

# Language model, RNN, RNN based model

모바일 플랫폼TF팀 이경임

# References

## 위키독스

- 언어 모델이란?
- 통계적 언어모델
- N-gram 언어 모델
- 순환신경망
- RNN 언어모델

## 실습

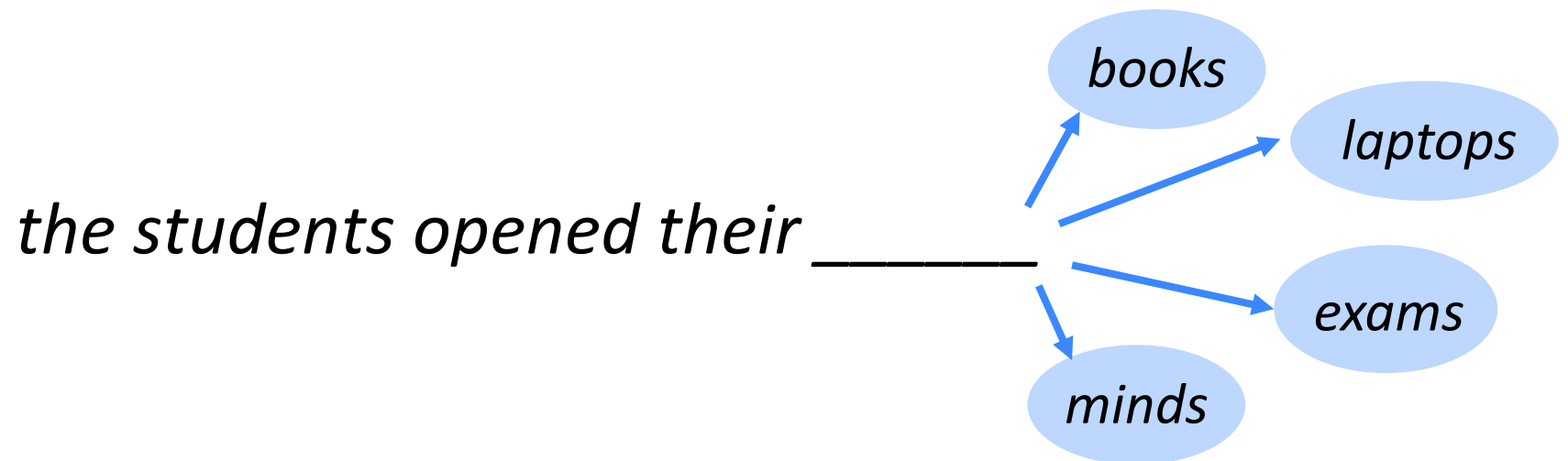
- 글자 단위 RNN

## 기타 자료

- 한국어 임베딩 - 이기창
- CS224n - Recurrent Neural Networks and Language Models

# 언어모델이란?

- 자연어의 통계적 패턴
  - 문장에 어떤 단어가 (많이) 쓰였는가 : 백오브워즈 가정
  - 단어가 어떤 순서로 등장하는가 : 언어모델
  - 어떤 단어가 같이 나타났는가 : 분포 가정



# 언어모델이란?

- 딥러닝의 발전 이전에도 있었던 개념
- 언어를 모델링하고자 **단어 시퀀스에 확률을 부여**하는 모델이다.
- 잘 학습된 언어모델은 어떤 문장이 더 “자연스러운지”, 또한 주어진 시퀀스 다음에는 무엇이 오는게 자연스러운지를 알 수 있다.
- 단어가 N개 주어진 상황에서 언어모델은 N개 단어가 동시에 나타날 확률, 즉  $P(w_1, w_2, w_3 \dots w_n)$ 을 반환합니다.

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

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

# 단어 시퀀스에서의 확률 할당

## A. 단어 시퀀스의 확률

하나의 단어를  $w$ , 단어 시퀀스를 대문자  $W$ 라고 한다면,  $n$ 개의 단어가 등장하는 단어 시퀀스  $W$ 의 확률은 다음과 같습니다.

$$P(W) = P(w_1, w_2, w_3, w_4, w_5, \dots, w_n)$$

## B. 다음 단어 등장 확률

이제 다음 단어 등장 확률을 식으로 표현해보겠습니다.  $n-1$ 개의 단어가 나열된 상태에서  $n$ 번째 단어의 확률은 다음과 같습니다.

$$P(w_n | w_1, \dots, w_{n-1})$$

|의 기호는 조건부 확률(conditional probability)을 의미합니다.

# 통계적 언어모델(SLM)

- 딥러닝의 발전 이전에도 있었던 개념, 전통적 언어 모델
- 문장의 확률을 구하기 위해 **조건부 확률** 사용
- 이때의 확률값은 **카운트에 기반해 계산**

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

 This is what our LM provides

# 통계적 언어모델(SLM)

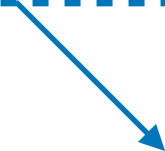
$$P(w_1, w_2, w_3, w_4, w_5, \dots, w_n) = \prod_{n=1}^n P(w_n | w_1, \dots, w_{n-1})$$

“An adorable little boy is spreading smiles” 문장이 등장할 확률

$P(\text{An adorable little boy is spreading smiles}) =$

$P(\text{An}) \times P(\text{adorable}|\text{An}) \times P(\text{little}|\text{An adorable}) \times P(\text{boy}|\text{An adorable little}) \times P(\text{is}|\text{An adorable little boy})$   
 $\times P(\text{spreading}|\text{An adorable little boy is}) \times P(\text{smiles}|\text{An adorable little boy is spreading})$

문장의 확률을 구하기 위해서 각 단어에 대한 예측 확률들을 곱합니다.


$$P(\text{is}|\text{An adorable little boy}) = \frac{\text{count}(\text{An adorable little boy is})}{\text{count}(\text{An adorable little boy})}$$

# N-gram 언어 모델

- 통계기반 언어모델의 일종. SLM과 같이 카운트 기반 통계적 접근을 사용한다.
- 전통적 SLM과 달리 이전에 등장한 모든 단어가 아닌 **일부 단어만 고려**하는 방법을 사용한다.
  - n-gram에서의 n은 코퍼스 내 단어들을 n개씩 묶어서 빈도를 학습했음을 의미한다.
- **이전 n-1개의 단어를 보고 n번째 단어를 예측**하는 방식
- 임의의 개수만큼의 이전 단어만 참고하여 확률을 근사
  - 코퍼스에서 해당 단어시퀀스를 카운트할 확률이 높아진다.



# N-gram 언어 모델 예시

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ *students opened their* \_\_\_\_\_  
discard 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
    - $\rightarrow P(\text{books} | \text{students opened their}) = 0.4$
  - “students opened their *exams*” occurred 100 times
    - $\rightarrow P(\text{exams} | \text{students opened their}) = 0.1$
- Should we have discarded the “proctor” context?

# N-gram 언어 모델 예시

표현	빈도
영원히	104
기억될	29
최고의	3503
명작이다	298
영원히 기억될	7
기억될 최고의	1
최고의 명작이다	23
기억될 최고의 명작이다	17
영원히 기억될 최고의 명작이다	0

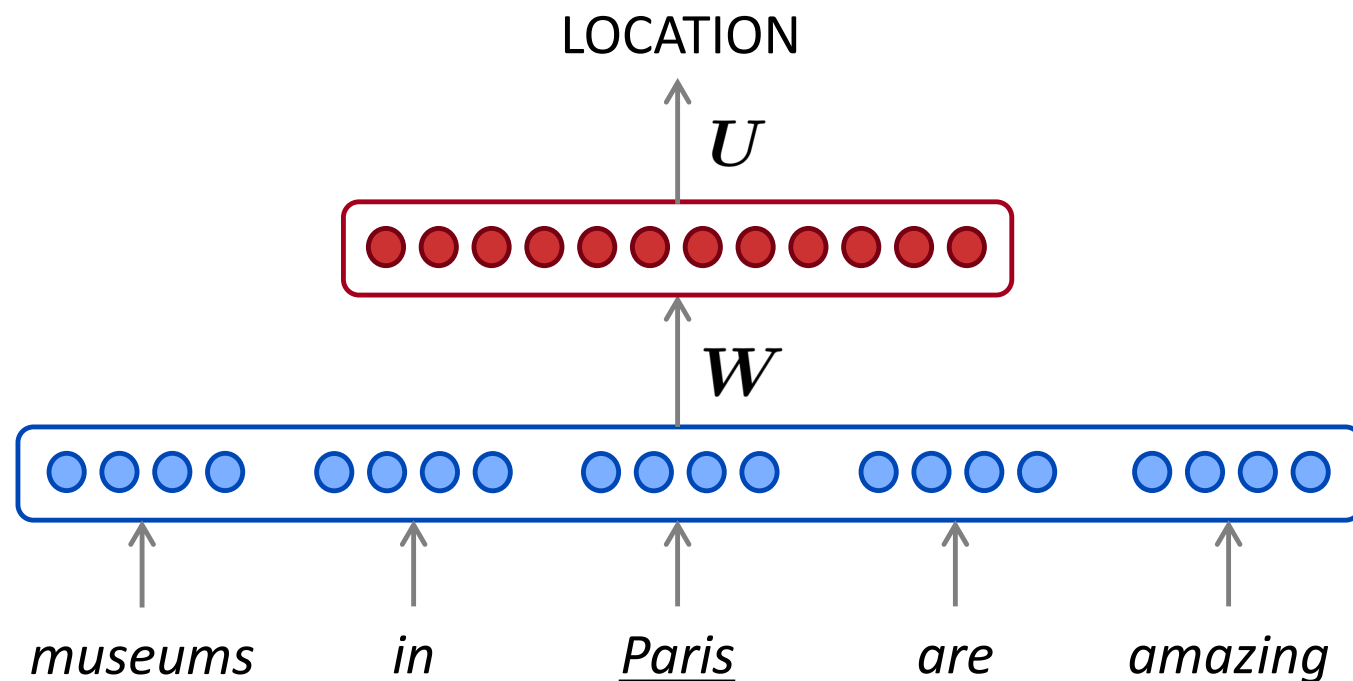
- 영원히 기억될 최고의 시퀀스 뒤에 '명작이다' 라는 단어가 올 확률을 **trigram**으로 근사해보면 얼마일까?

# N-gram 언어모델의 한계

- 희소문제(Sparsity problem)
  - 카운트 기반 접근방식의 본질적 한계
  - 코퍼스 내에 단어시퀀스가 없을 확률은 여전히 존재
- n의 선택은 trade-off
  - n크기를 키우면 : 예측 정확도 상승 but 희소문제 증가, 모델사이즈 증가
  - n크기를 줄이면 : 희소문제 감소 but 예측 정확도 감소
- long-term dependency : 정해진 개수의 이전 토큰만을 반영하므로 고려할 수 있는 시퀀스 범위 한정됨. >> 모델의 정확도와 연관

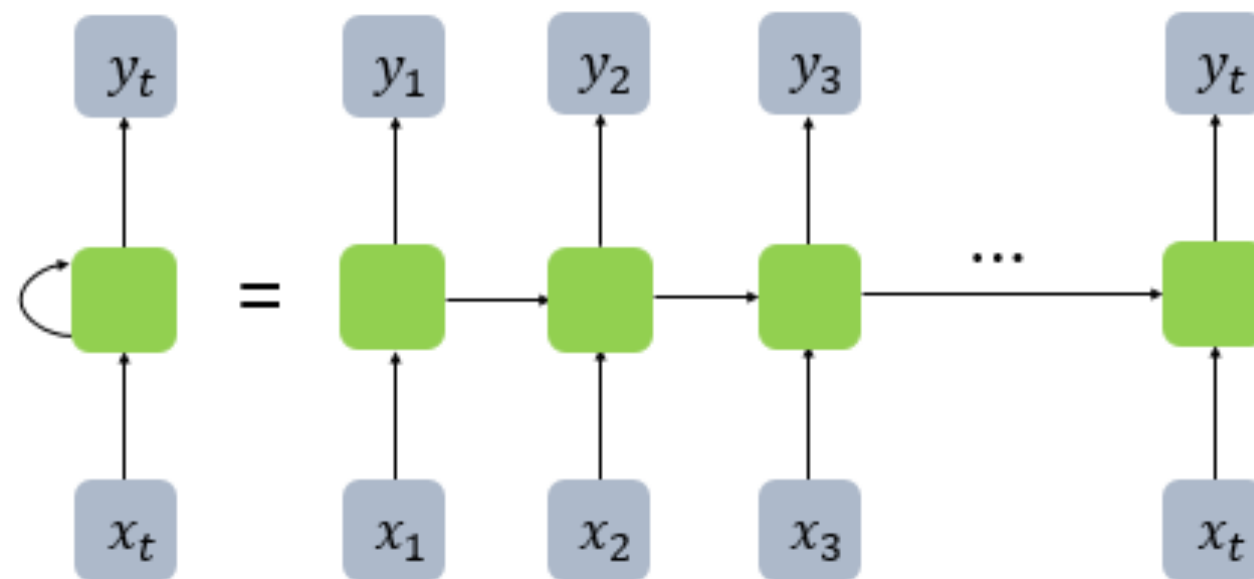
# NNLM : 신경망 기반 언어모델

- Recall the Language Modeling task:
  - Input: sequence of words  $x^{(1)}, x^{(2)}, \dots, x^{(t)}$
  - Output: prob dist of the next word  $P(x^{(t+1)} | x^{(t)}, \dots, x^{(1)})$
- How about a **window-based neural model**?
  - We saw this applied to Named Entity Recognition in Lecture 3:



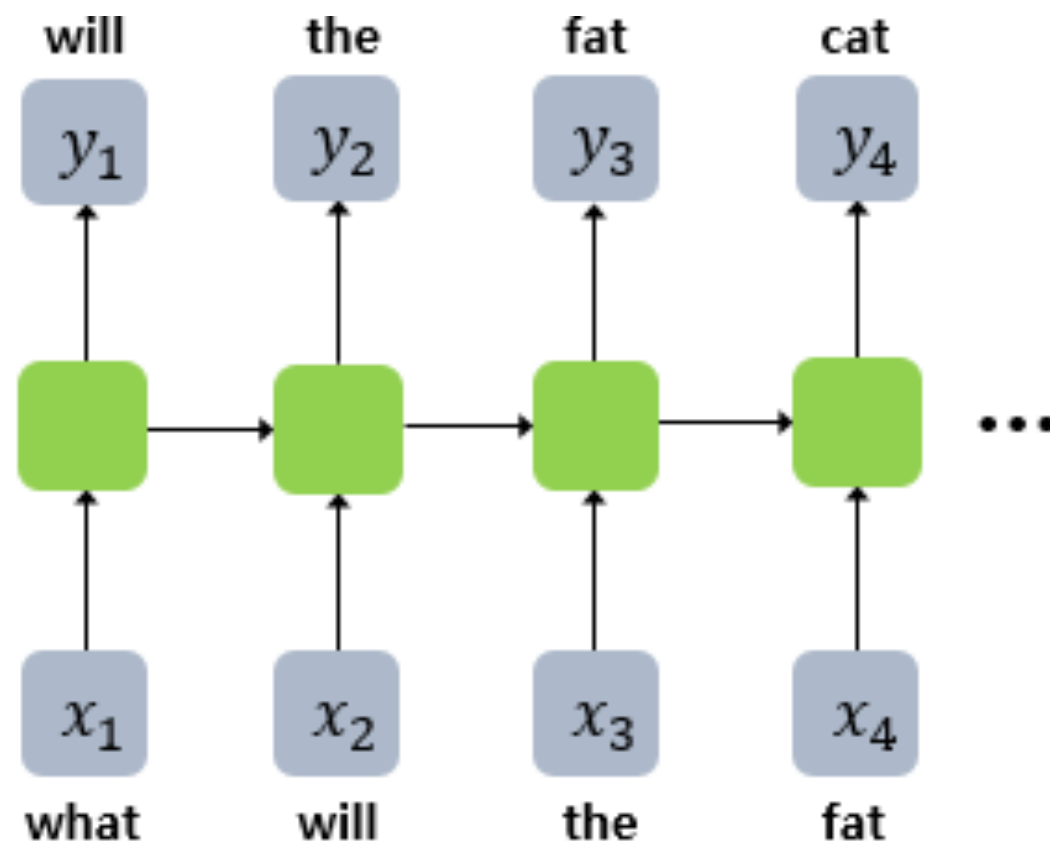
# RNNLM : RNN 언어모델

- RNN을 이용해 구현한 언어 모델
- n-gram, NNLM의 한계 : 고정된 개수의 단어를 입력으로 받는다
- **timestep**의 개념이 도입된 RNN 언어모델은 입력의 길이를 고정할 필요가 없다.



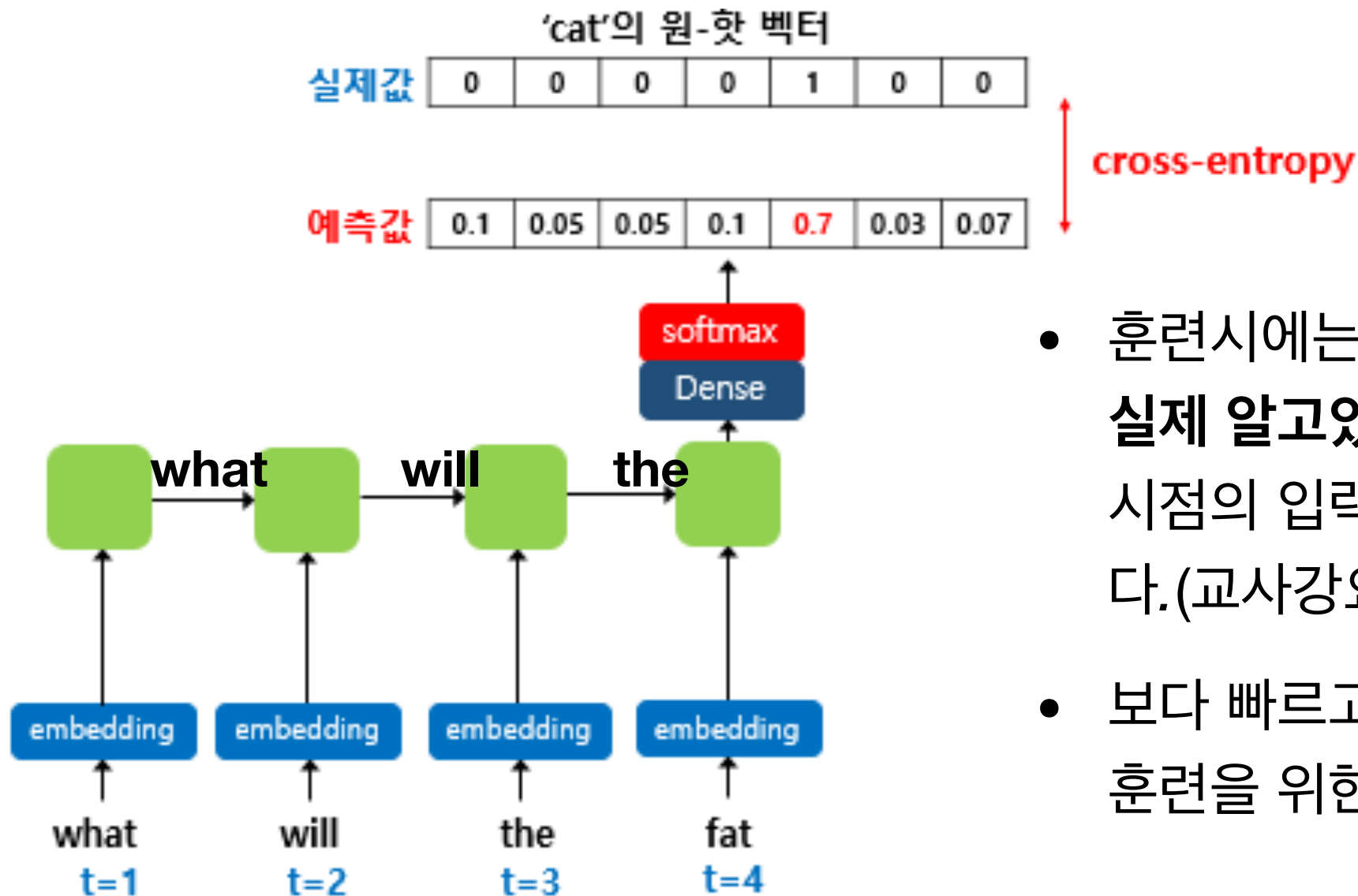
# RNNLM의 사용

예문 : What will the fat cat sit on



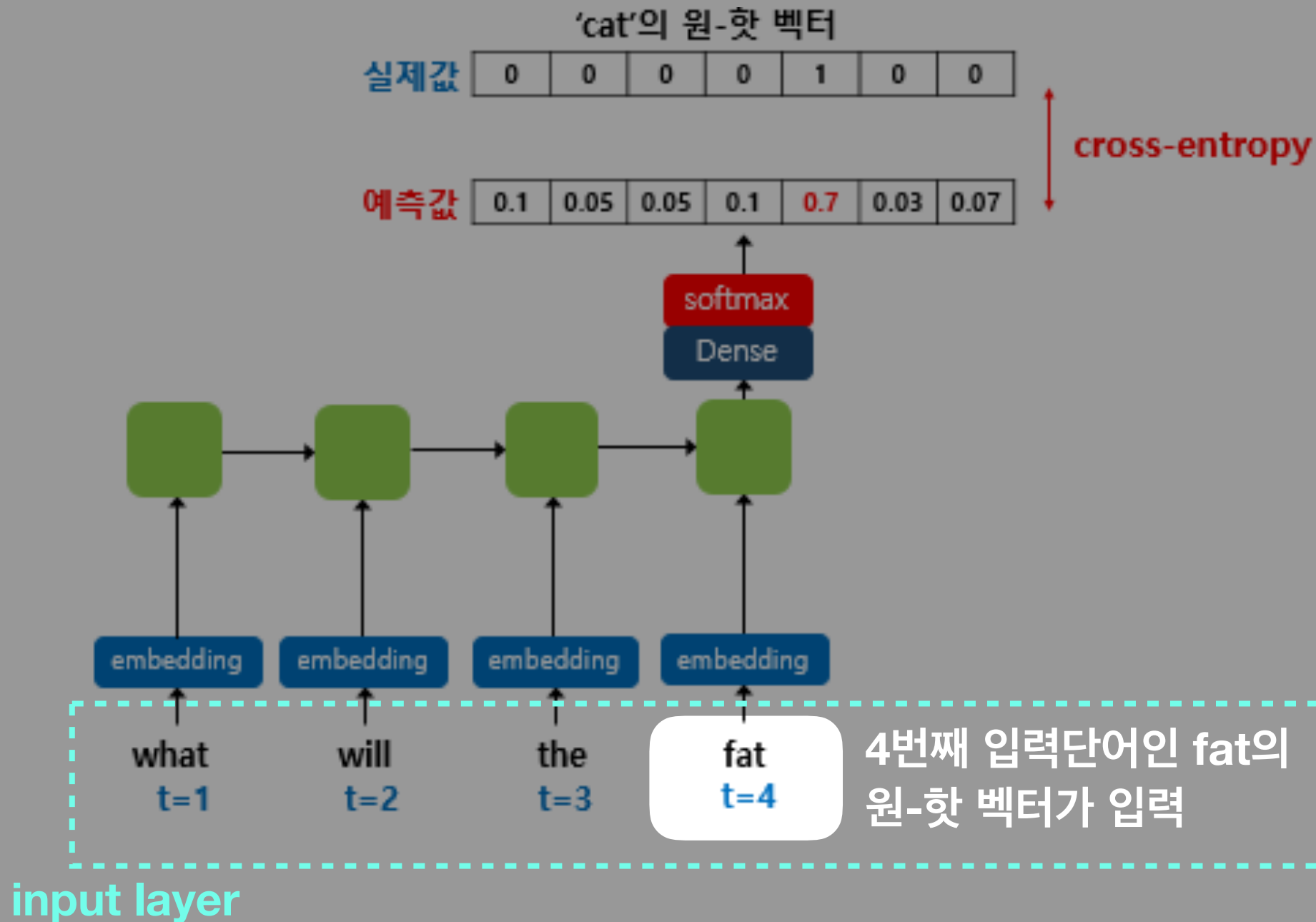
- 주어진 단어 시퀀스로부터 다음 단어를 예측하는 모델.
- **이전 시점의 출력**(예측값)이 **현재 시점의 입력**이 된다.
- RNNLM은 what을 입력받으면, will을 예측하고 이 will은 다음 시점의 입력이 되어 the를 예측한다.
- 이것이 반복되어 네번째 시점의 **cat**은 앞서 나온 **what, will, the, fat**이라는 시퀀스로 인해 결정된 단어가 된다.

# RNNLM의 훈련



- 훈련시에는 예측값이 아닌 실제 알고있는 정답을  $t+1$  시점의 입력으로 사용한다.(교사강요)
- 보다 빠르고 효과적인 모델 훈련을 위한 기법

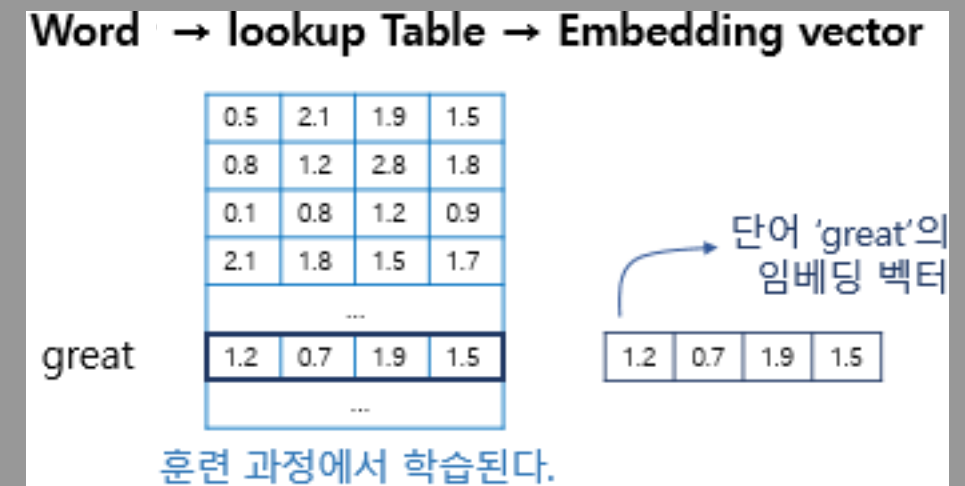
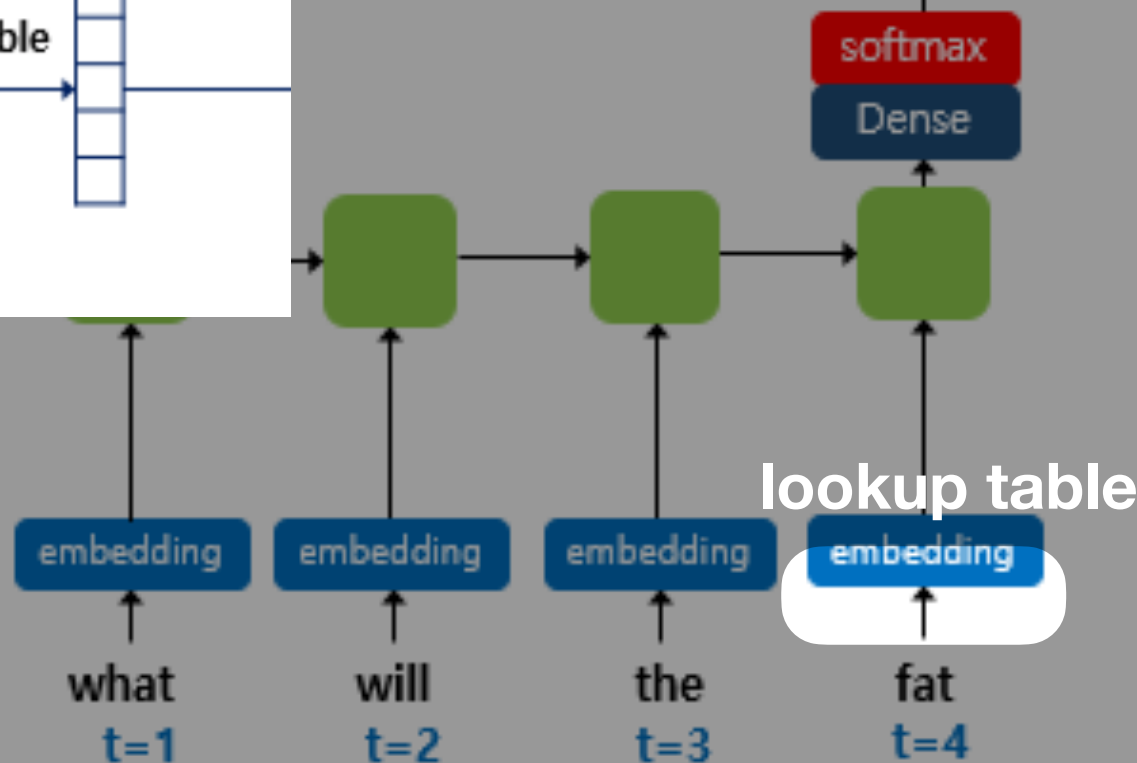
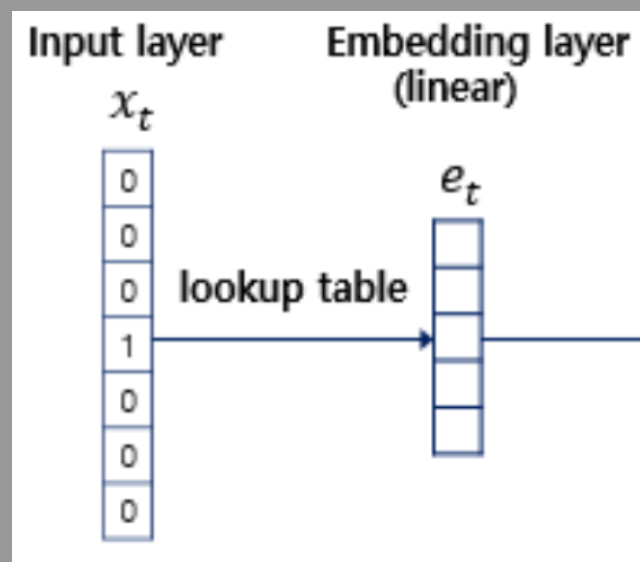
# RNNLM의 구조





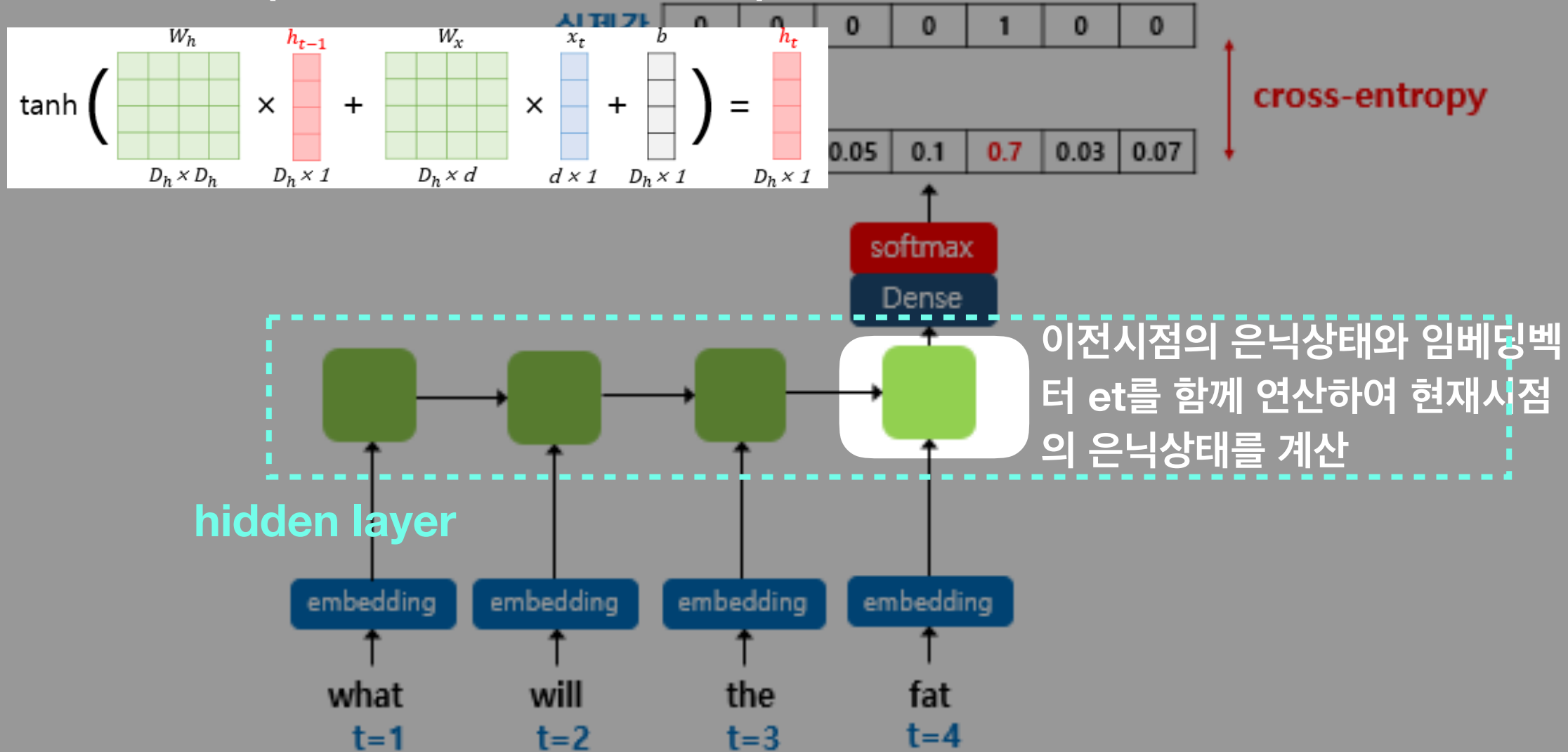
# RNNLM의 구조

\* 임베딩층 :  $e_t = \text{lookup}(x_t)$



# RNNLM의 구조

\* 은닉층 :  $\tanh(W_x * e_t + W_h * h_{t-1} + b)$

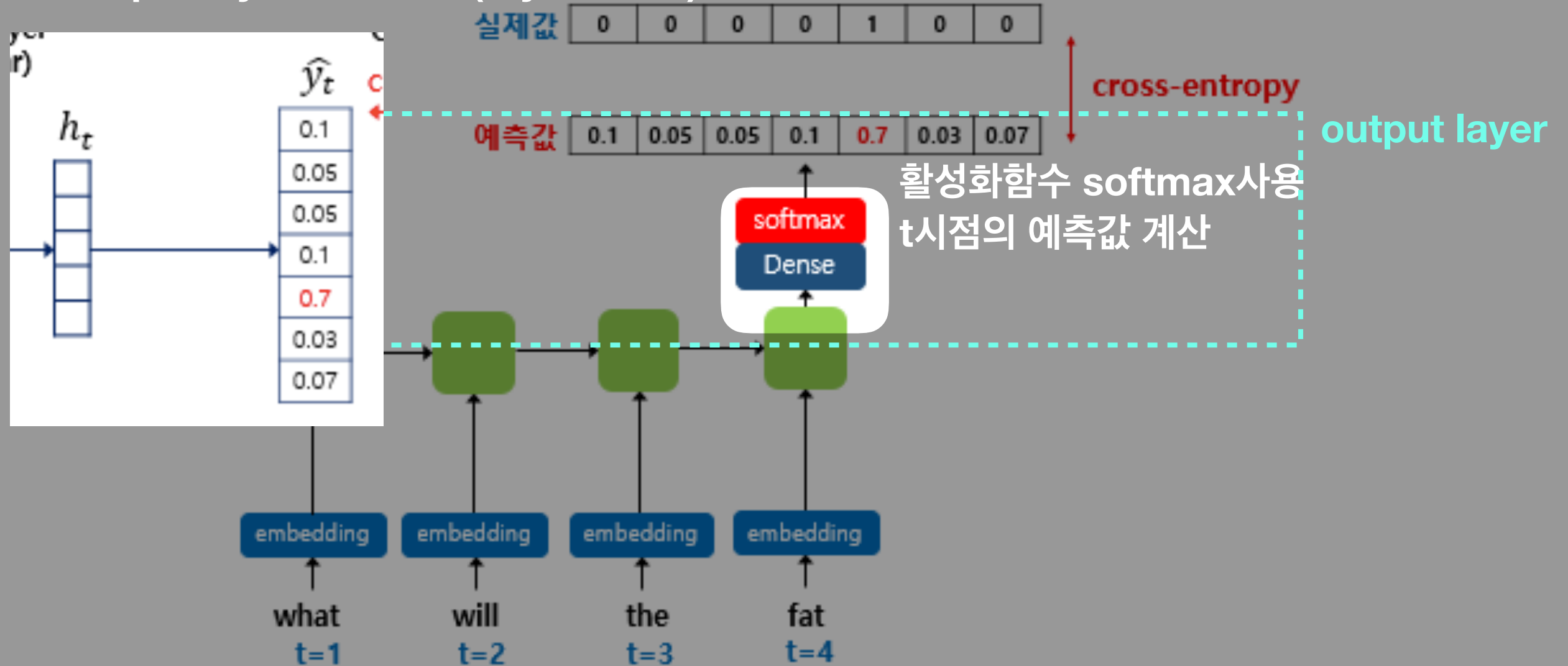


# RNNLM의 구조

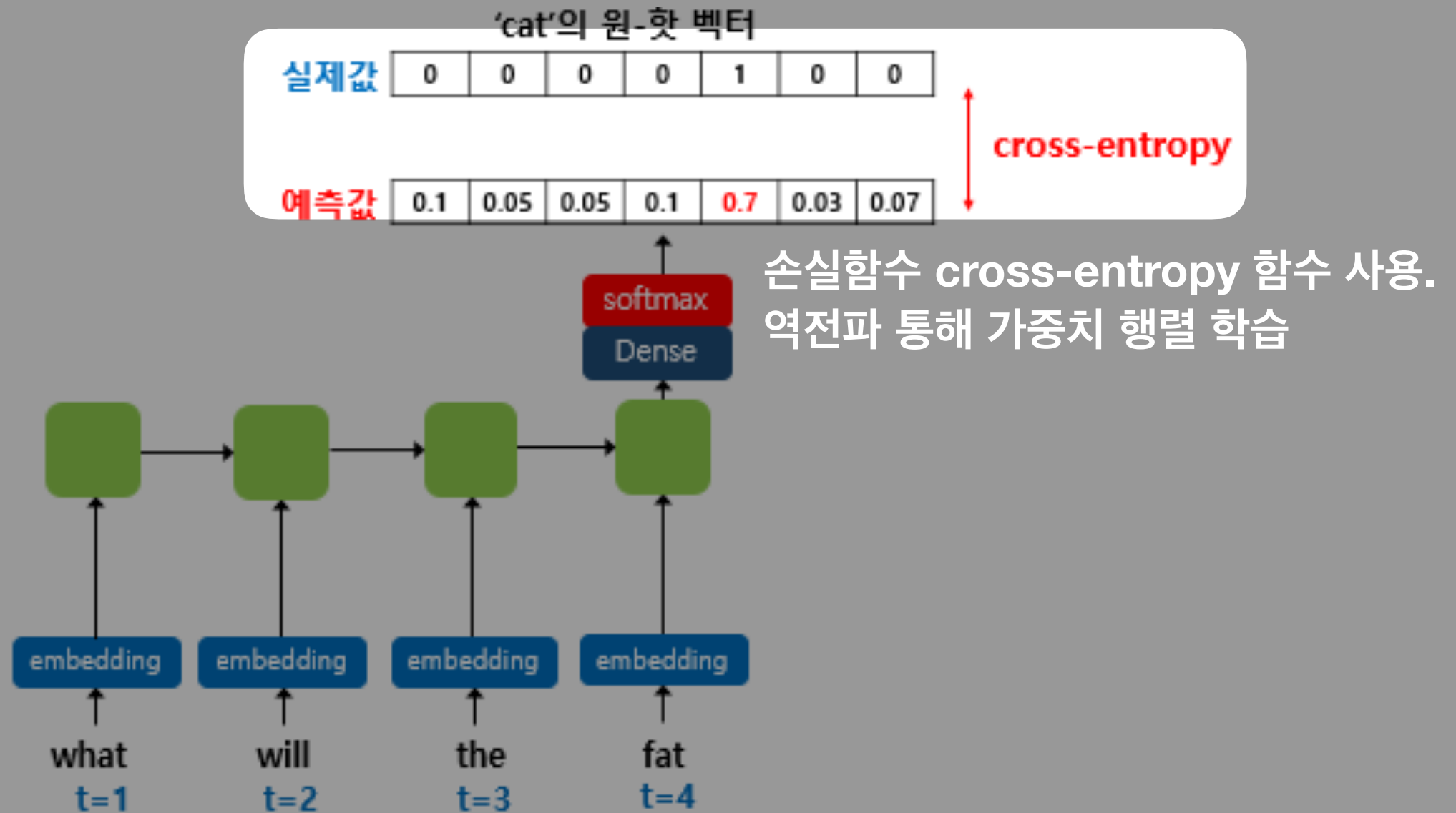


# RNNLM의 구조

\* 출력층 :  $\text{pred\_yt} = \text{softmax}(W_y * h_t + b)$



# RNNLM의 구조



# RNNLM의 실습

순환 신경망을 활용한 문자열 생성

[https://www.tensorflow.org/tutorials/text/text\\_generation](https://www.tensorflow.org/tutorials/text/text_generation)