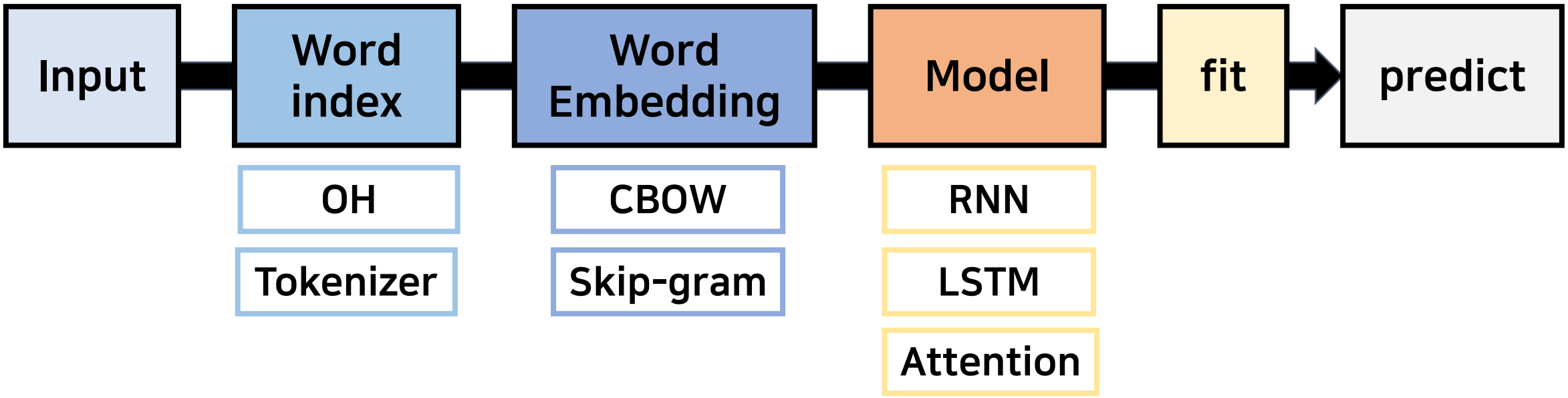
Contents

7장 자연어 처리

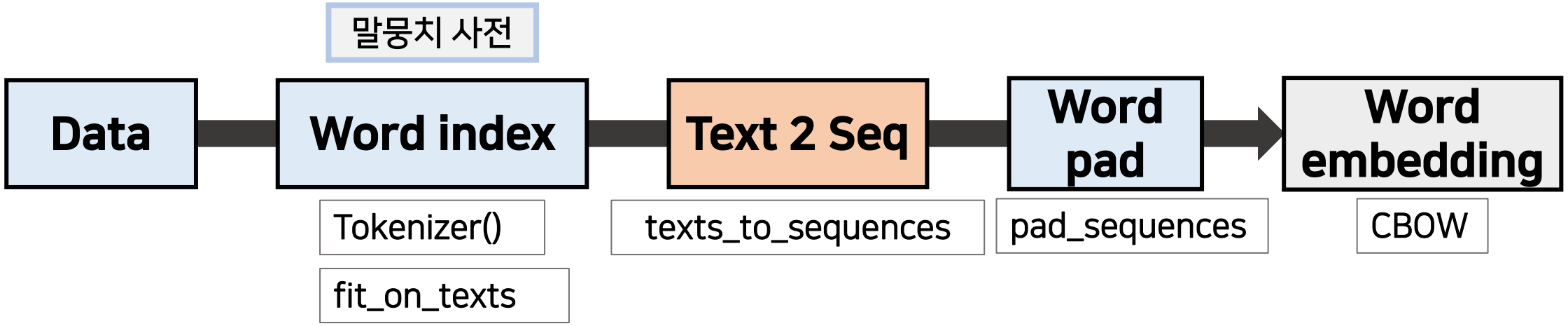
자연어 처리 과정


Input
Word index
Word Embedding
Model
fit
predict
OH
Tokenizer
CBOW
Skip-gram
RNN
LSTM
Attention

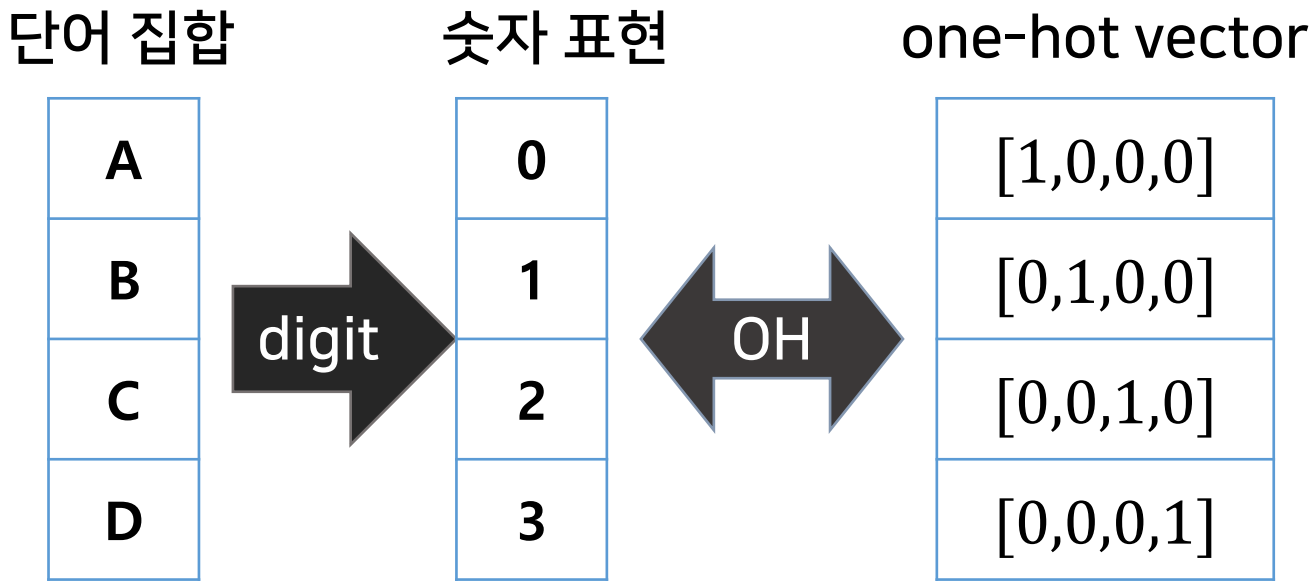
Contents

7장 자연어 처리

1. Word Embedding 과정



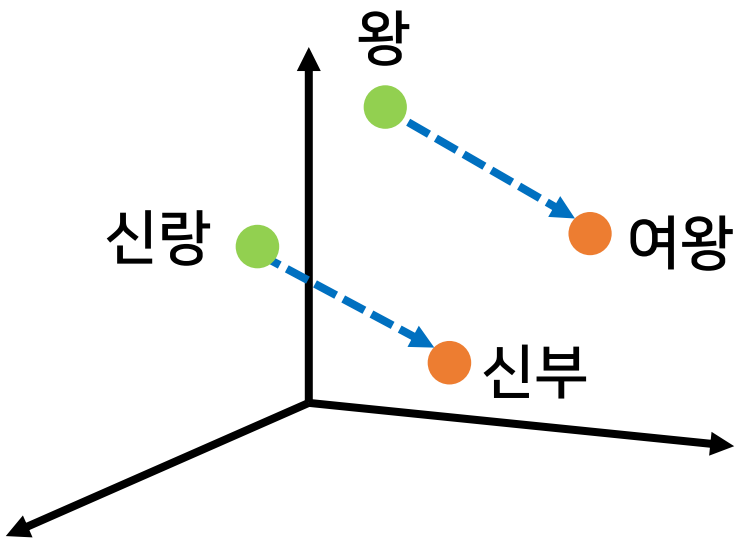
One-hot vector

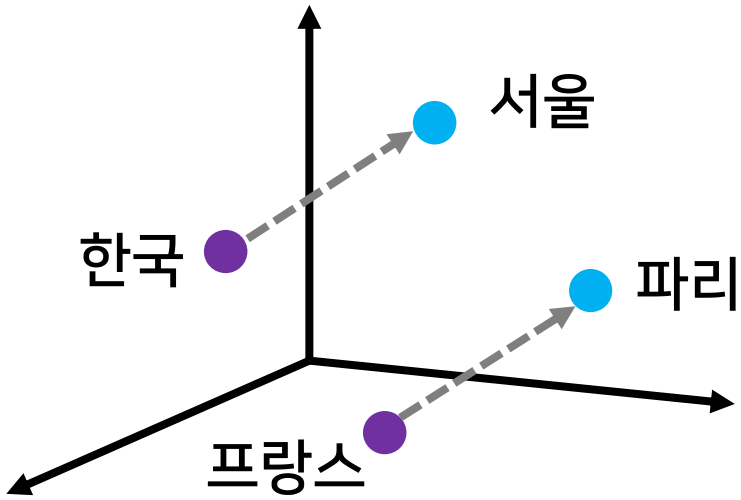


Embedding vector

	One-Hot vector	Embedding vector
차원	고차원 (단어 집합의 크기)	저차원 (128, 256 등)
표현 방식	희소 벡터	밀집 벡터
표현 방법	수동	훈련 데이터로부터 학습
값의 형식	하나의 값만 1, 나머지는 0	실수

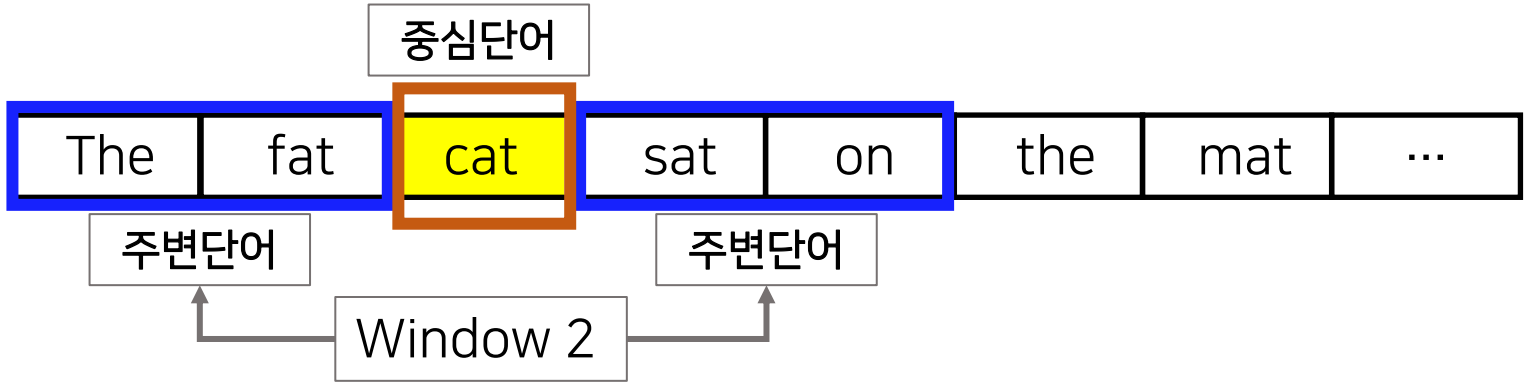
Word Embedding

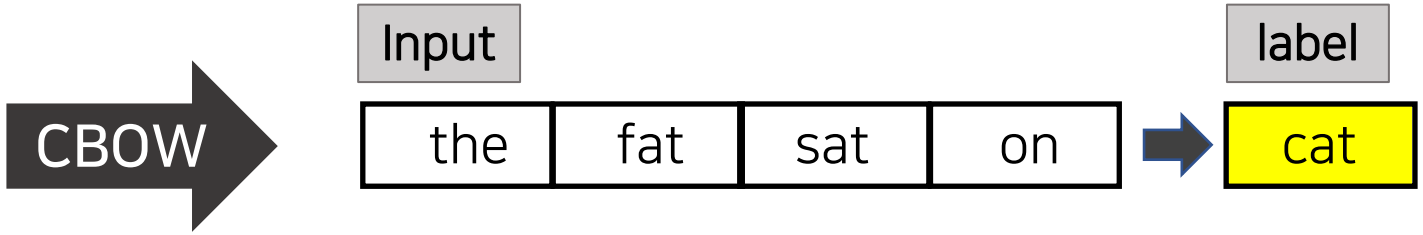




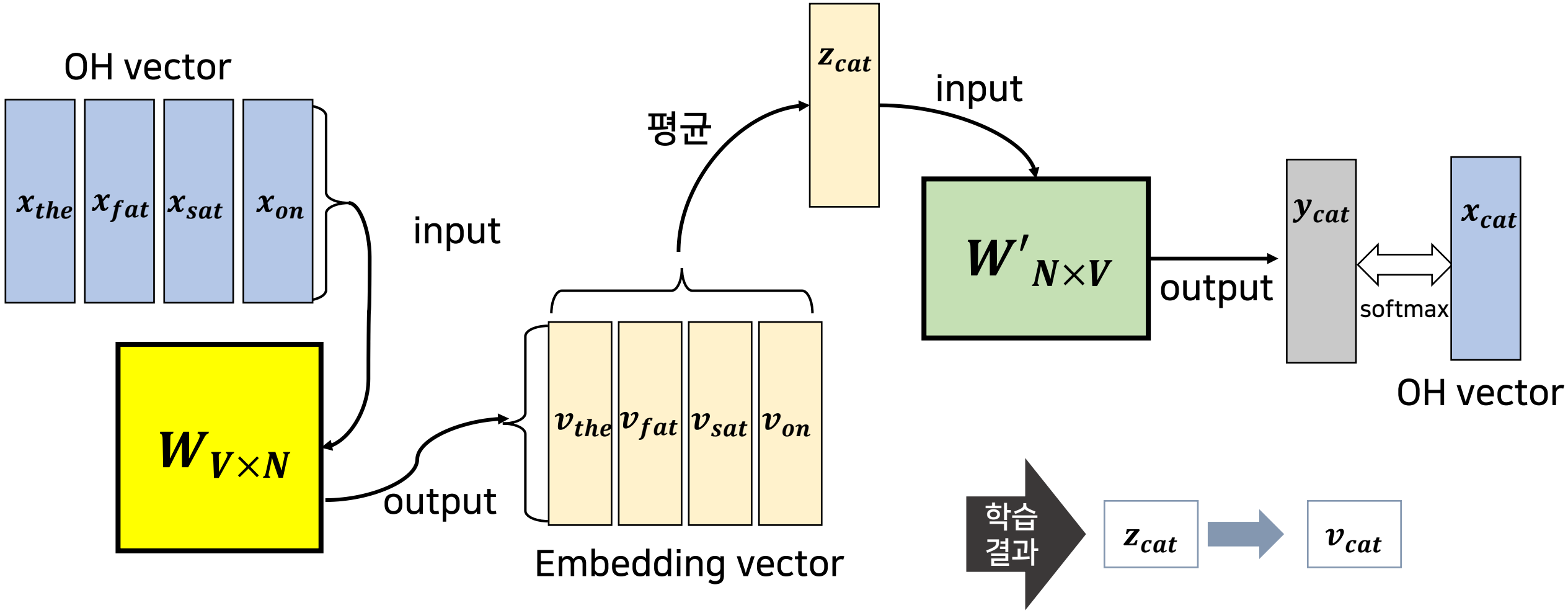
2. CBOW 원리

주변 단어로 중심 단어 예측



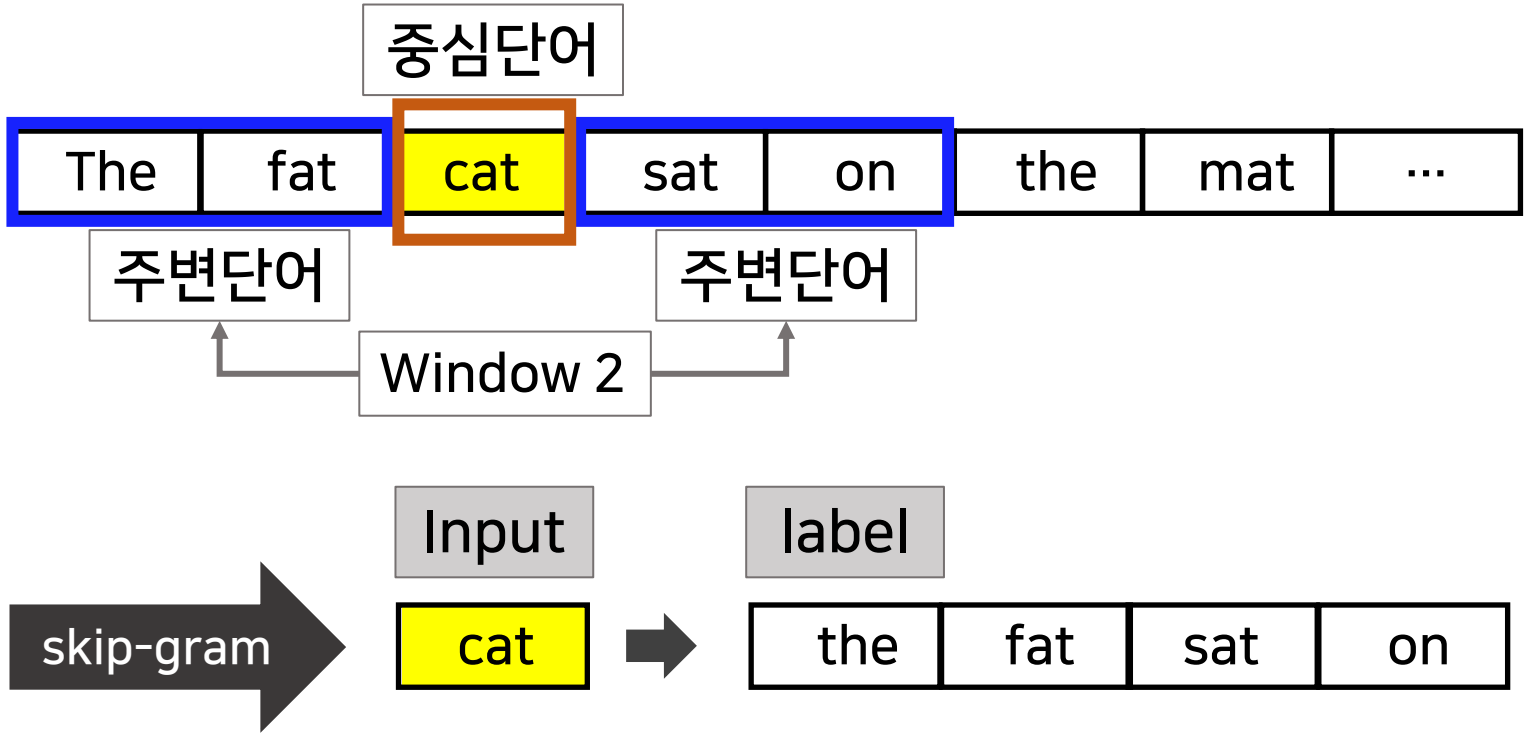


CBOW 모델

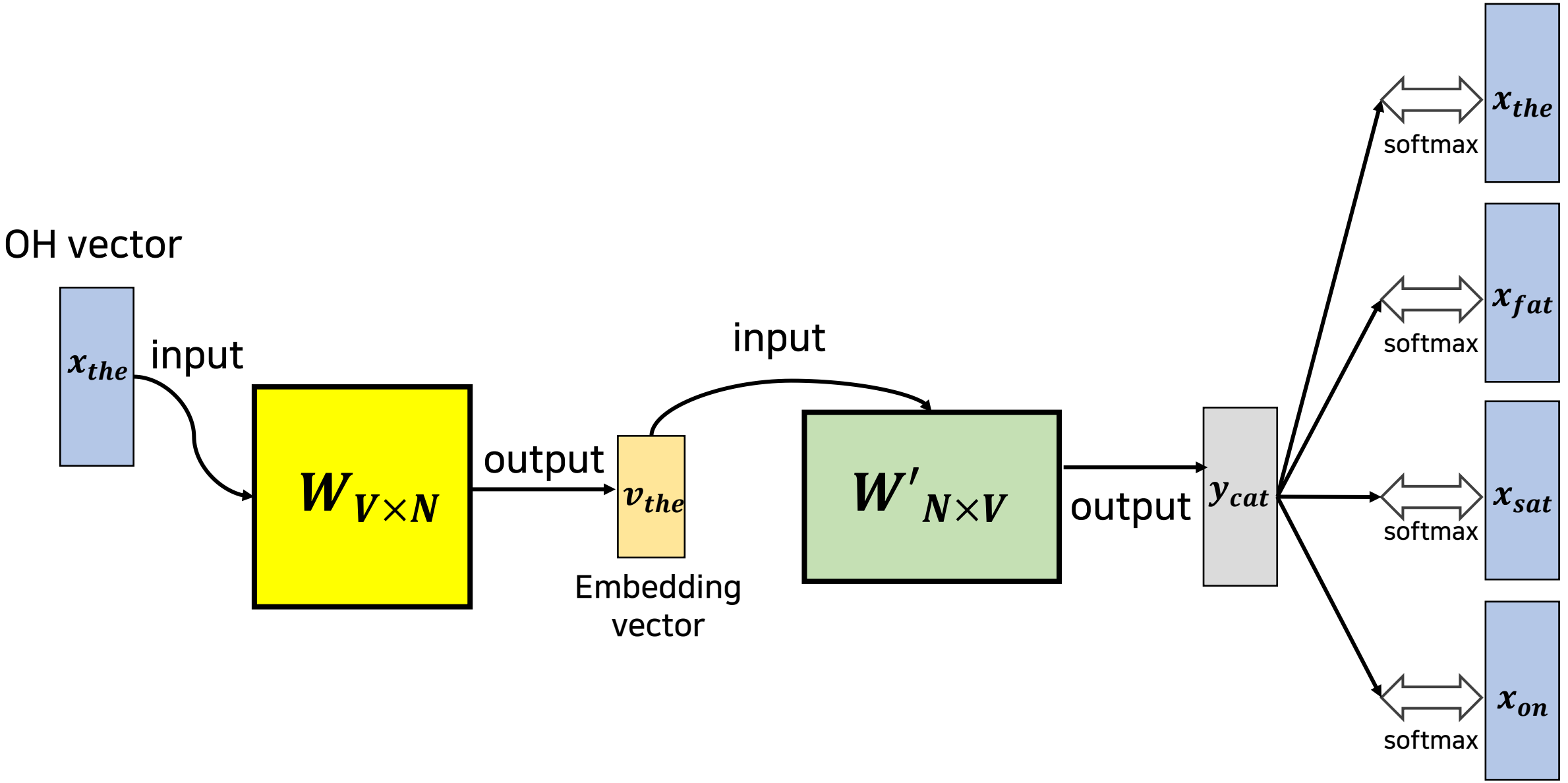

OH vector
x_{the} x_{fat} x_{sat} x_{on}
input
$W_{V\times N}$
output
v_{the} v_{fat} v_{sat} v_{on}
Embedding vector
평균
z_{cat}
input
$W'_{N\times V}$
output
y_{cat}
softmax
x_{cat}
OH vector
학습 결과
z_{cat}
v_{cat}

3. Skip-gram 원리

중심 단어로 주변 단어 예측



skip-gram 과정


OH vector
x_{the}
input
$W_{V\times N}$
output
v_{the}
Embedding vector
input
$W'_{N\times V}$
output
y_{cat}
softmax
x_{the}
softmax
x_{fat}
softmax
x_{sat}
softmax
x_{on}

Contents

7장 자연어 처리

수열과 점화식

수열 sequence

$$\{a_n\}_{n=0}^{\infty}$$

$$a_0, a_1, a_2, \cdots, a_n, a_{n+1}, \cdots$$

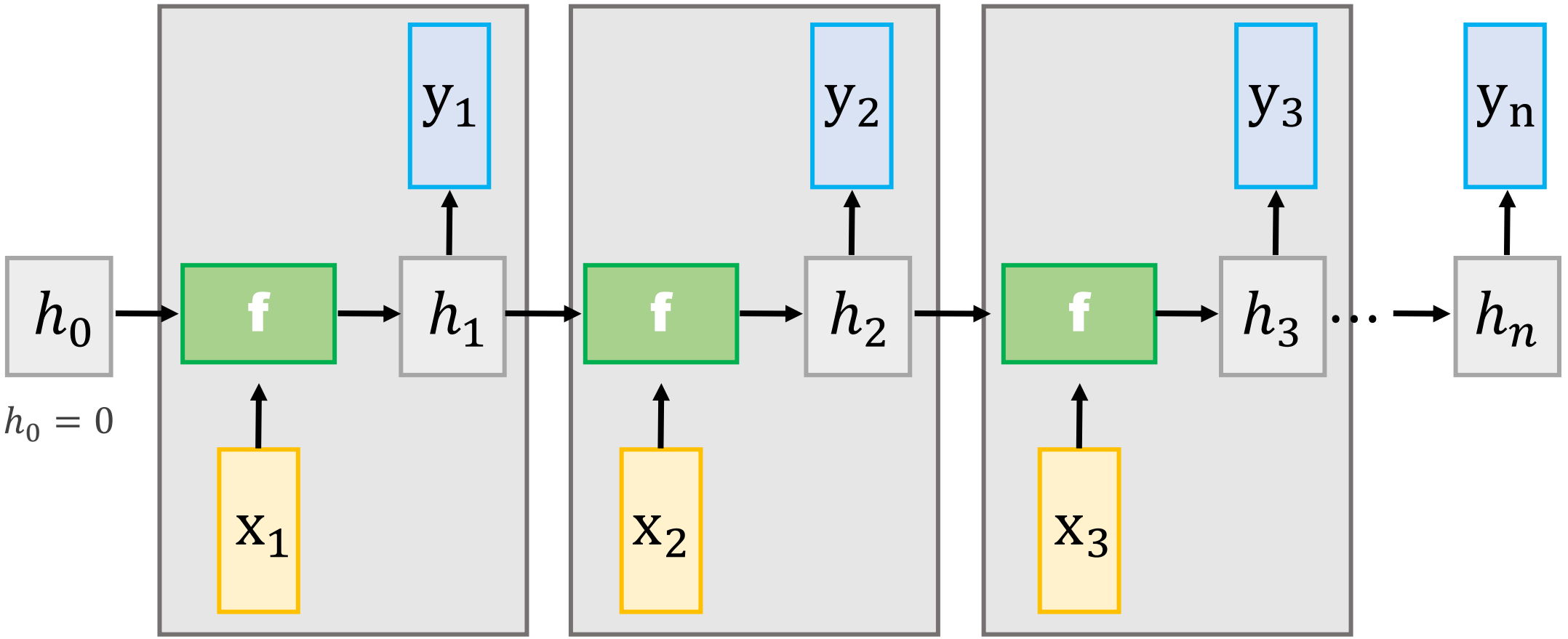
점화식 recurrence relation

$$h_{n+1} = f(h_n, h_{n-1})$$

$$a_{n+1} = a_n + a_{n-1} \Rightarrow a_n?$$

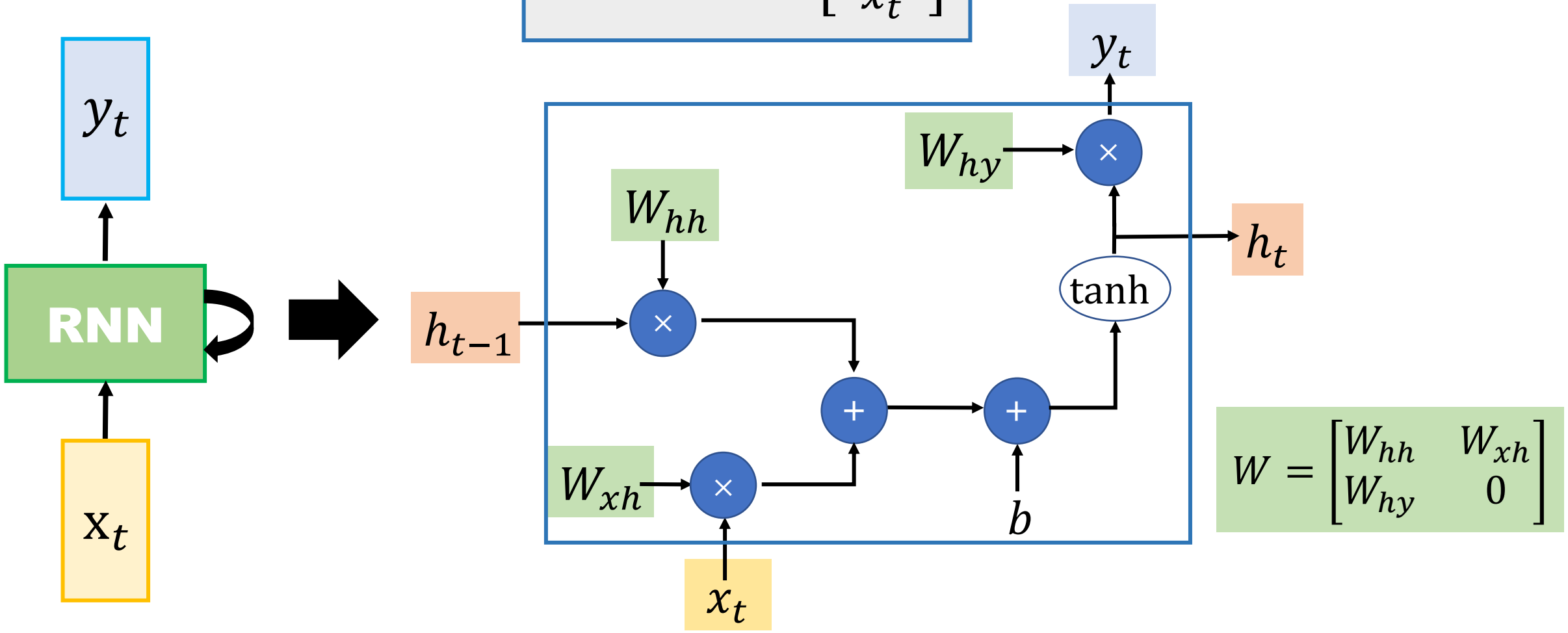
$$b_{n+1} = 2b_n - b_{n-1} \Rightarrow b_n?$$

1. RNN 모델



SimpleRNN 모델

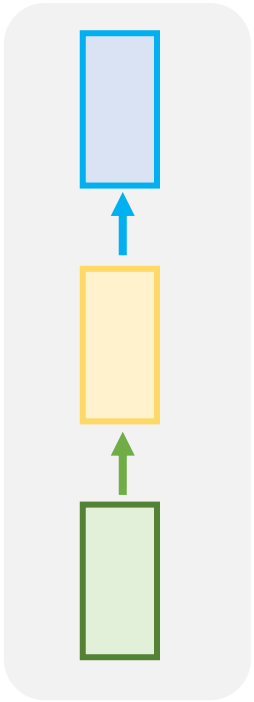
$$h_t = \tanh W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$



$$W = \begin{bmatrix} W_{hh} & W_{xh} \\ W_{hy} & 0 \end{bmatrix}$$

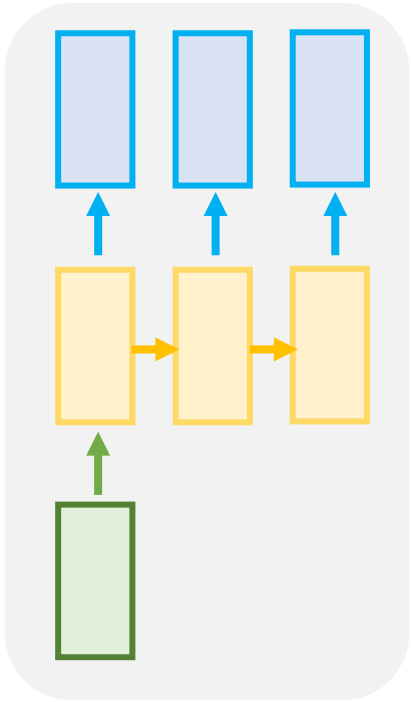
다양한 RNN 모델

One to one



Simple Neural Network

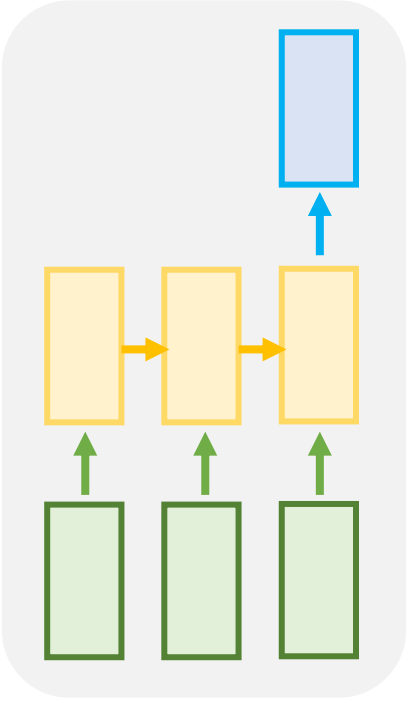
One to many



이미지 자막

이미지 → 문자열

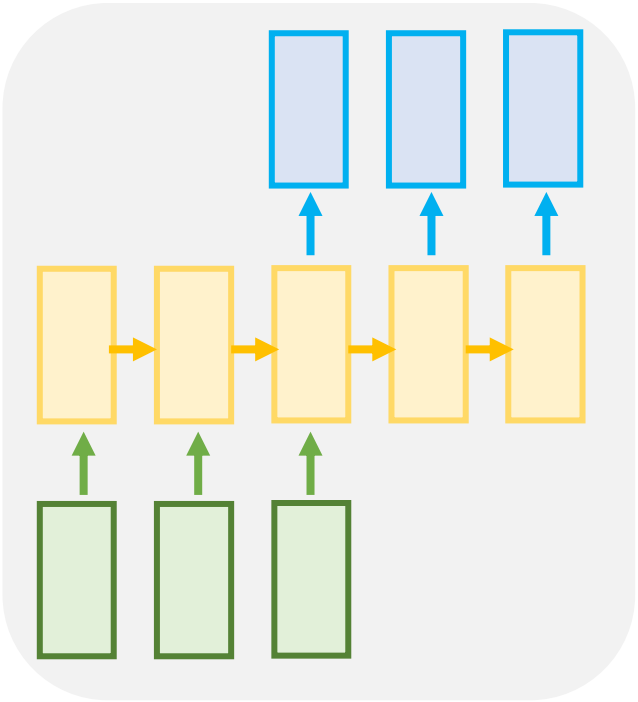
many to one



감정 분류

문자열 → 감정

many to many

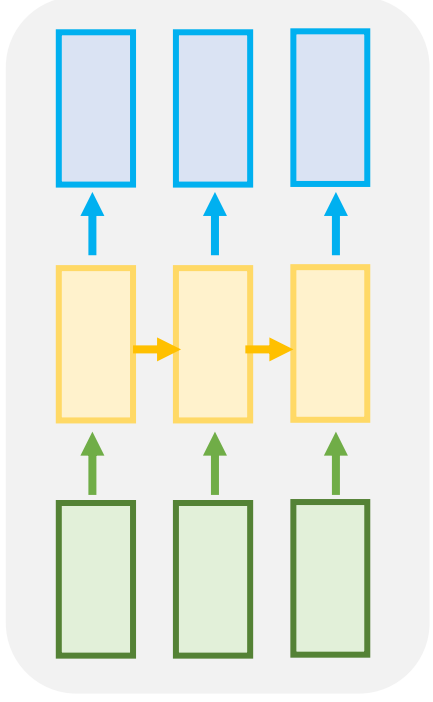


기계 번역

문자열 → 문자열

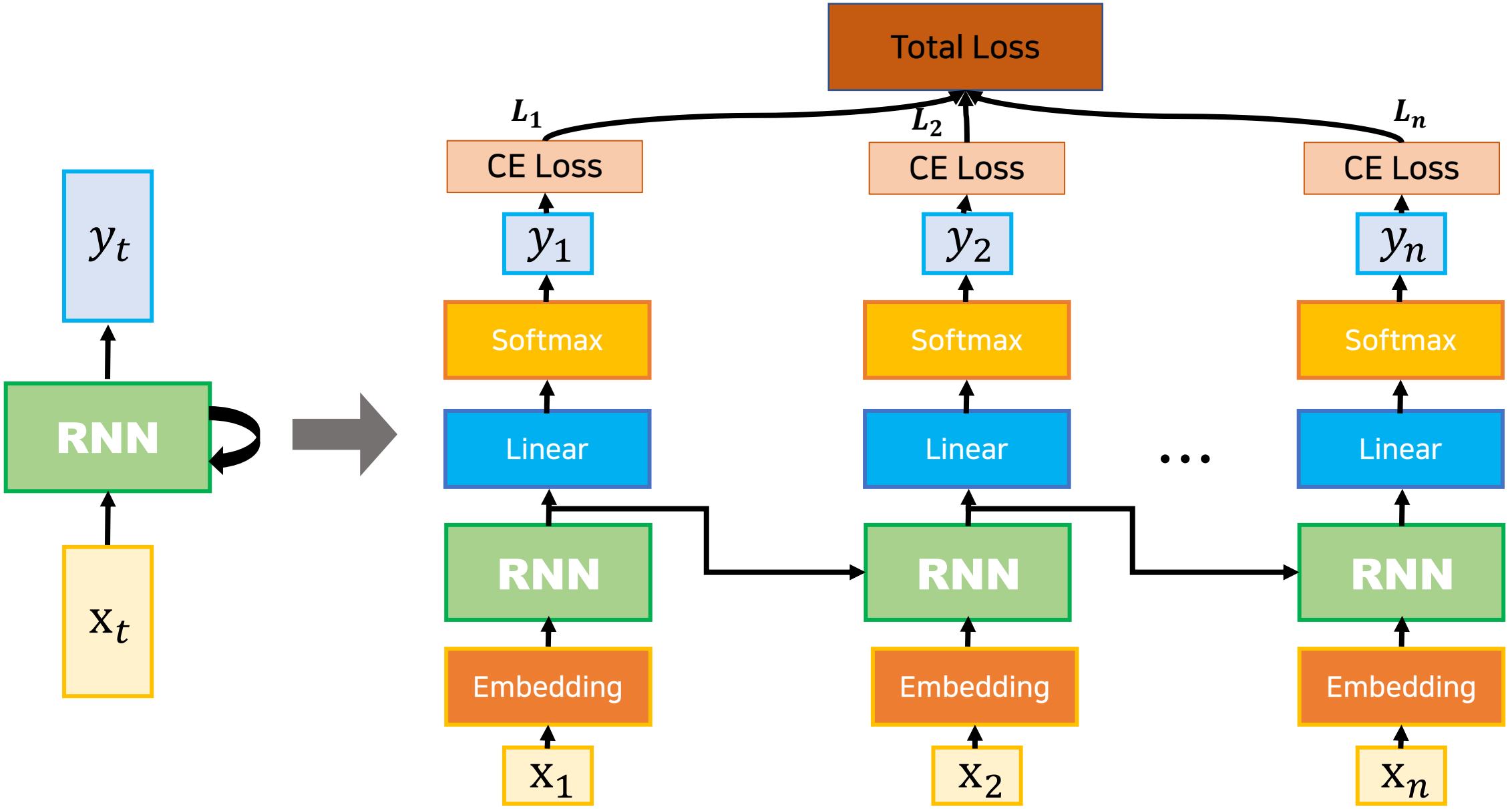
seq2seq

many to many

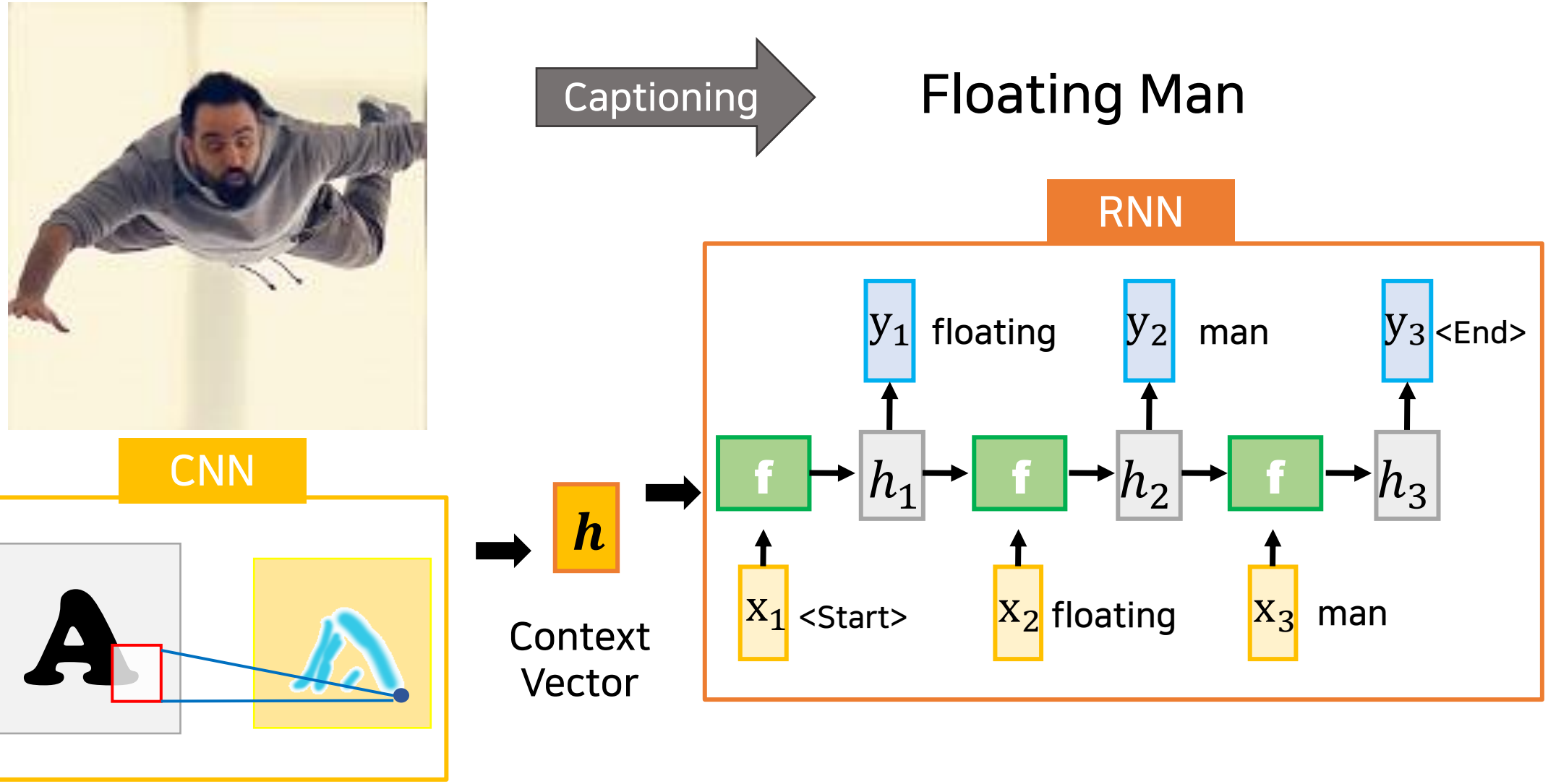


프레임 레벨
비디오 분류

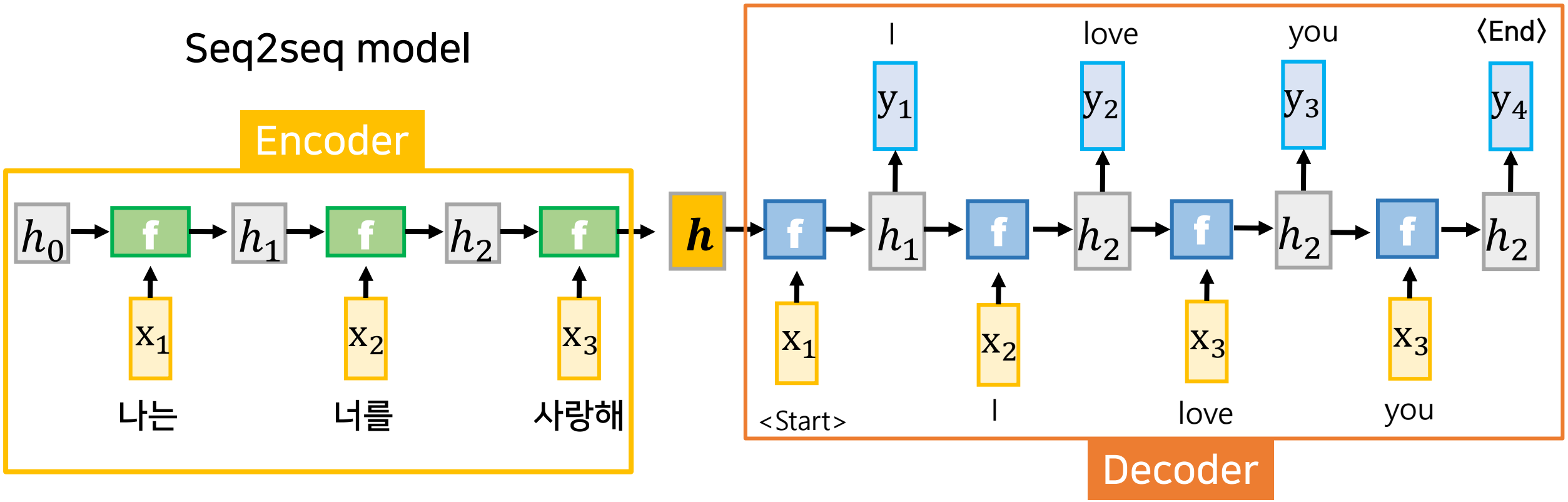
RNN 모델과 손실 계산


y_t
RNN
x_t
Total Loss
L_1
L_2
L_n
CE Loss
CE Loss
CE Loss
y_1
y_2
y_n
Softmax
Softmax
Softmax
Linear
Linear
Linear
...
RNN
RNN
RNN
Embedding
Embedding
Embedding
x_1
x_2
x_n

CNN+RNN : Image Captioning



RNN+RNN : Machine Translation



Sutskever et al., "Sequence to Sequence Learning with Neural Networks", NIPS 2014

2. LSTM 구조

i: input gate

셀에 정보를 저장할지 여부

f: forget gate

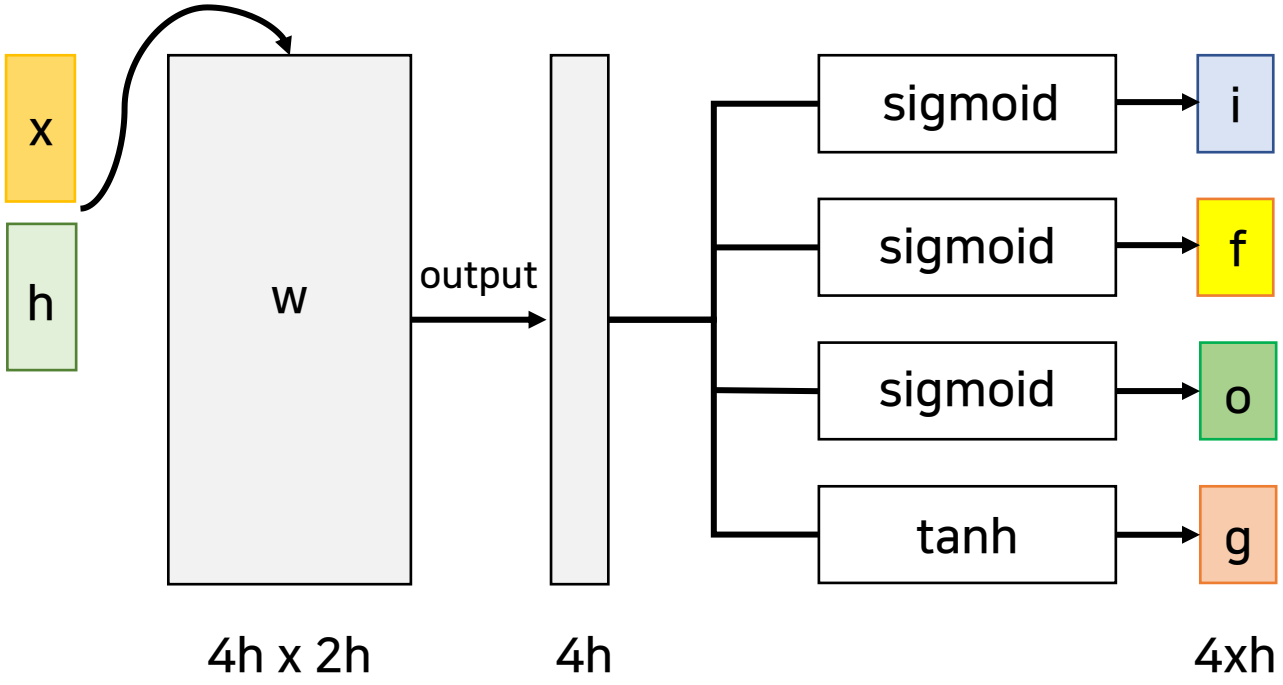
셀에 정보 반영하는 정도

o: output gate

출력값 결정

g: update gate

셀에 저장할 정보의 정도를 결정

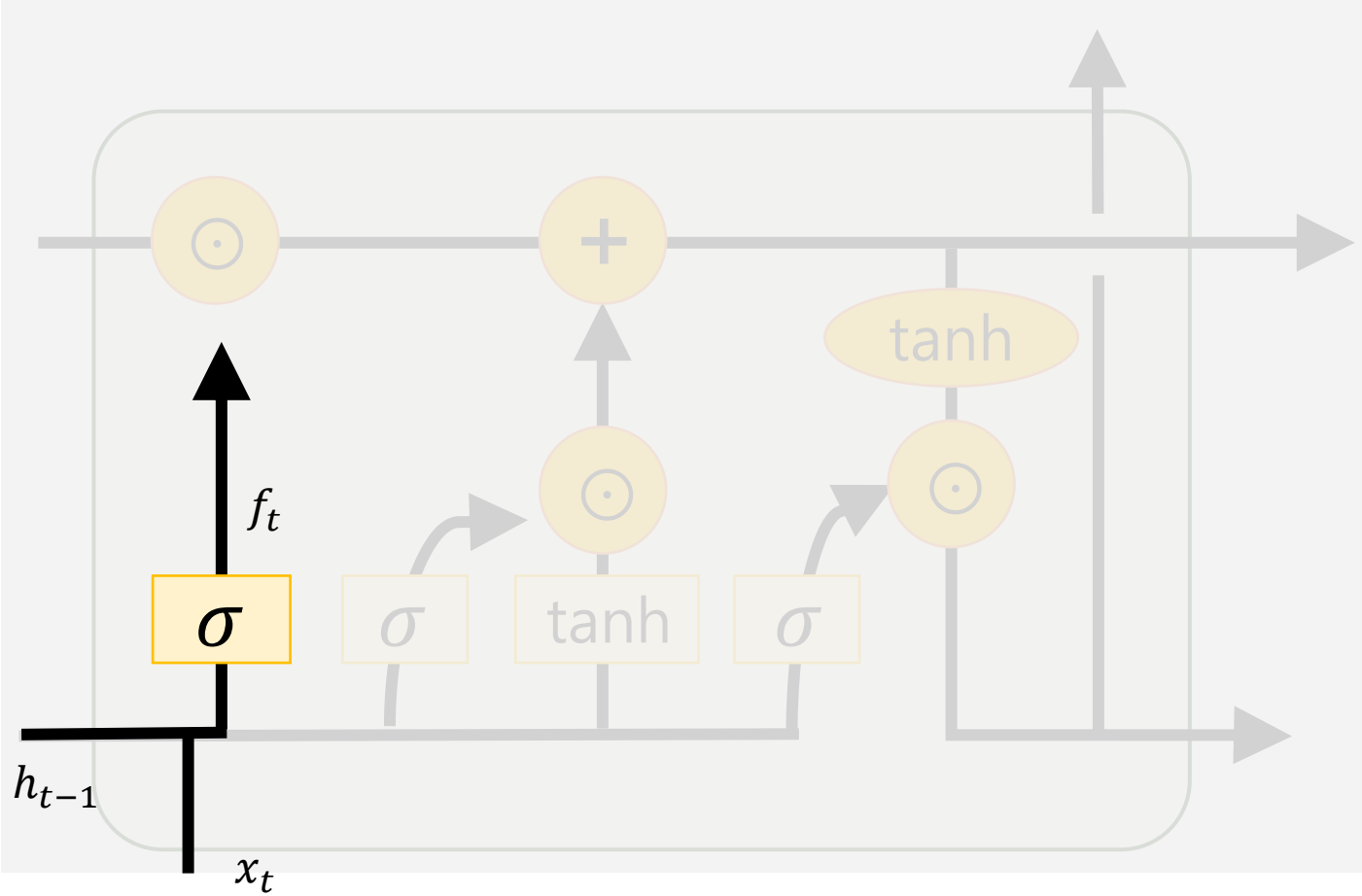


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$C_t = f \odot C_{t-1} + i \odot g$$

$$h_t = o \odot \tanh C_t$$

Forget Gate

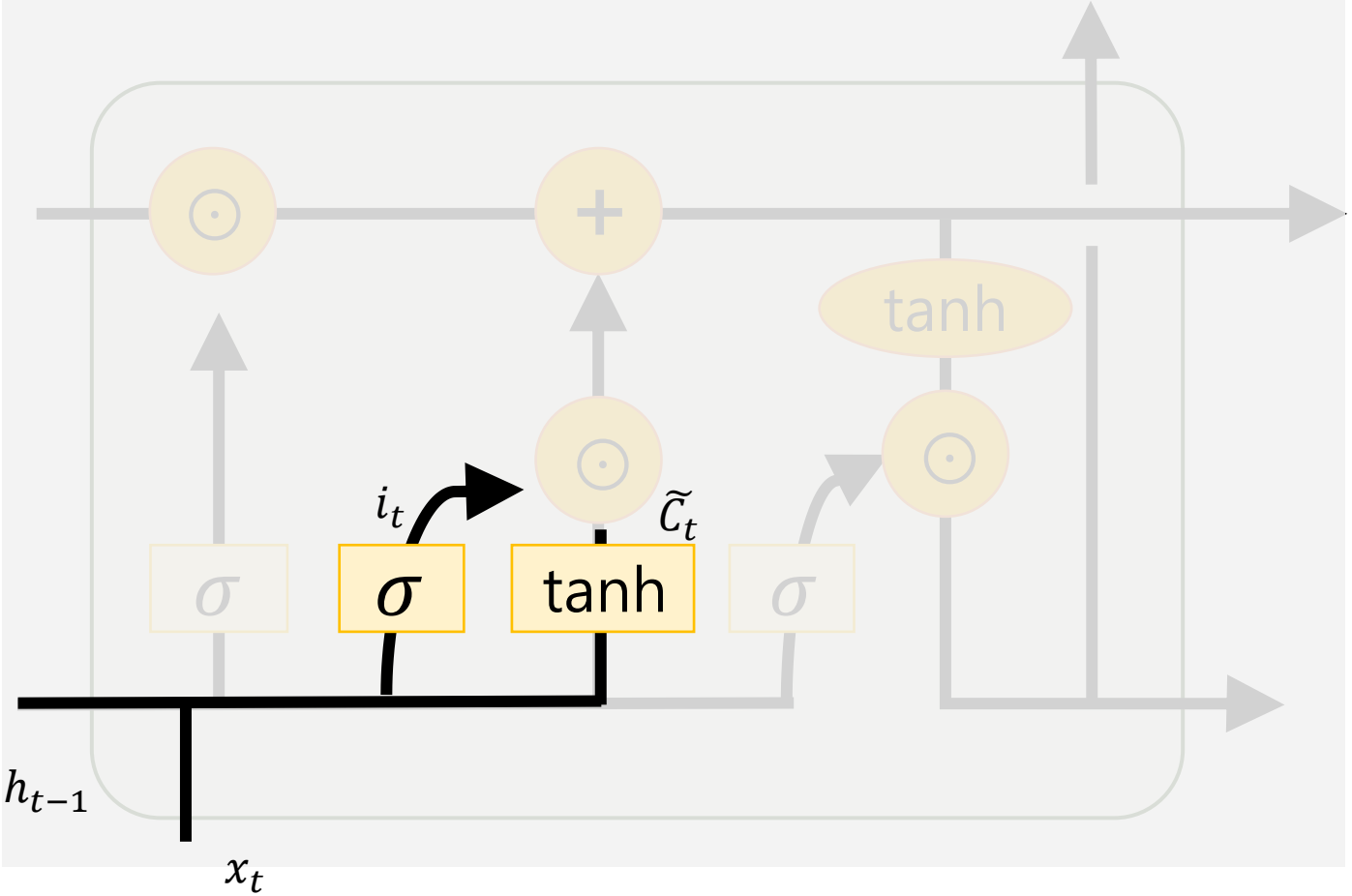


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

입력 → 활성화 → 출력

h_{t-1}, x_t → sigmoid → $0 < \sigma < 1$

Input Gate

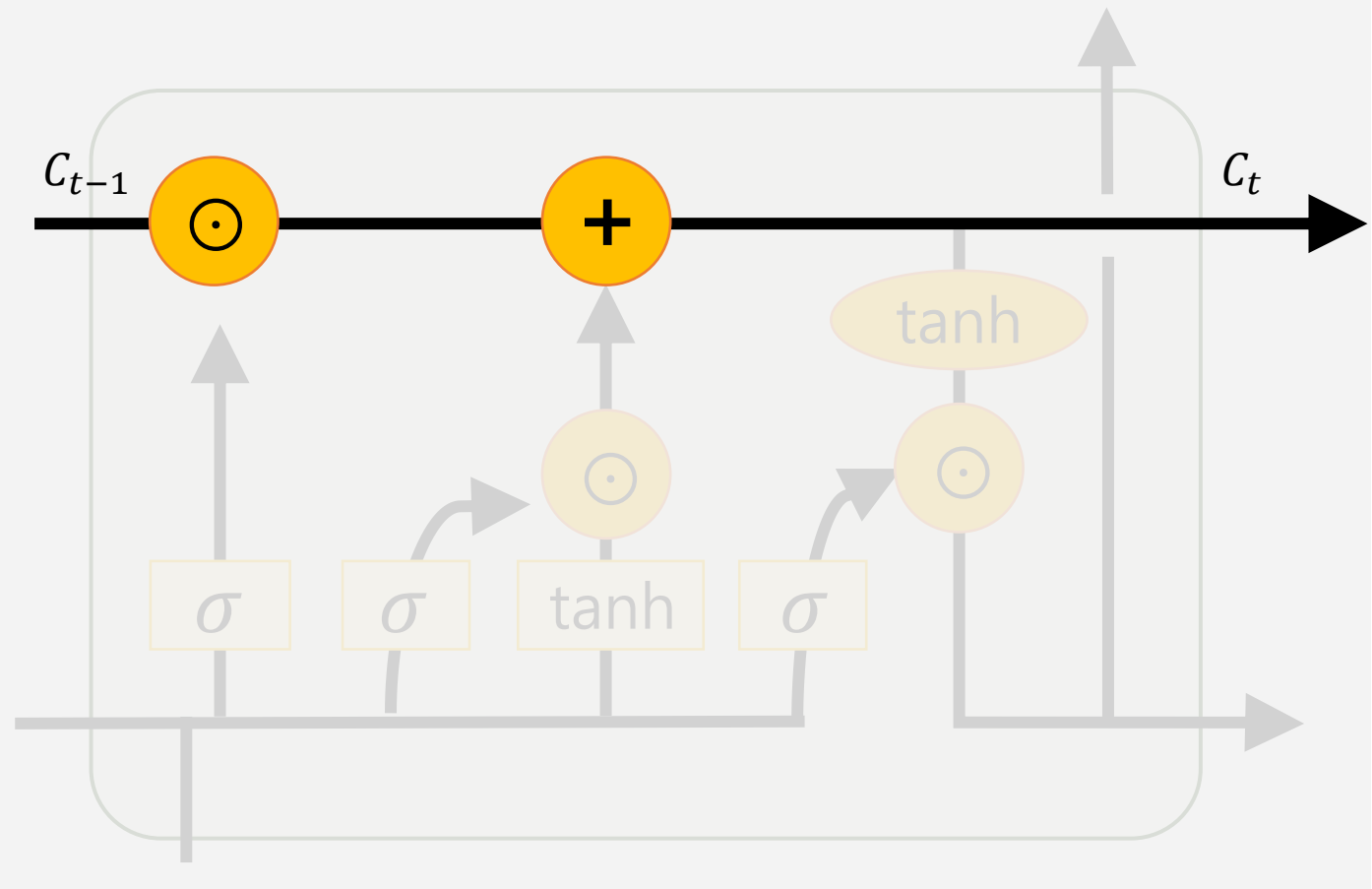


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

	입력	활성화	출력
(1)	h_{t-1}, x_t	sigmoid	$0 < \sigma < 1$
(2)	h_{t-1}, x_t	tanh	$-1 < t < 1$

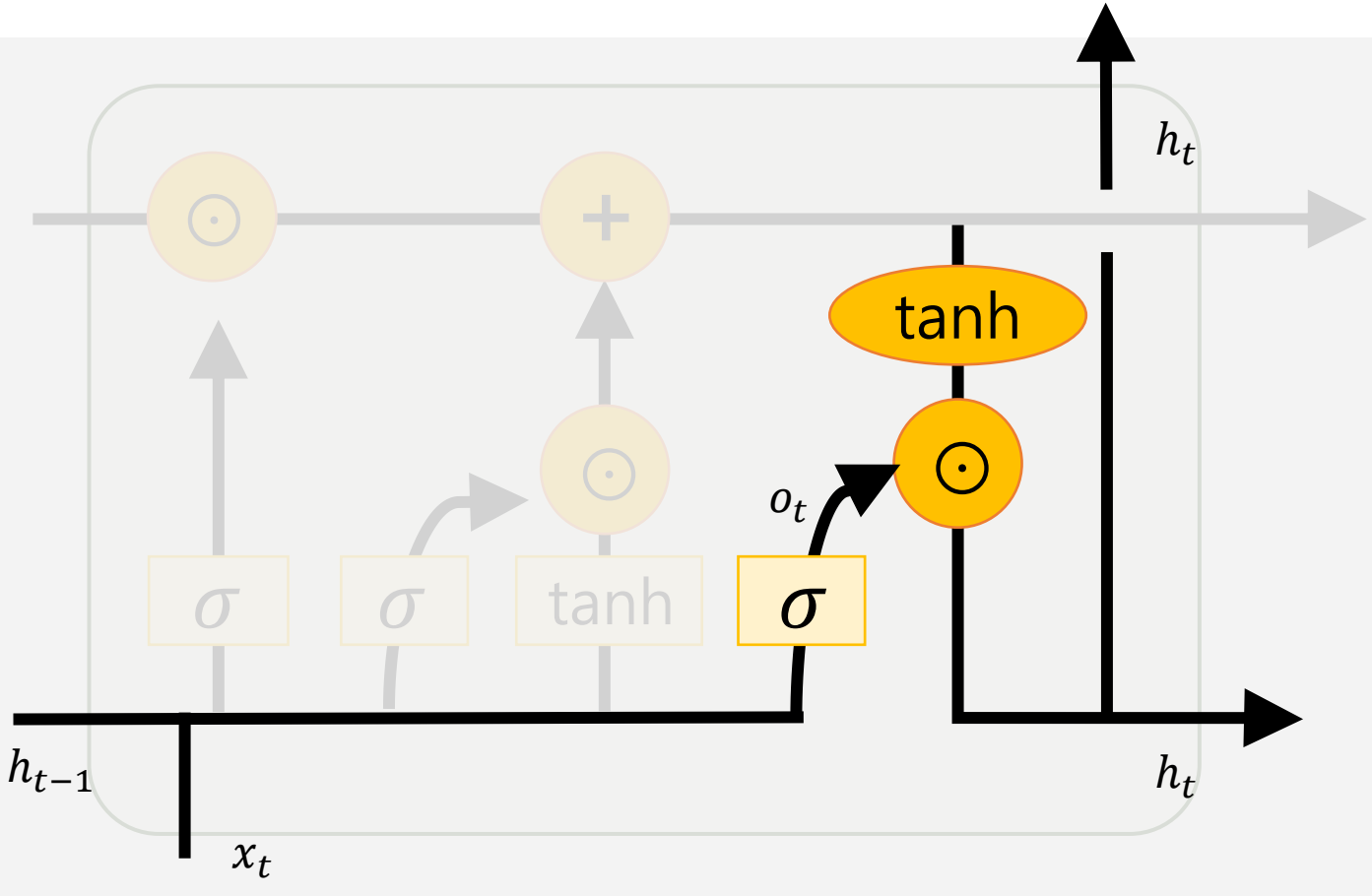
Cell State



$$C_t = f \odot C_{t-1} + i \odot g$$

입력 C_{t-1} → 연산 $\odot$ $+$ → 출력 C_t

Output Gate

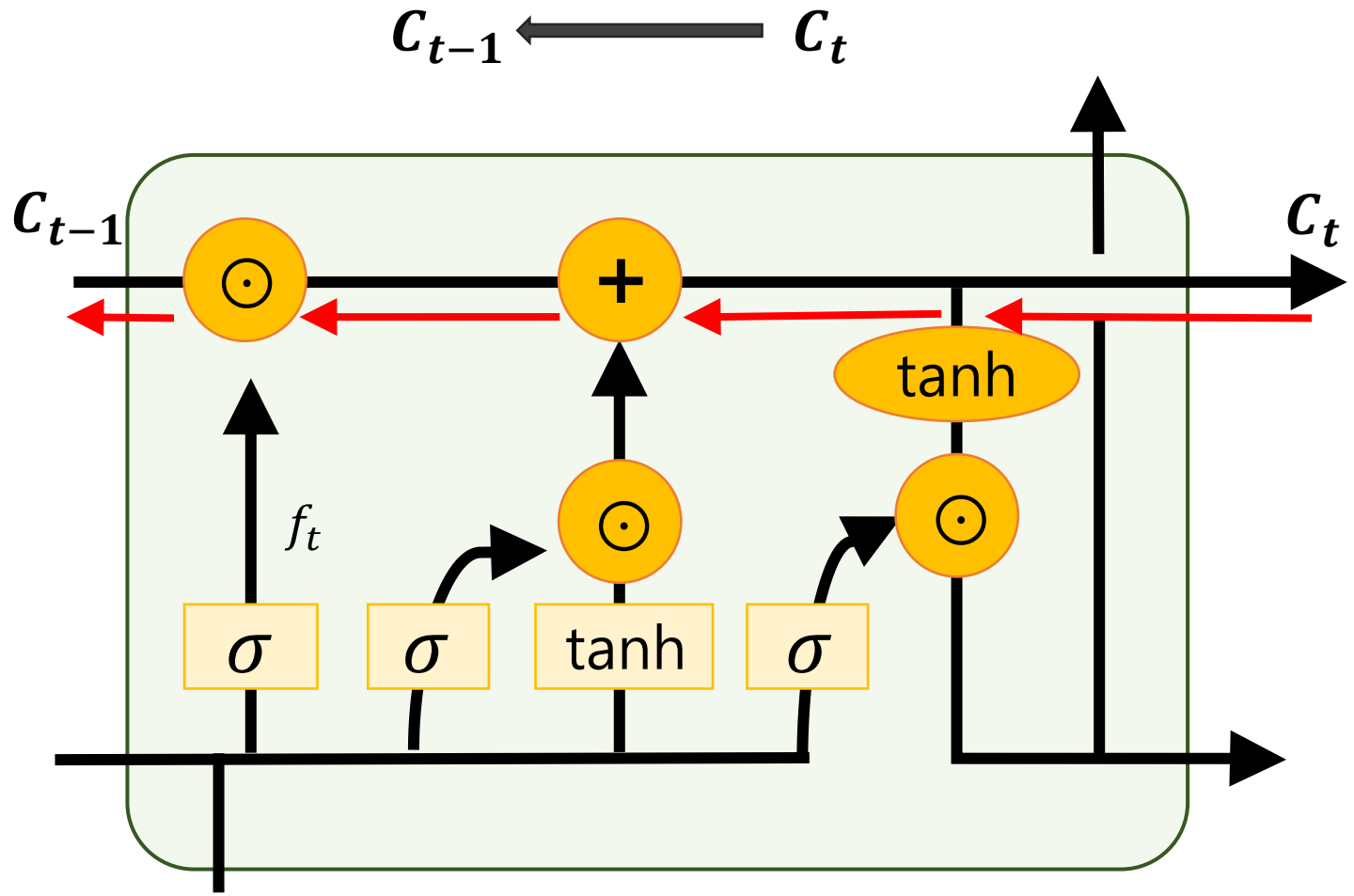


$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_0)$$

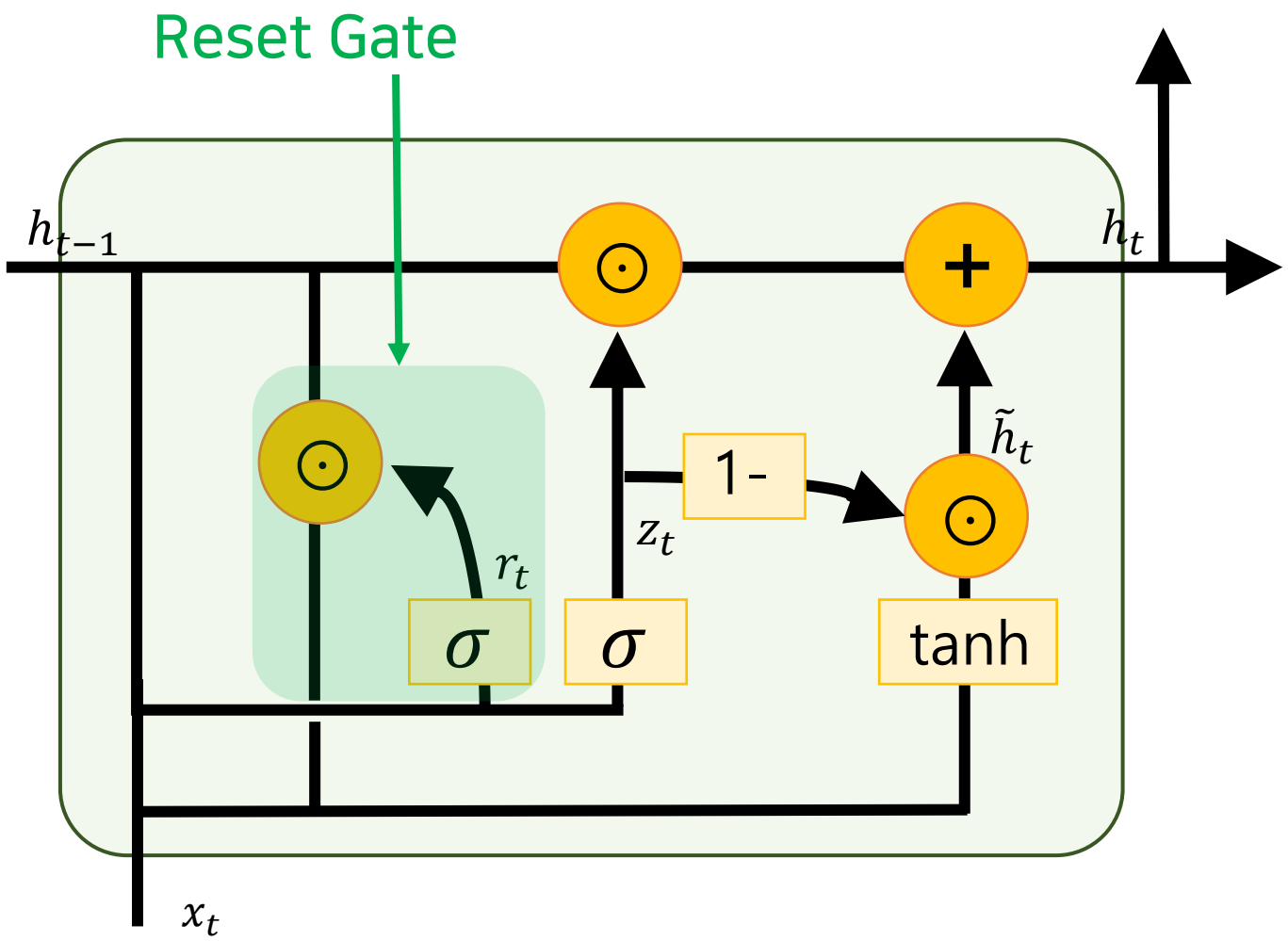
$$h_t = o_t \odot \tanh C_t$$

입력	활성화	출력
h_{t-1}, x_t	sigmoid	$0 < \sigma < 1$
C_t	tanh	h_t

LSTM 오류역전파

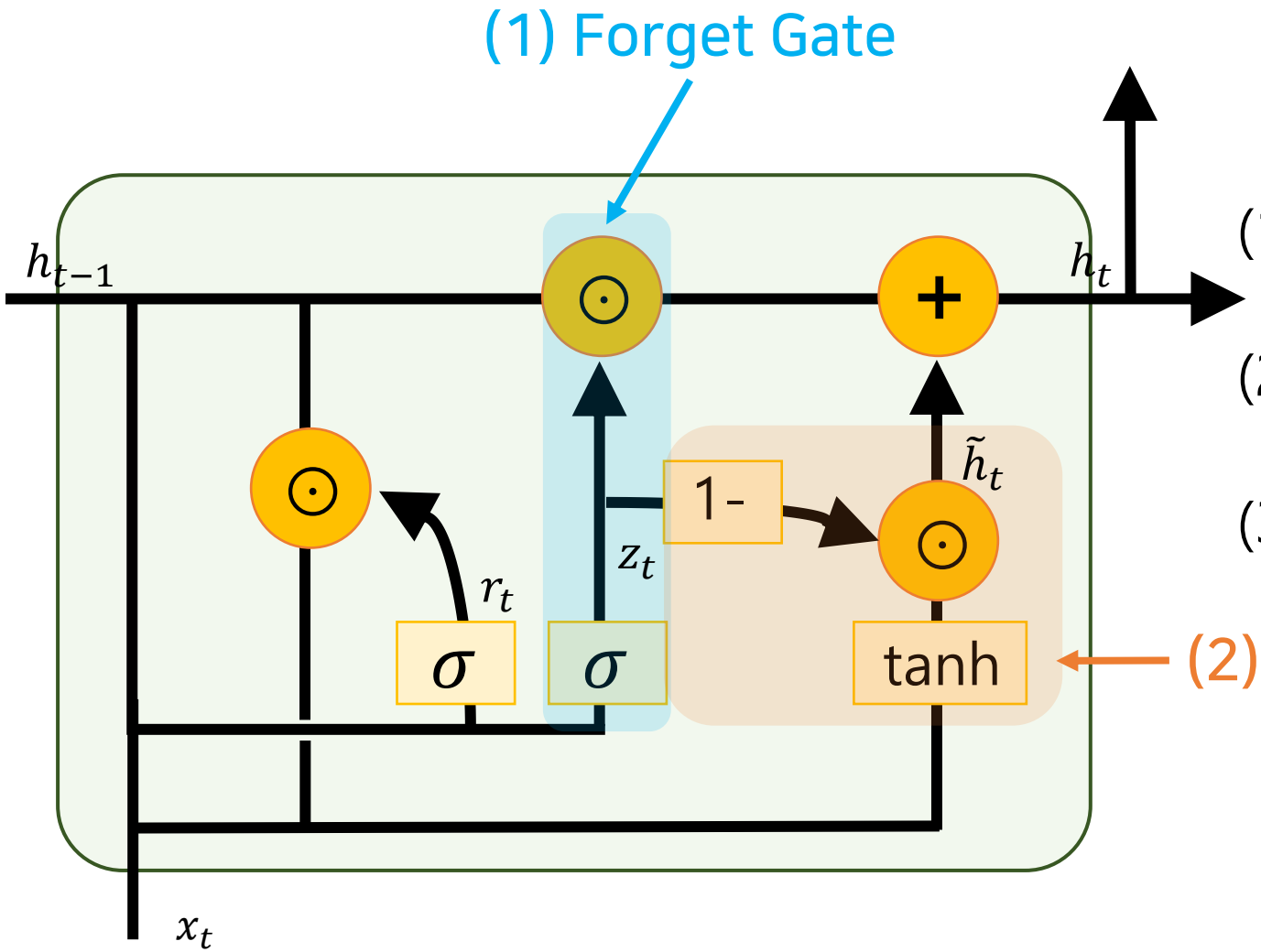


3. GRU : Reset Gate



$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

GRU : Update Gate



(1) $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$

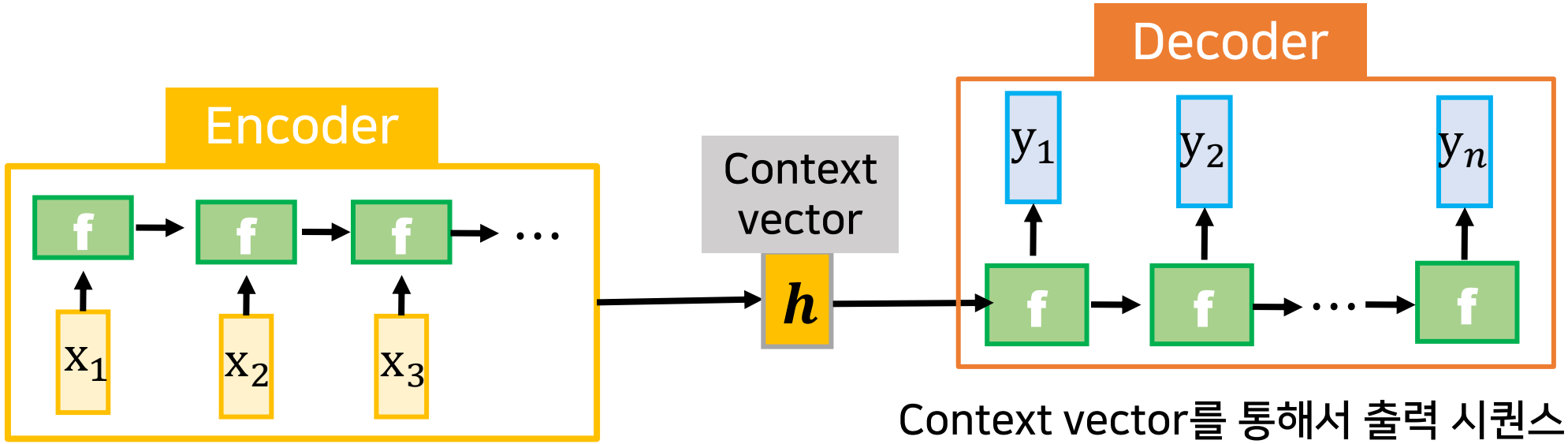
(2) $\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$

(3) $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$

Contents

7장 자연어 처리

Encoder-Decoder



Context vector를 통해서 출력 시퀀스 생성

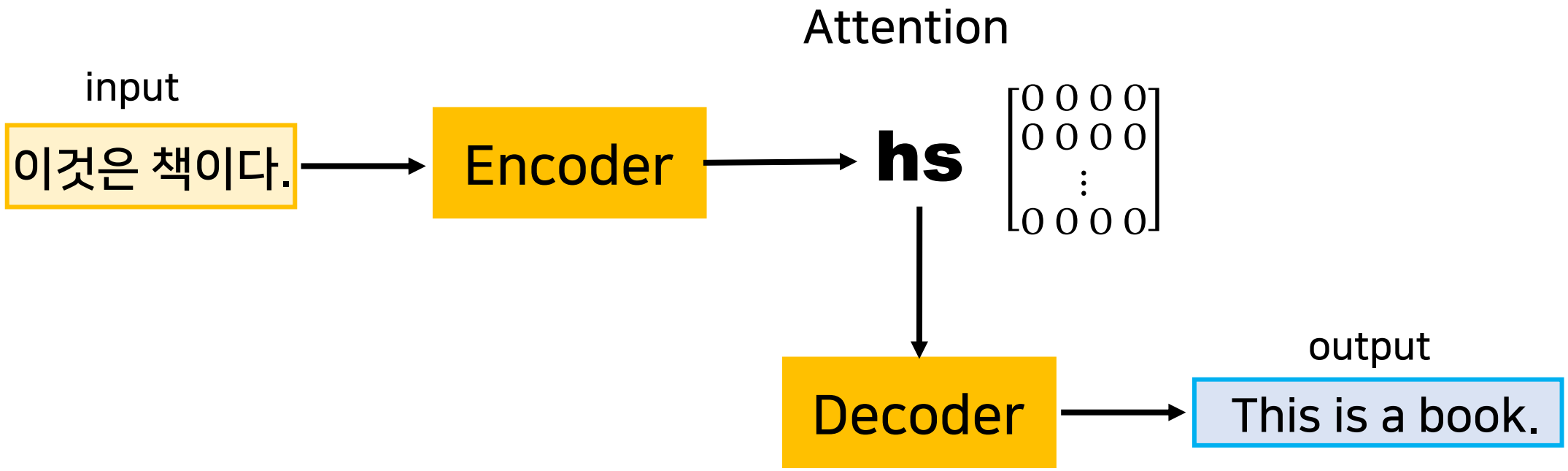
입력 시퀀스를 하나의 고정된 크기의 벡터 표현으로 압축

$$x \rightarrow Decoder(Encoder(x))$$

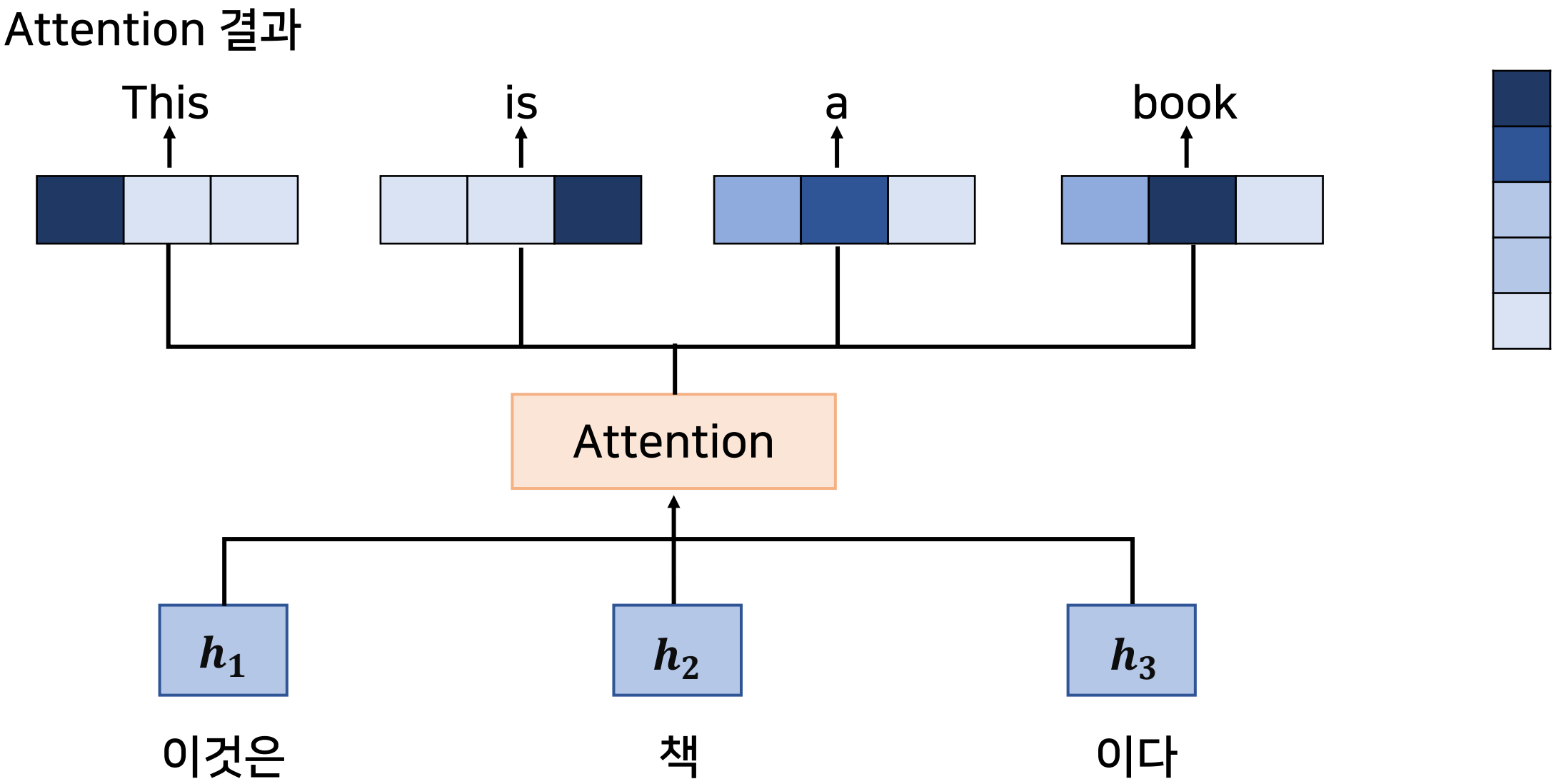
→ 정보 압축으로 인한 decode의 한계

Attention Mechanism

Decoder 개선

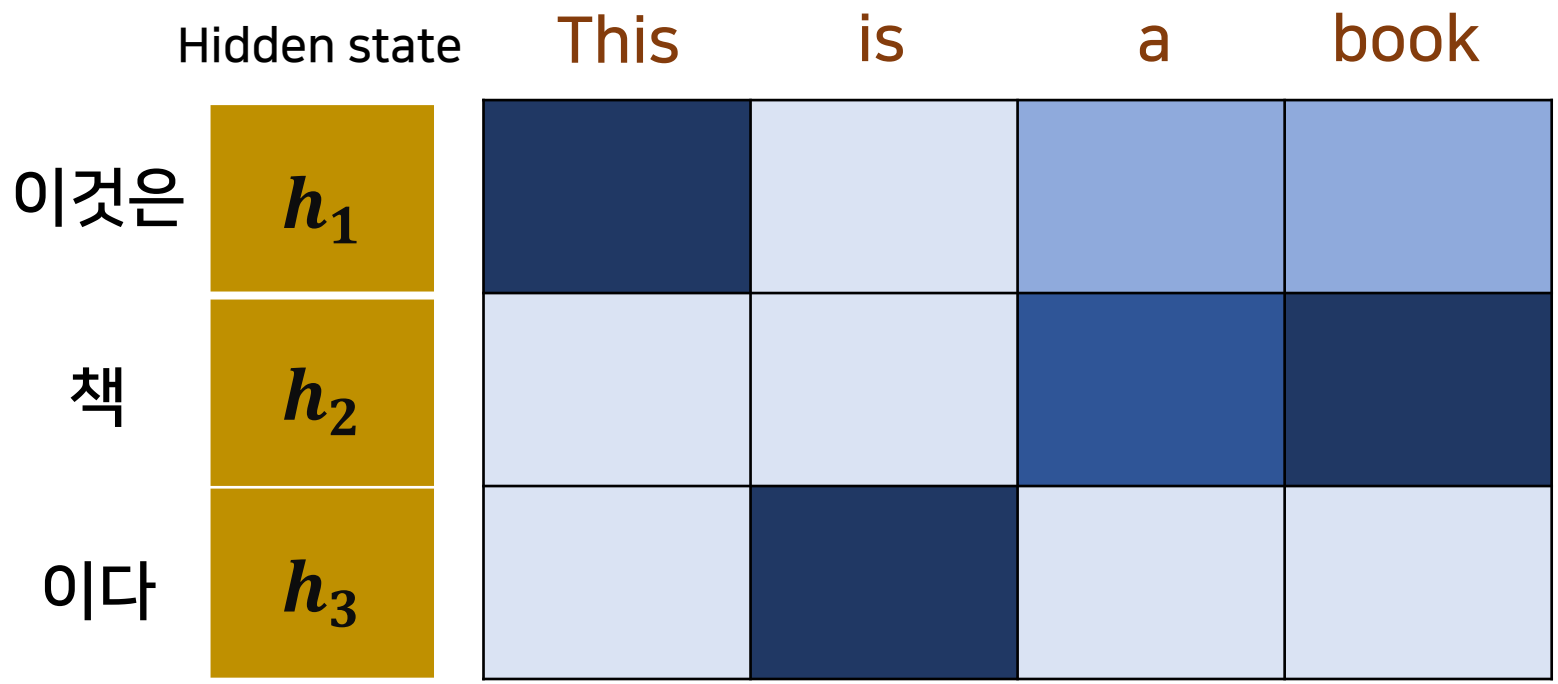


Attention Mechanism



Attention Mechanism

단어 사이의 관계



Transformer

Positional Encoding

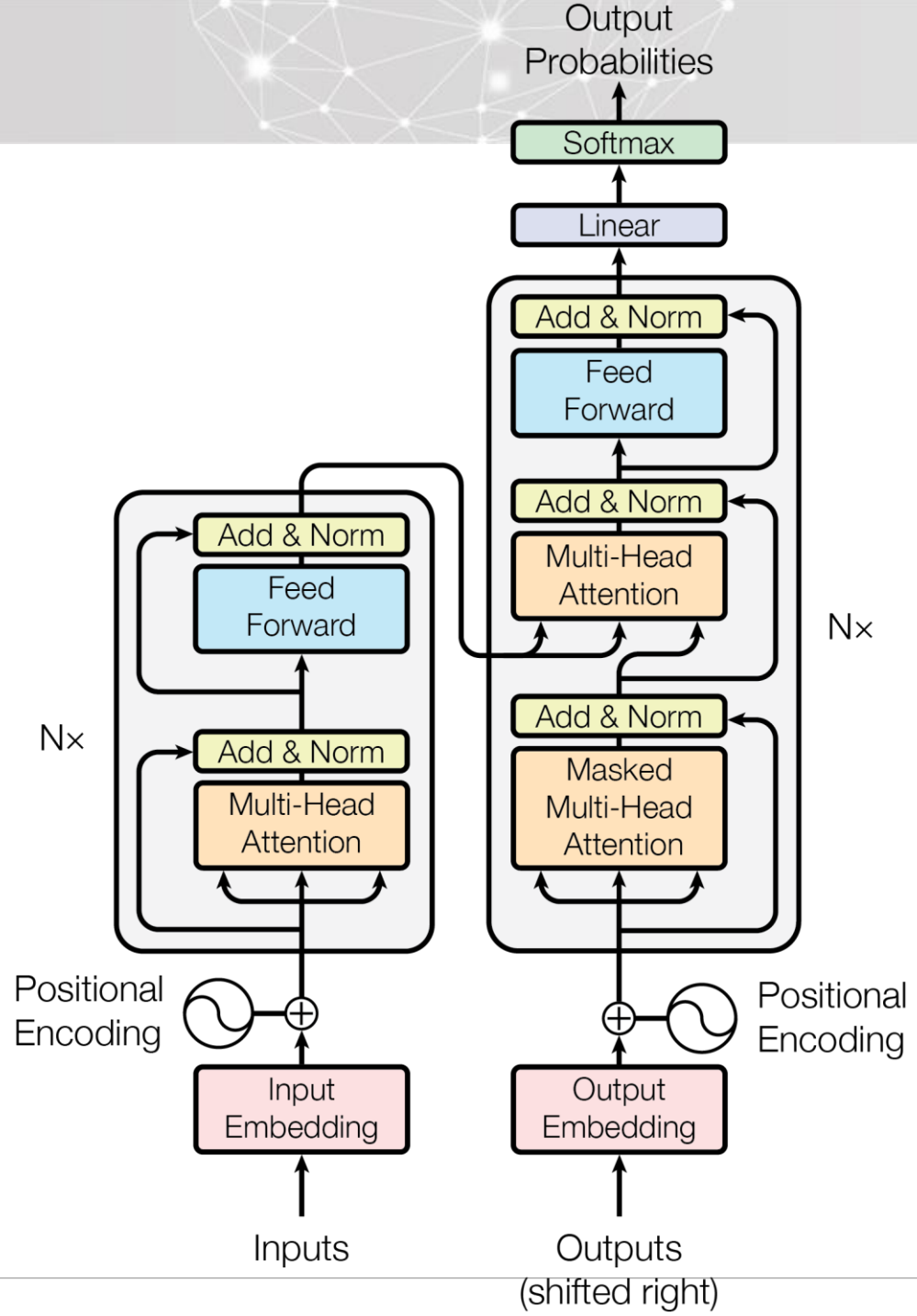
Self Attention

Multi-Head Attention

Residual Connection

Encoder N-Layer

Decoder N-Layer



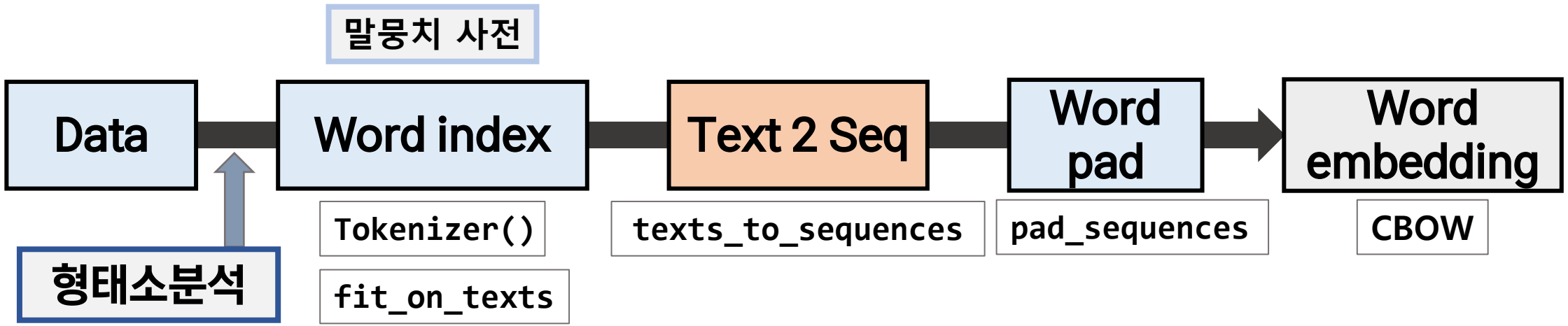
Contents

7장 자연어 처리

실습예제

1. Bag of Words
2. KoNLPy

Word Embedding 과정



Tokenizer

Tokenizer() #text를 전처리하는 클래스

```
keras.preporcessing.text.Tokenizer
tk=Tokenizer()
tk.fit_on_texts() #사용빈도에 따른 단어 정렬
tk.word_index #사용빈도 순으로 단어 뭉치(corpus) 전체 출력
tk.word_index['word'] #'word' 단어의 사용빈도 순위 출력
tk.word_count #전체 단어의 사용된 횟수 출력
tk.word_count['word'] #'word' 단어가 사용된 횟수 출력
all_text=tk.texts_to_sequences(alldata)
#문장을 만들어진 말뭉치 사전의 index를 기준으로 벡터로 변환
```

pad_sequences

pad_sequences() #서로 다른 길이의 문장을 같은 길이로 만들어주는 함수

```
pad_sequences(seq, maxlen=None, padding='pre')
```

-seq : 입력 데이터 (각 성분이 sequence)

-maxlen : sequence의 최대 길이

-padding : 각 sequence의 처음(pre) 또는 끝(post)을 디폴트 값(0)으로 패딩하여 길이를 맞춤

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
all_pad=pad_sequences(all_text, maxlen=150, padding='post')
```

Embedding

Embedding() #one-hot 벡터를 밀집 벡터로 전환해주는 함수

```
Embedding(input_dim, output_dim, input_length=None)
```

-input_dim : 입력 차원 설정

-output_dim : 출력 차원 설정

-input_length : 입력 데이터(sequence)의 정해진 길이 값 설정

```
from tensorflow.keras.layers import Embedding
model.add(Embedding(len(tk.word_index)+1, 300, input_length = 150))
```

SimpleRNN

SimpleRNN() #output이 input에 연결되는 완전연결 순환망(RNN)

```
SimpleRNN(units, activation='tanh')
```

-units : output 텐서의 차원 설정

-activation : 활성화 함수 설정. 기본값은 tanh

```
from tensorflow.keras.layers import SimpleRNN
model.add(SimpleRNN(10))
```

LSTM

LSTM()

#LSTM 함수

```
LSTM(units, activation='tanh')
```

-units : output 텐서의 차원 설정

-activation : 활성화 함수 설정. 기본값은 tanh

```
from tensorflow.keras.layers import LSTM
model.add(LSTM(32))
```

Bidirectional

Bidirectional()

#양방향 RNN 입력 함수

```
Bidirectional(layer)
```

-layer : 양방향으로 변환한 후 적용되는 층

```
from tensorflow.keras.layers import Bidirectional

model.add(Bidirectional(LSTM(32)))
```

RNN 모델 설정

RNN 모델

```
model = Sequential()

model.add(Embedding(len(tk.word_index)+1, 300, input_length = 150))

model.add(SimpleRNN(10))

model.add(Dense(1, activation = 'sigmoid'))

model.summary()
```

RNN 모델 학습

compile/fit

```
model.compile(loss='binary_crossentropy', optimizer='adam',
      metrics='acc')
hist=md.fit(train2, train['sentiment'], epochs=2, batch_size=512)
```

평가

```
md.evaluate(test)
```

RNN 모델 결과 예측 및 제출

predict

```
result = model.predict(test2) #결과 예측

sub['sentiment'] = result

sub.to_csv('popcorn.csv', index=False)

#결과를 popcorn.csv에 index는 제외하고 저장
```

LSTM 모델 설정

LSTM 모델

```
model2 = Sequential()

model2.add(Embedding(len(tk.word_index)+1, 300, input_length = 150))

model2.add(Bidirectional(LSTM(60)))

model2.add(Dropout(0.3))

model2.add(Dense(64, activation = 'relu'))

model2.add(Dropout(0.3))

model2.add(Dense(1, activation = 'sigmoid'))

model2.summary()
```