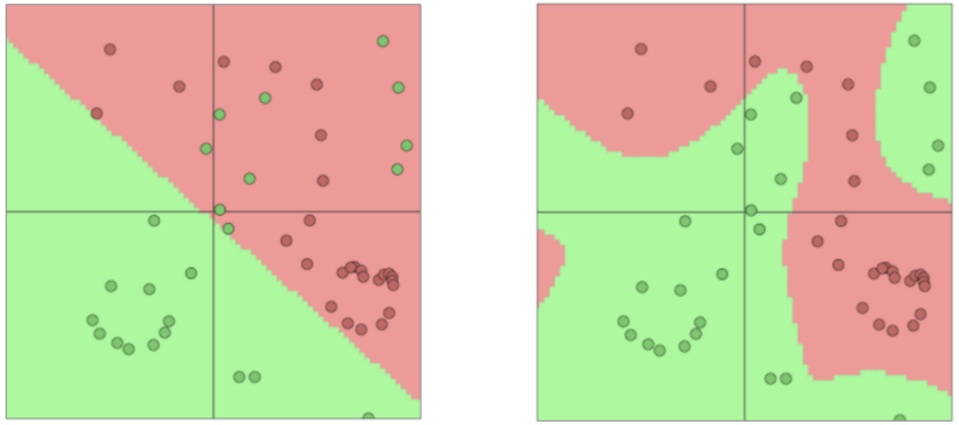
Second win: Deep Learning classifiers are non-linear classifier

- 전통적인 ML 분류기는 linear decision boundaries 때문에 그다지 강력 x

  (Naïve Bayes, SVMs, logistic regression and softmax classifier)

- Neural networks는 non-linear decision boundaries -> learn more complex function

- Non-linear in the original space, linear for the softmax at the top of the neural network

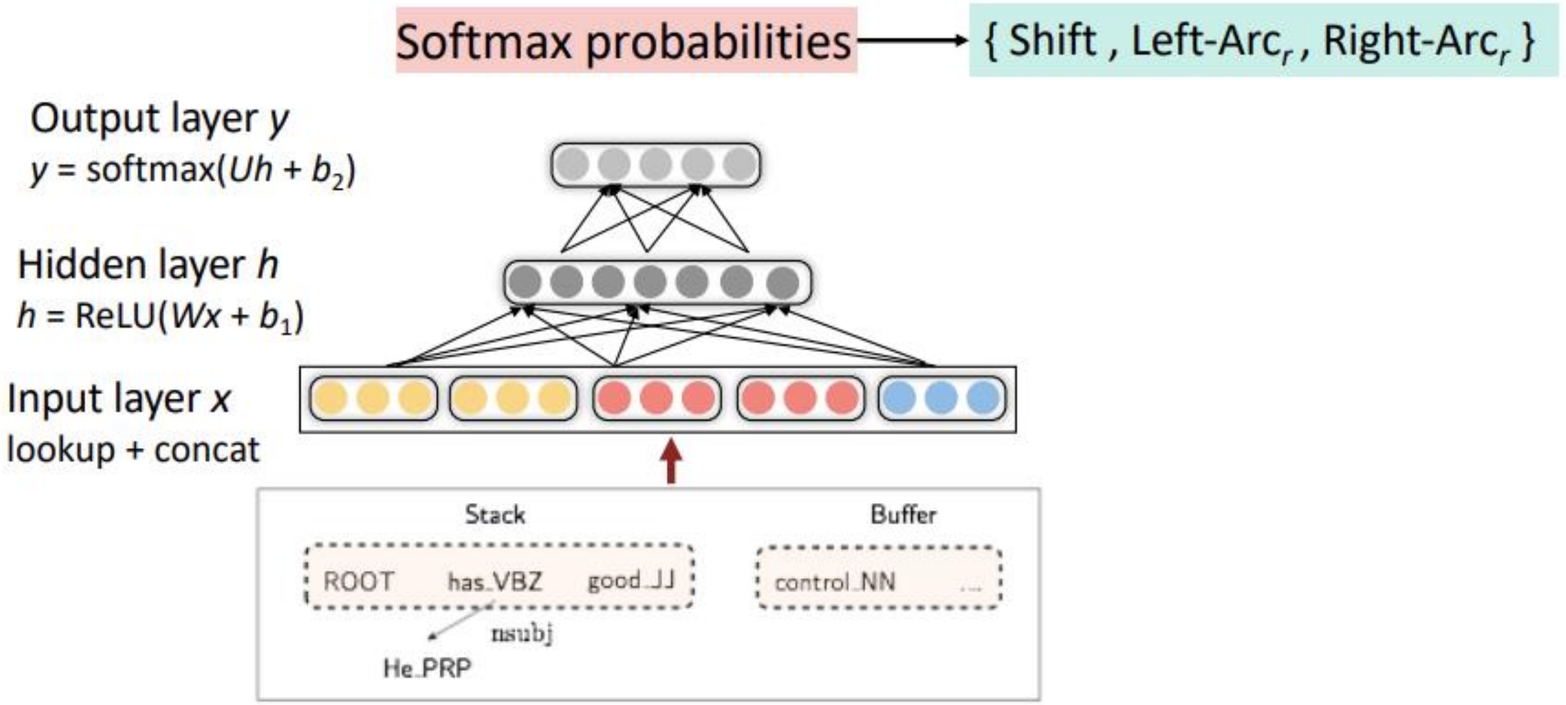


Visualizations with ConvNetJS by Andrej Karpathy!

http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

#01 Dependency Parser

Neural Dependency Parser Model Architecture



#01 Dependency Parser

Developments in transition-based dependency parsing

- Bigger, deeper networks with better tuned hyperparameters
- Beam search
- Global, conditional random field (CRF)-style inference over the decision sequence

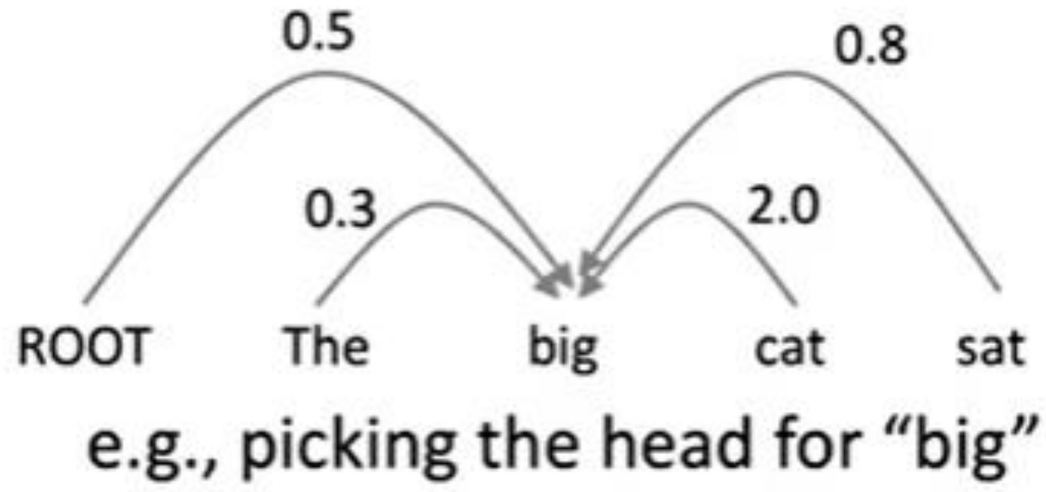
Method	UAS	LAS (PTB WSJ SD 3.3)
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79

#01 Dependency Parser

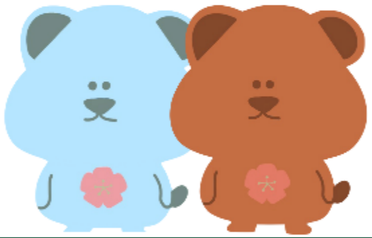
Graph-based Dependency Parser

각 단어에 대해 가능한 모든 dependency score를 연산

- 단순히 두 단어에 대해서가 각 word token에 대한 아니라 알맞은 문맥상의 표현



Language Model



EWHA EURON

#02 Language Model

Language Modeling

Language Modeling : 확률 분포를 기반으로 주어진 문맥 이후에 위치할 단어를 예측

- The students opened their (books? laptops? exams? minds?)

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

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

This is what our LM provides

#02 Language Model

n-gram Language Model

n-gram Language Model : 이전에 등장한 n개의 단어를 바탕으로 다음에 위치할 단어를 예측

- 단어의 문장 안 등장 빈도를 기반으로 확률 계산
- n-gram : a chunk of n consecutive words

종류 (조건부 확률 또는 확률 값을 계산할 때 고려하는 단어 개수에 따라서 분류)

- 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”

#02 Language Model

n-gram Language Model

Suppose we are learning a 4-gram Language Model.

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

~~as the proctor started the clock, the~~ : discard

students opened their : condition on this

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

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(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?

#02 Language Model

n-gram Language Model

단점

1) Sparsity Problem
   - N-gram이 문서 내에서 등장하지 않는 경우, 확률 계산 식의 분자나 분모가 0이 되어서 확률 값이 0이 되거나, 아예 계산 자체가 불가능한 상황이 됨

2) Storage Problem
   - 문맥 안의 모든 n-gram 빈도를 저장해야 함
   - N이 늘어나거나 문맥이 길어질수록 모델의 용량도 증가

#02 Language Model

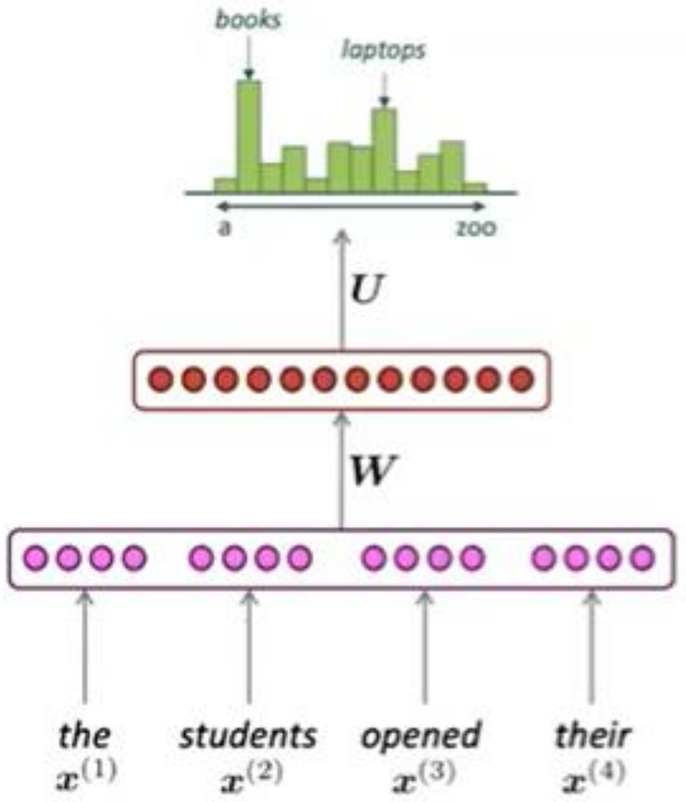
Neural Network Language Model

Neural Network Language Model :

미리 지정한 window size 이전 단어를 바탕으로 문맥

다음에 위치할 단어 예측

- Window size를 미리 지정함으로써 Sparsity Problem 방지
- Word embedding : 단어의 distributed representation (임베딩 공간 상에서 비슷한 의미를 가진 단어들은 비슷한 위치에 존재) 학습



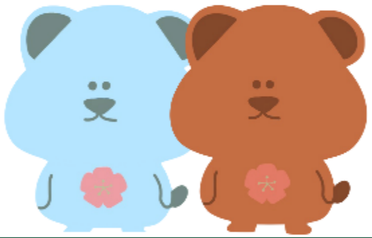
#02 Language Model

Neural Network Language Model

단점

- Window size가 커지면 계산할 파라미터 수 증가 -> 연산 부담 과중, 과적합 가능성 증가
- 입력 길이가 고정되어 이전에 등장하는 모든 단어를 고려할 수는 없음 (n-gram도 동일)

순환신경망(Recurrent neural network ,RNN)



#3 순환신경망(Recurrent neural network ,RNN)

#1 contents **순환신경망 언어모델(Recurrent Neural Network Language Model, RNNLM)**

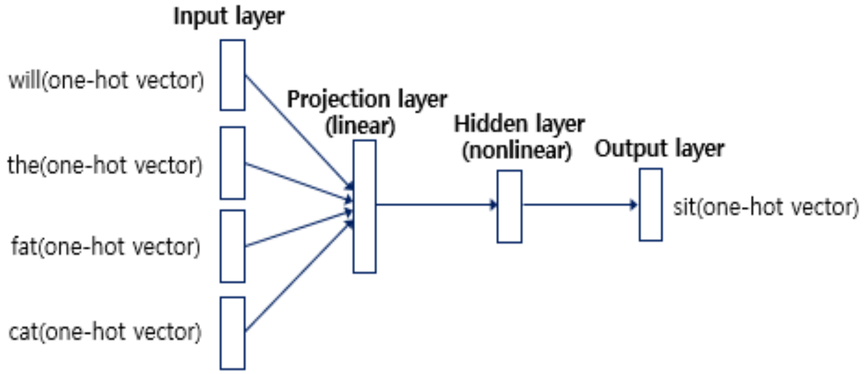
#2 contents **RNNLM의 학습과정**

#3 contents **Backpropagation(BPTT)**

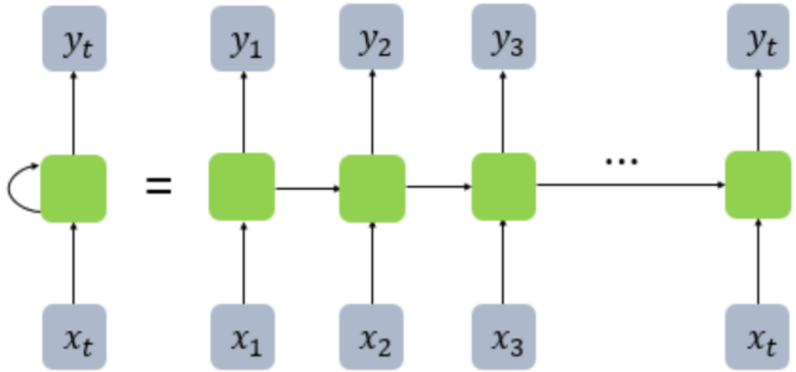
피드포워드 신경망
(Neural Network Language Model, NNLM)

순환신경망
((Recurrent neural network ,RNN)

"what will the fat cat sit on"



A: 시점도입

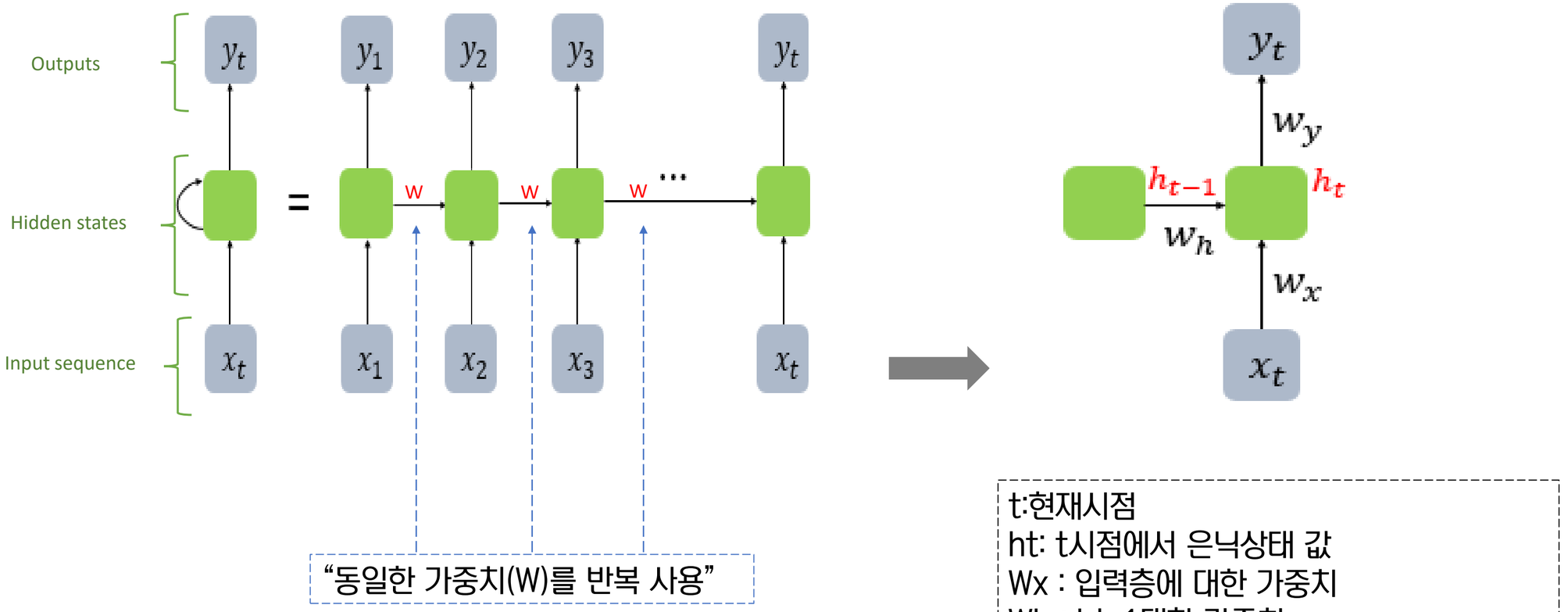


Q : 특정 단어만 인코딩 (문맥 반영이 어려움.)

위치에 따라 다른 가중치(비슷한 내용 여러 번 학습해야 함.
, 입력크기가 증가하면 모델크기도 증가함.)

∴ 가중치를 반복적으로 사용
(입력길이 제한이 개선됨.
,입력에 따른 모델의 크기 문제가 개선됨.)

#3 순환신경망((Recurrent neural network ,RNN)

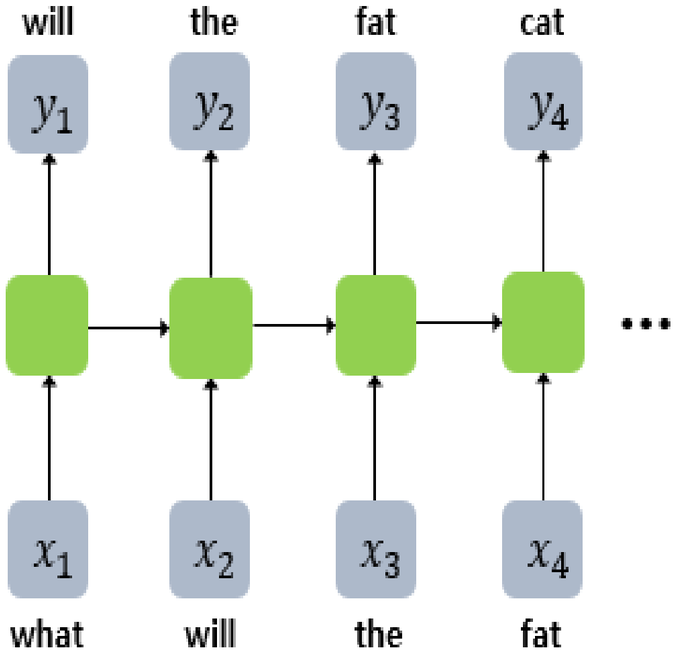


t:현재시점
ht: t시점에서 은닉상태 값
Wx : 입력층에 대한 가중치
Wh : ht-1대한 가중치

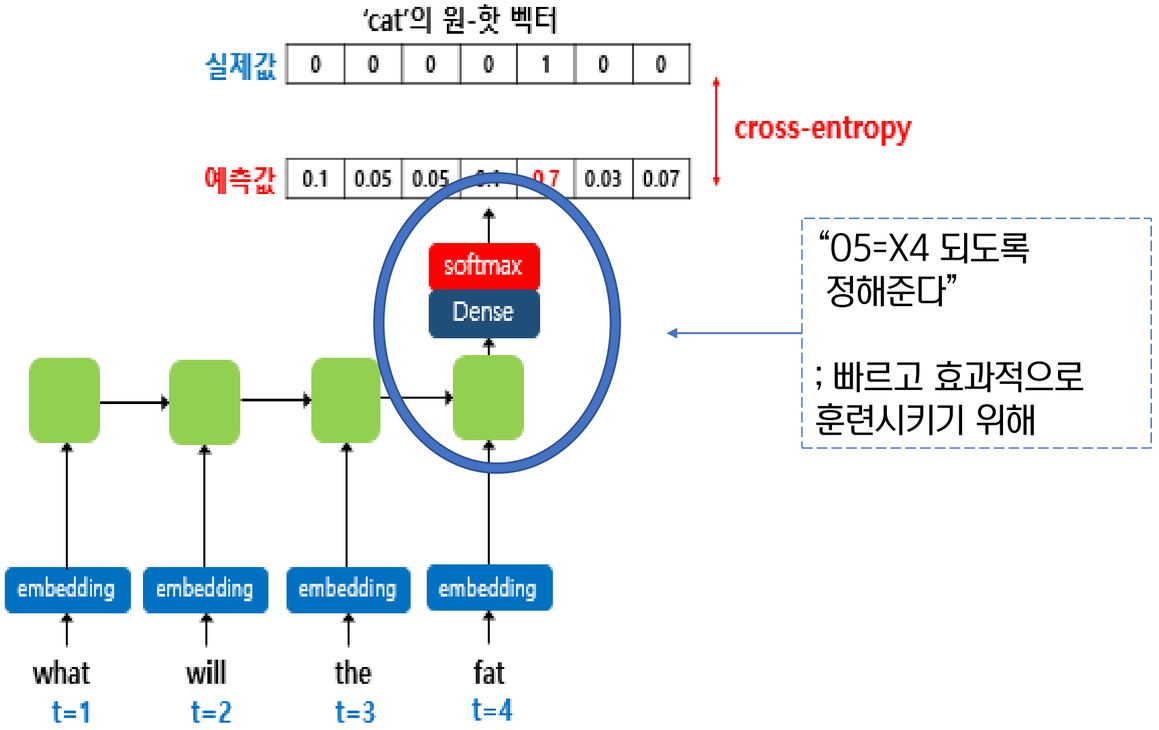
은닉층 >ht=tanh(Wx*xt+ Wh* ht-1 +b)
출력층 > yt=f(wy*ht+b)

#3 순환신경망 언어모델(Recurrent Neural Network Language Model, RNNLM)

•예문 : 'what will the fat cat sit on'



-테스트 과정(실제 사용할 때)



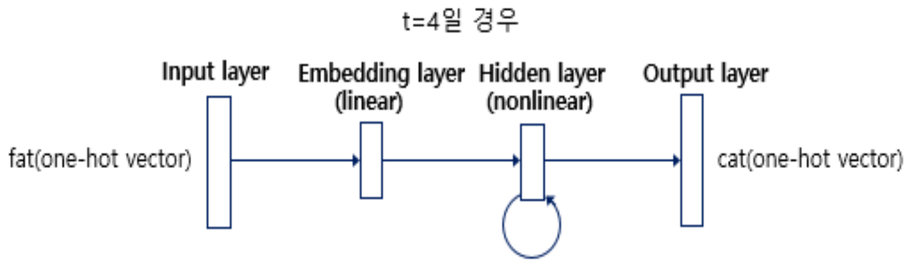
"05=X4 되도록 정해준다"

; 빠르고 효과적으로 훈련시키기 위해

-훈련 과정(teacher forcing)

#3 순환신경망 언어모델(Recurrent Neural Network Language Model, RNNLM)

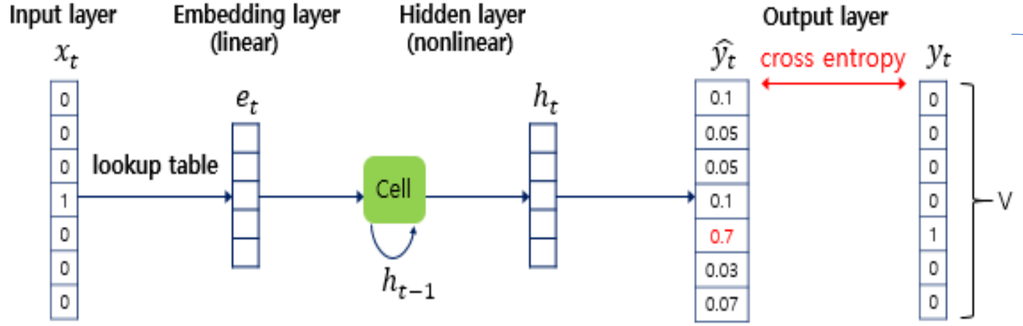
•예문 : 'what will the fat cat sit on'



입력층 :4번째 입력 단어인 fat의 원-핫 벡터가 입력

출력층 : 정답에 해당되는 단어 cat의 원-핫 벡터 사용
(오차를 구하기 위해)

→ 오차로부터 손실 함수를 사용해 인공 신경망이 학습됨.



Cf) lookup table:입력벡터와 W행렬의 곱

임베딩층 : et=lookup(xt)
은닉층: ht=tanh(Wx*et+ Wh* ht-1 +b)
출력층: y^t=softmax(wy*ht+b)
; j번째 인덱스가 가진 값은 j번째 단어가 다음 단어일 확률을 나타냄

↓

cross entropy를 통해 실제 값과 예측 값의 오차 계산

↓

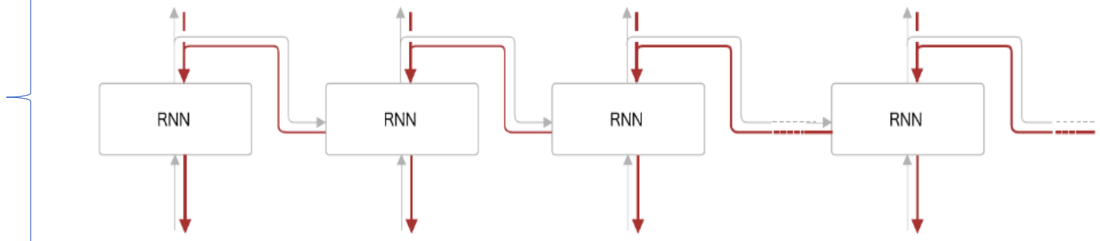
역전파가 이루어지면서 가중치 행렬들이 학습됨

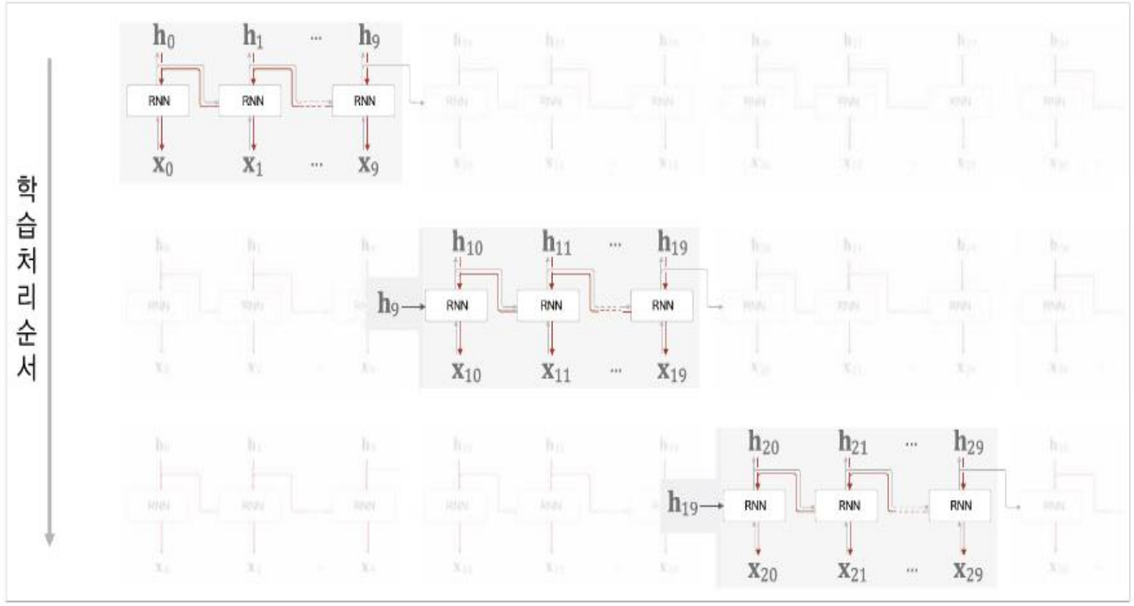
#3 시간 방향으로 펼친 신경망의 오차역전파법(BPTT)

•BPTT (Backpropagation Through Time)
:시간 방향으로 펼친 신경망의 오차역전파

•시계열 데이터를 학습할 때,
시간 크기가 커지면서 BPTT가 소비하는 컴퓨팅
자원 증가 -> 역전파시 기울기 불안정

그림 5-10 순환 구조를 펼친 RNN 계층에서의 오차역전파법





• “적당한 지점에서 잘라내 작은 RNN 신경망 여러 개로 만들자”
(작은 신경망에서 오차 역전파법을 수행)

•순전파의 흐름은 끊어지지 않고 전파하고
역전파의 연결은 적당한 길이로 잘라,
잘라낸 신경망 단위로 학습 수행

EWHA EURON

#4 Evaluating Language Model - perplexity

EWHA EURON

#4 Evaluating Language Model - perplexity

#1 contents Evaluating Language Model - perplexity

#4 언어모델의 평가 -펄플렉서티(perplexity,PPL)

문장 W의 길이가 N이라 하면,

$PPL(W)=P(w1, w2, .., wn)^{-1/n}$

→ $\sqrt{\frac{1}{\pi_{i=1}^{n}}} p(w_i|w_1, .., w_{i-1})$

PPL(W)=10 'PPL이 100이다'

→ 해당 언어 모델이 모든 스텝마다 평균 10개의 단어를 가지고 어떤 것이 정답인지 고민하고 있다.

펄플렉서티(PPL)

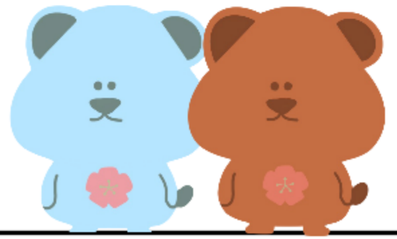
:조금은 부정확할 수 있으나 모델 내에서 자신의 성능을 수치화하여 결과를 내놓는 평가방법

;낮을수록 언어 모델의 성능이 좋음

;PPL=문장의 길이로 정규화된 문장 확률의 역수

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

THANK YOU



EWHA EURON