Day2

딥러닝을 이용한 자연어 처리 입문

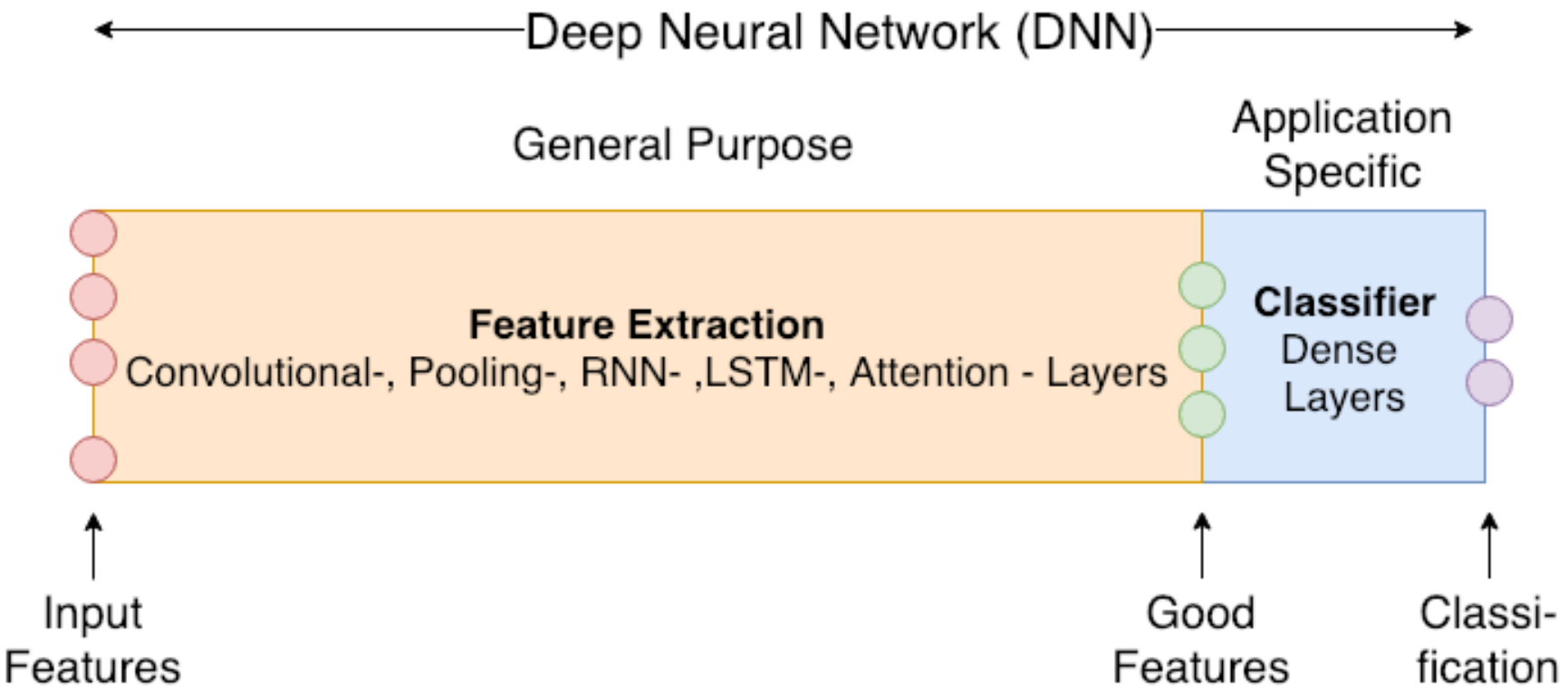
한국과학기술정보연구원

이 홍 석
(hsyi@kisti.re.kr)

Lecture 07

순환 상태 이해 및 RNN 모델 소개

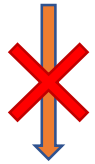
DNN에서 특성추출 알고리즘과 분류기



What is Sequential data?

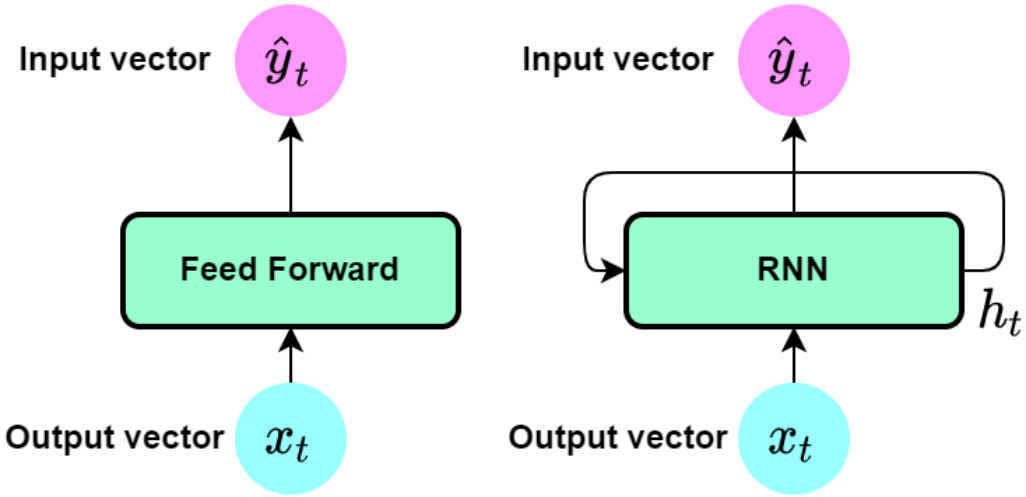
- 순차 데이터는 시퀀스에 있는 데이터로, 데이터의 순서가 중요한 일종의 데이터이다.
- **순차데이터 예제**
  - ✓ 언어 데이터/문장 : 단어 순서를 변경할 수 없다.
  - ✓ Time Series Data ~ 일별 주식 데이터
  - ✓ Biological Data ~ DNA의 서열은 유지되어야 한다.

"My name is Yi Hongsuk“



"Name is my Yi Hongsuk"

FFN과 RNN 차이는 순환하는 정보가 더 있다.



기존 MLP 신경망과 RNN의 차이점은 무었인가?

❖ 기존 다층 퍼셉트론 신경망(MLP)은 기계학습 보다 성능이 우수지만 몇 가지 단점이 있다.

(1) MLP는 입력 길이가 고정되어 있다.

길이 4: ["Hello", "How", "are", "you]

길이 9개: ["My", "Name", "is", "Yi", "Hongsuk", "and", "I", "am", "sleeping"]

(2) MLP는 순차 데이터를 잘 다룰 수 없다.

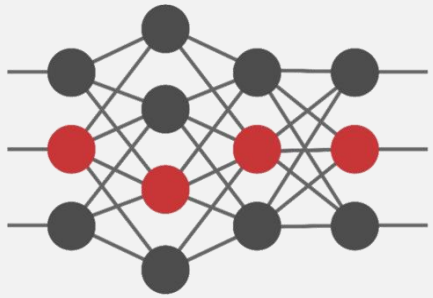
"I am Yi Hongsuk, not Dongsuk",

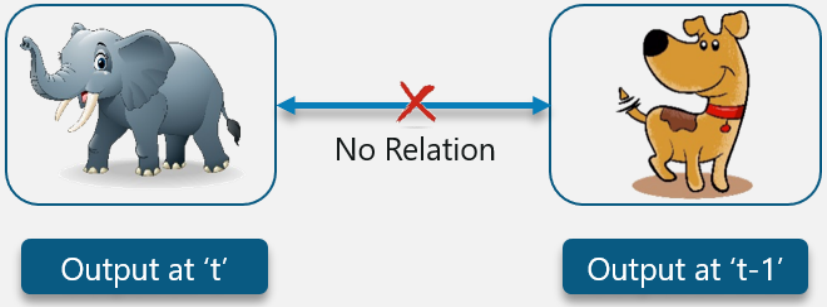
"I am Dongsuk, not Hongsuk"

(4) 시퀀스 전체에서 매개변수를 공유할 수 없다.

"what is your name? My name is Hongsuk"

(3) MLP는 입력 순서는 중요하지 않다.

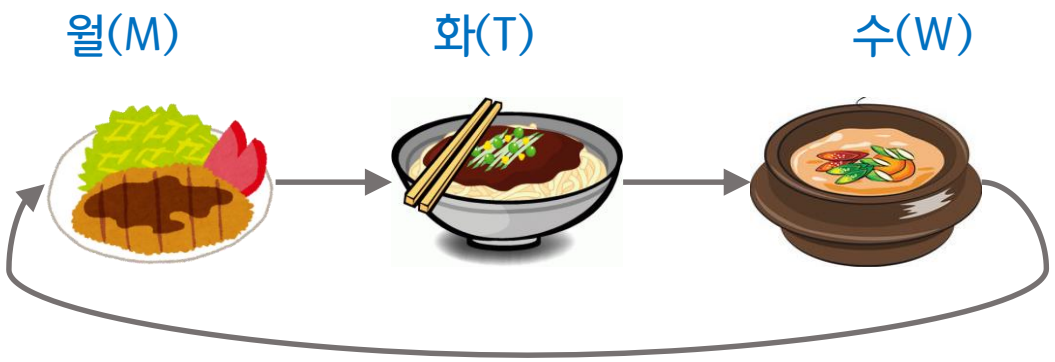




순서가 있는 데이터의 순환하는 것은 무엇인가?

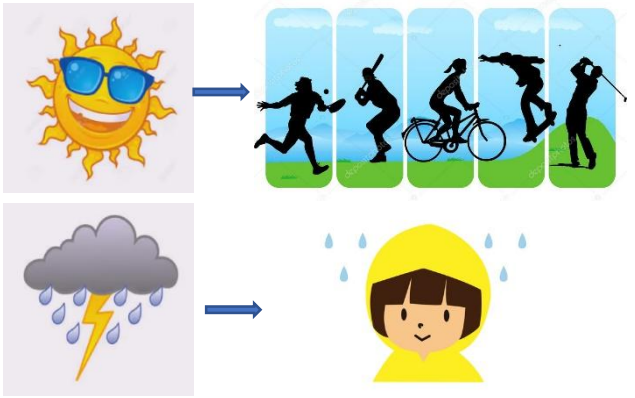
- 순서가 있는 데이터에서 순서 유지를 표현을 어떻게 할 것인가?
  - ✓ 책을 읽을 경우, 기억해야 할 것은 바로 이전 페이지의 내용을 기억해야 한다.
  - ✓ 순서를 유지하면서 요리를 한다면, 기억해야 할 것은 바로 전에 먹었는 요리

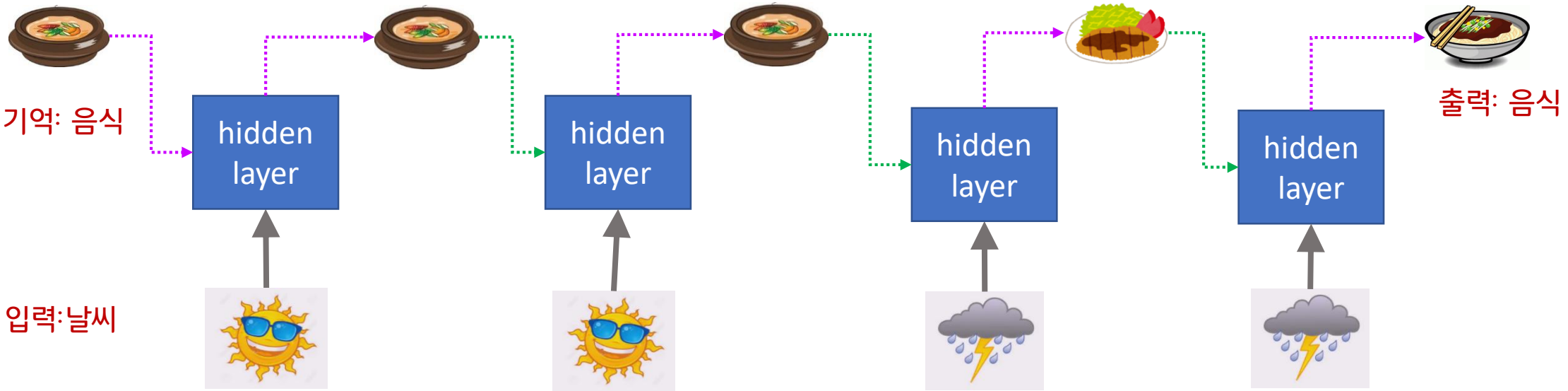
시퀀스를 처리하기 위해 모든 타임스탬프에 적용된 recurrence relation가 있다.



음식에 대한 패턴: 기억(memory) 혹은 상태(state)

- 메뉴는 날씨에 따라서 변하더라!
  - ✓ 맑은 날이면 전날과 같은 메뉴
  - ✓ 비온 날이면 패턴에 따른 메뉴
- RNN 신경망을 설계해보자
  - ✓ 입력 : 날씨
  - ✓ 기억(상태): 음식
  - ✓ 출력 : 음식

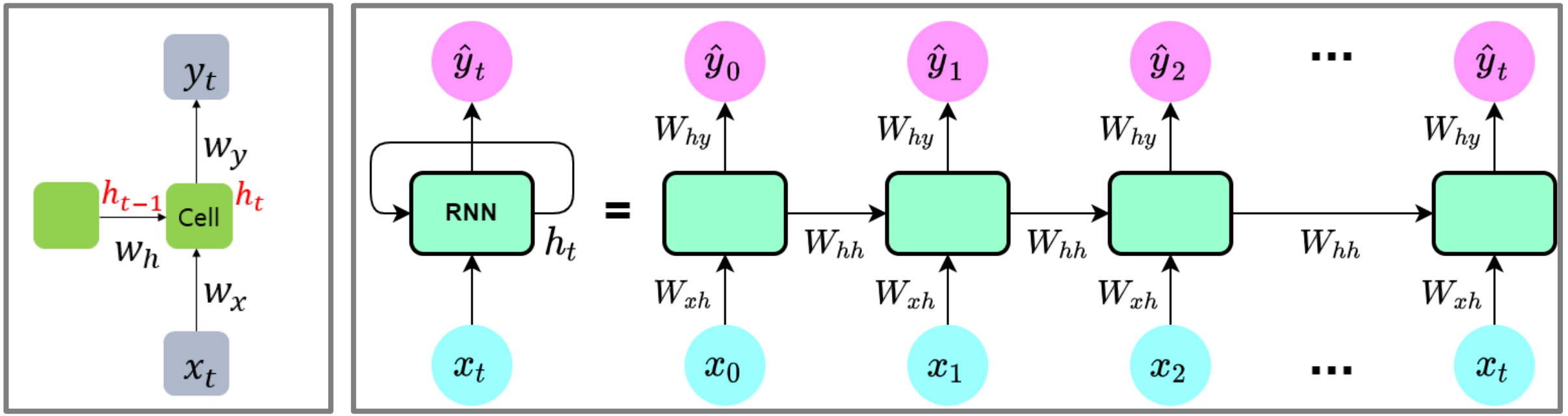




RNN 모델에서 은닉상태(hidden state)

- 은닉층에 있는 RNN의 처리 단위를 셀(cell)이라고 한다.
  - ✓ 셀의 출력을 은닉 상태(hidden state)라고 한다.
- RNN이 과거의 정보를 기억하고 있는 비결
  - ✓ 현재 시점의 h_t연산을 위해 직전 시점의 h_{t-1}를 입력으로 사용한다.

$$h_t = g(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$$



RNN 모델의 특징

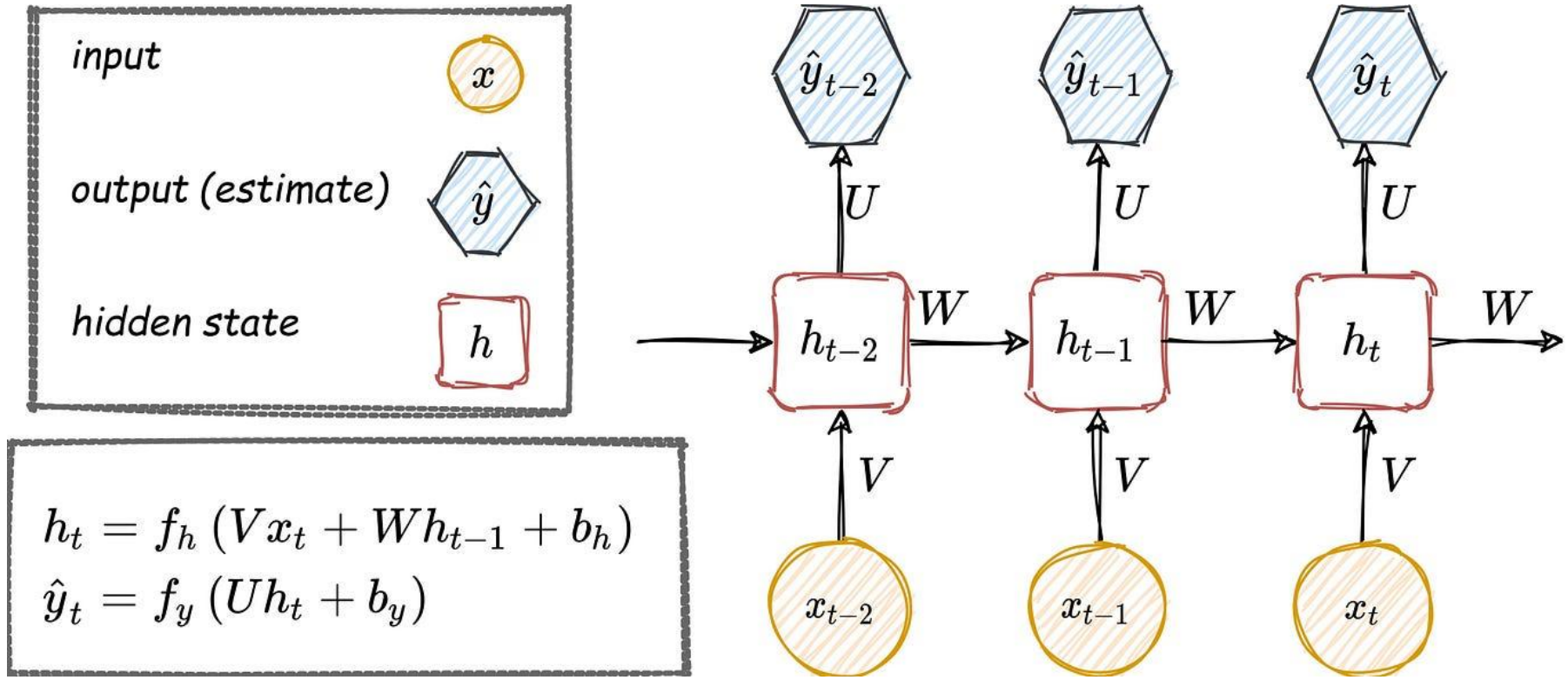
❖ 시퀀스 데이터를 모델하기 위해 필요 조건 4가지와 RNN의 적합성

- ✓ 가변 길이(variable-length)를 잘 다루어야 한다.
- ✓ 시퀀스의 순서를 유지해야 한다.
- ✓ 장기적인 의존성을 잘 추적해야 한다.
- ✓ 시퀀스 간 파라미터를 공유해야 한다.

❖ RNN 모델이 가능하다.

- ✓ RNN은 입력 시퀀스의 길이에 관계없이 처리 할 수 있다.
- ✓ RNN은 시퀀스의 순서를 유지해야 한다.
- ✓ LSTM이난 GRU는 장기적인 의존성 문제를 잘 다룬다.
- ✓ RNN은 시퀀스 내의 모든 타임스텝에서 동일한 파라미터를 사용한다.

RNN에서 과거 기억: 메모리(state) 및 업데이트



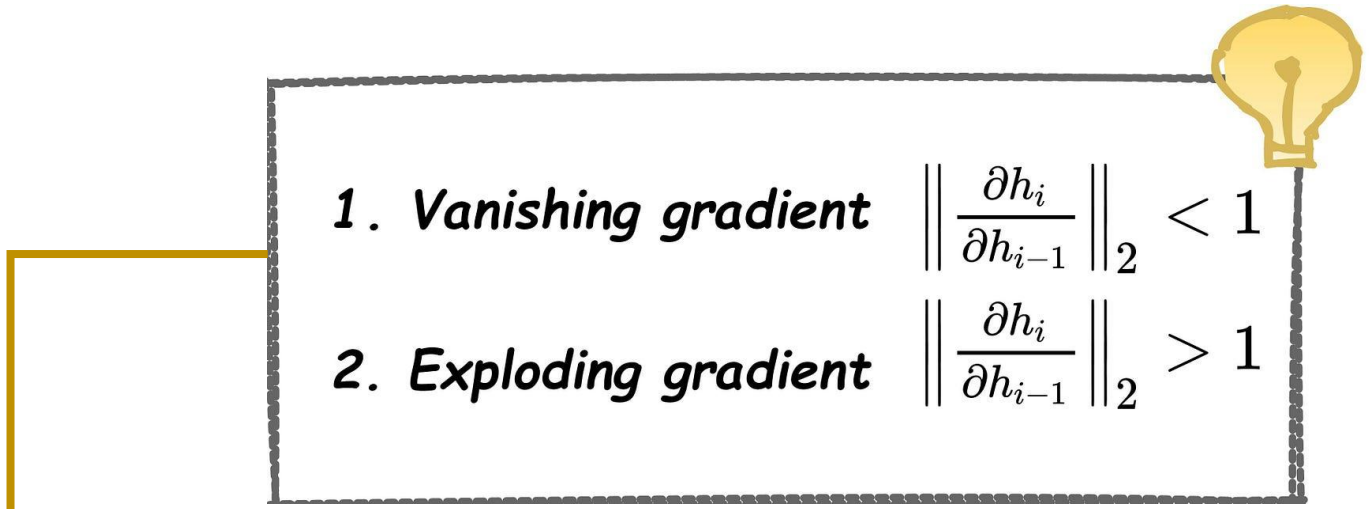
(source) https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

RNN에서 발생할 수 있는 두 가지 주요 문제

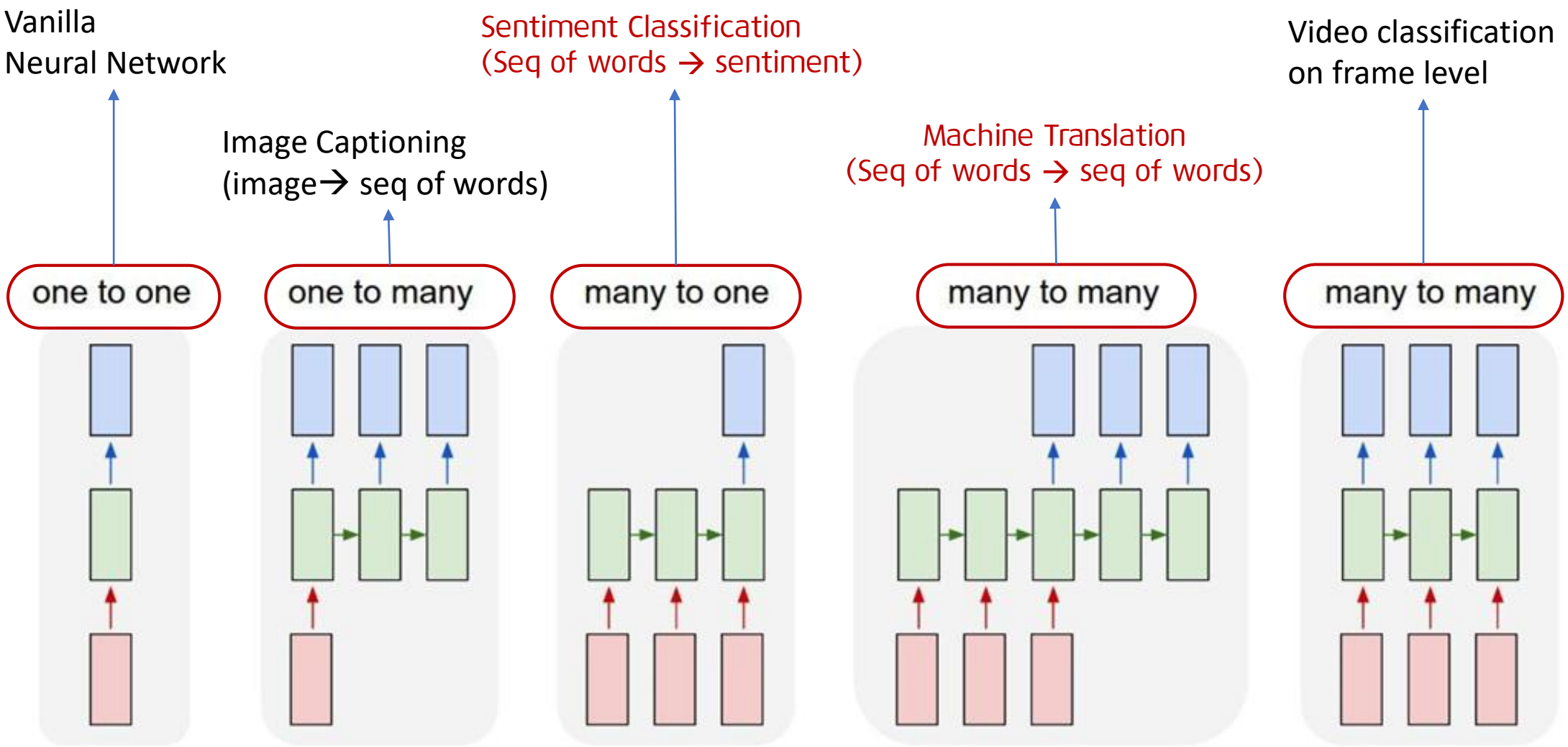
❖ 기억에 대한 편미분을 시간에 대하여 진행하며, 이 값을 계속 곱하는 연산을 할 때 생기는 문제점

✓ Exploding Gradients는 반복되는 기울기 계산과 관련된 많은 값이 1보다 클 때 이 문제를 폭발 기울기라고 합니다

✓ Vanishing Gradients는 반복되는 그래디언트 계산 값이 너무 작거나 1보다 작을 때 발생합니다.



해결하고자 하는 문제 유형에 따른 RNN 아키텍처



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

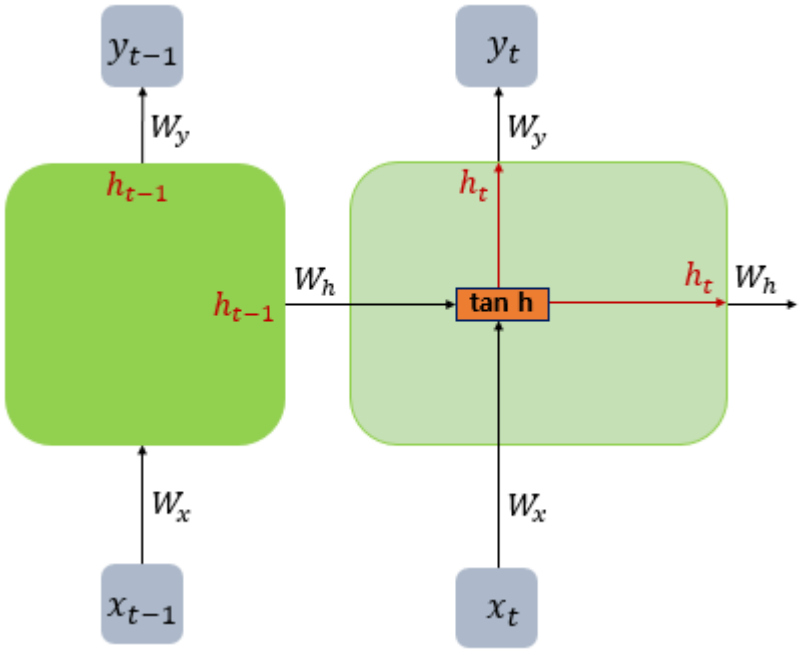
가장 단수한 형태의 RNN

❖ 가장 단순한 RNN인 바닐라(Vanilla) RNN의 한계점

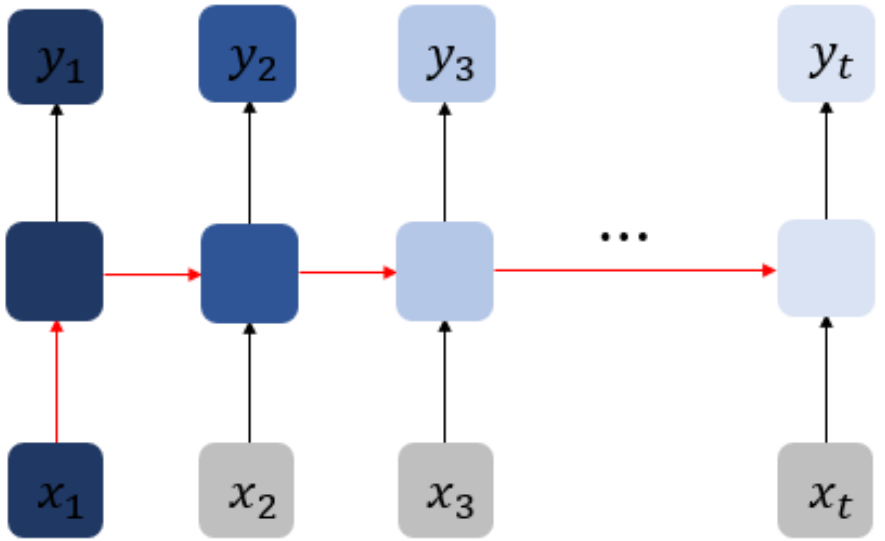
✓ 시퀀스 길이가 길어질 수록 앞의 정보가 뒤로 충분히 전달되지 못하는 '장기 의존성 문제'를 갖고 있다.

✓ 정보량은 시간이 갈수록 점점 소실되는 것을 색의 얕아짐으로 표현

Vanilla RNN의 내부

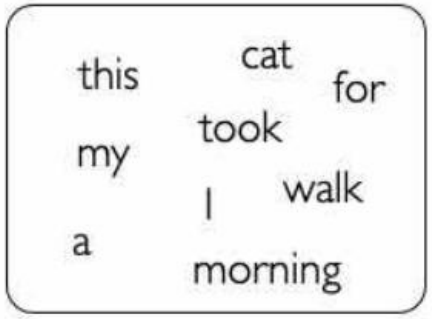


Vanilla RNN의 장기의존성 문제

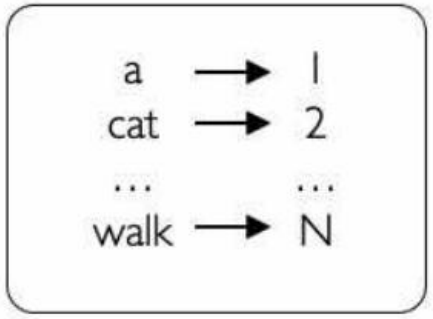


신경망용 인코딩 언어: 임베딩

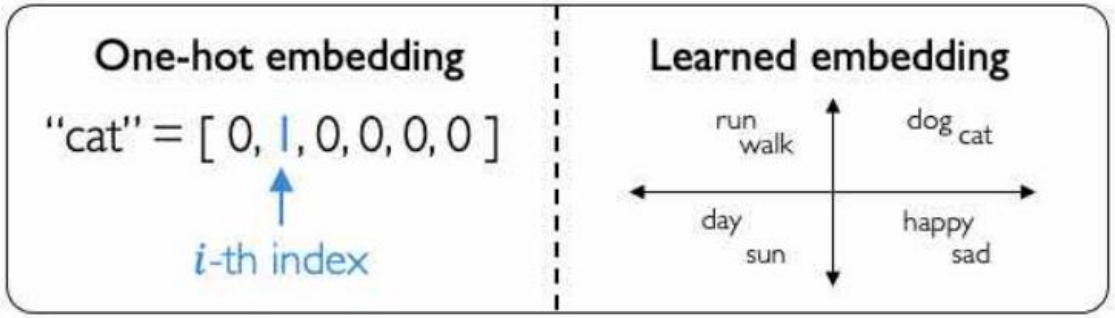
Embedding: transform indexes into a vector of fixed size.



1. Vocabulary:
Corpus of words



2. Indexing:
Word to index

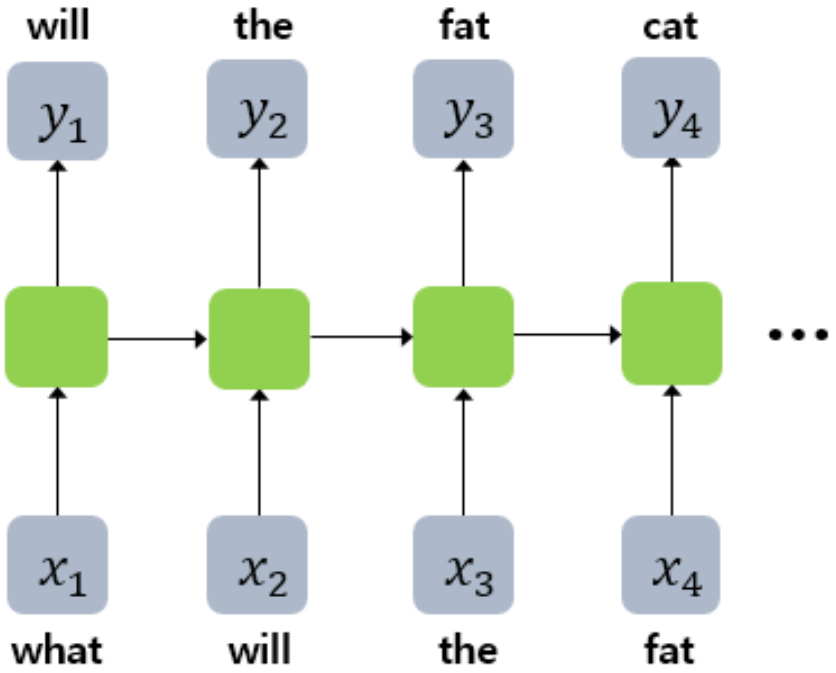


3. Embedding:
Index to fixed-sized vector

(생성) 다-대-다 모델에서 Teacher Forcing(교사 강요)

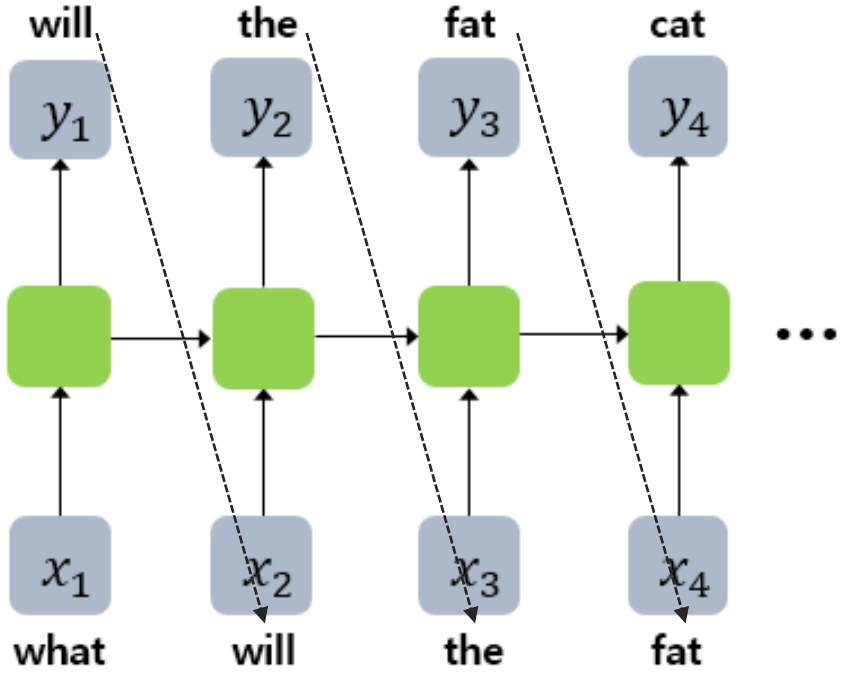
- 훈련과정은 교사학습을 적용으로 실제값을 다음 시간의 입력으로 사용한다.
- 테스트과정은 교사학습을 적용하지 않고 지금 시간의 예측값을 다음 시간의 입력으로 사용한다.

훈련과정에서 Teacher forcing 적용한다.



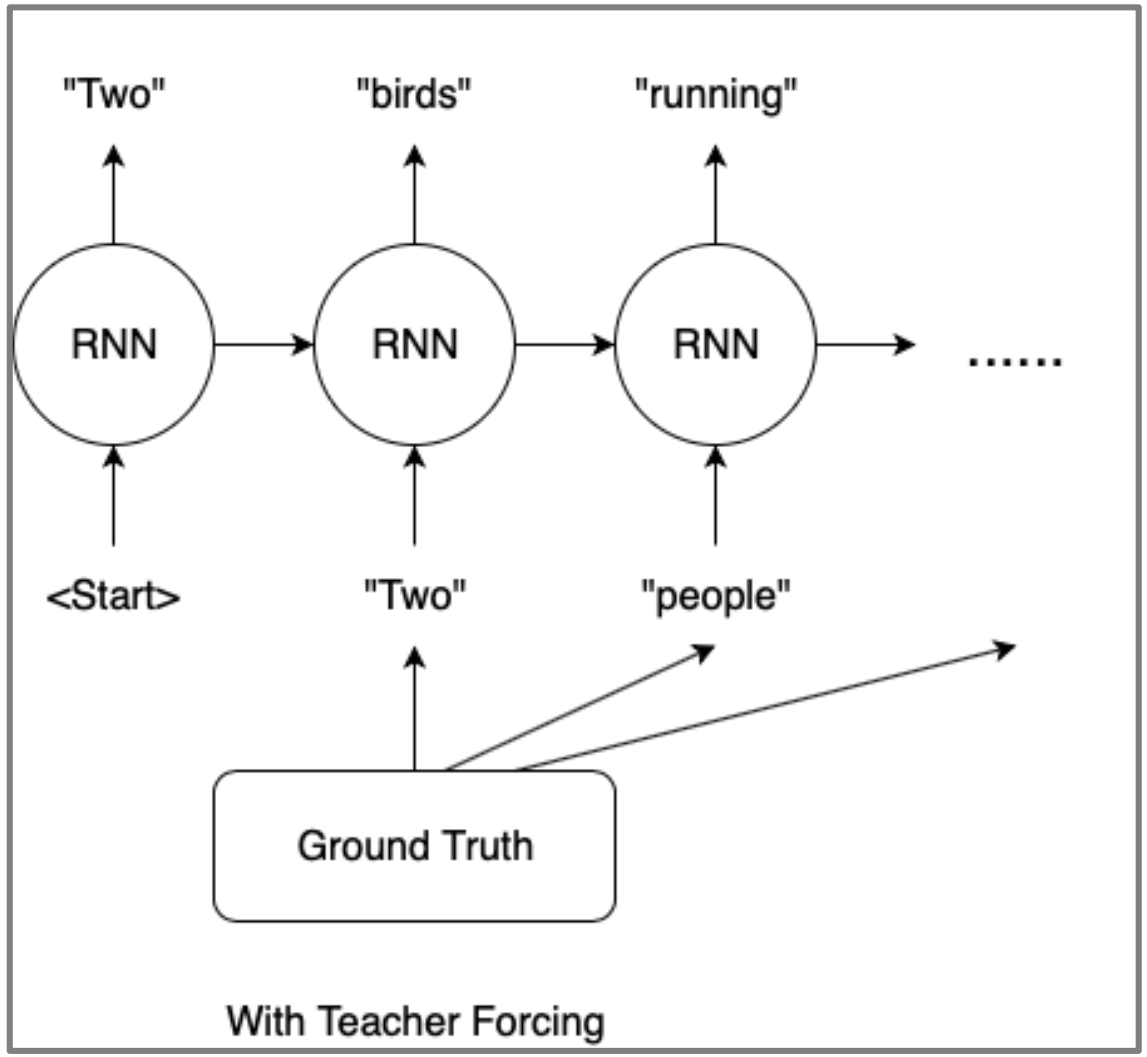
실제값 사용

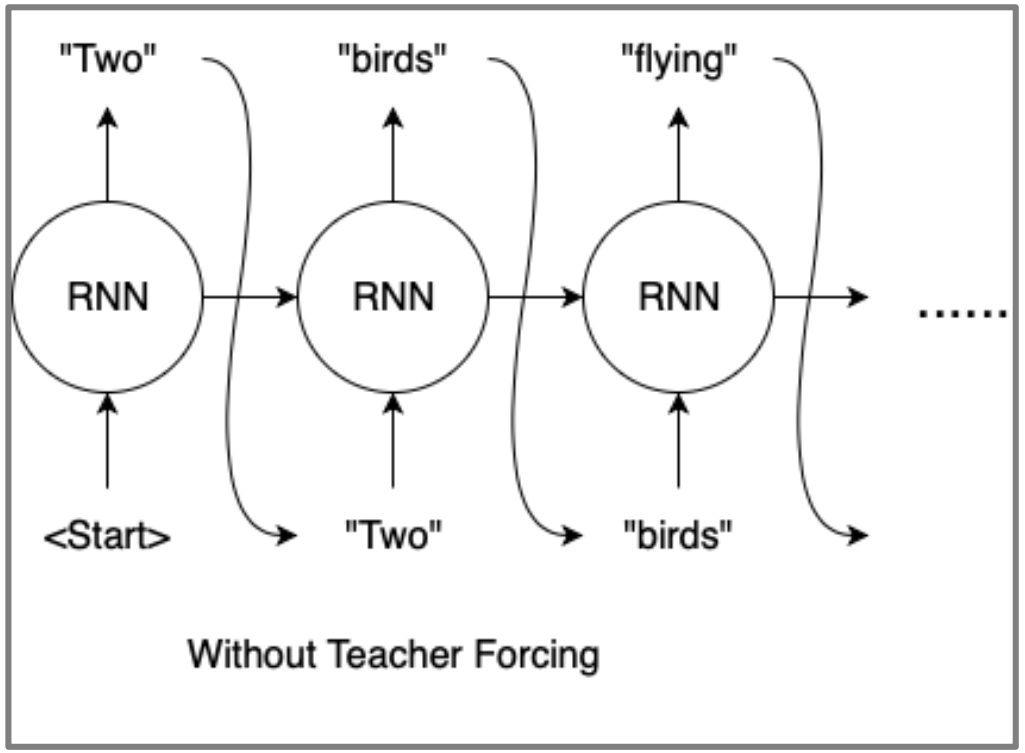
테스트 과정에서 Teacher forcing 적용하지 않는다.



예측값 사용

Teacher Forcing





Lecture 08

RNN을 이용한 텍스트 생성 모델 실습

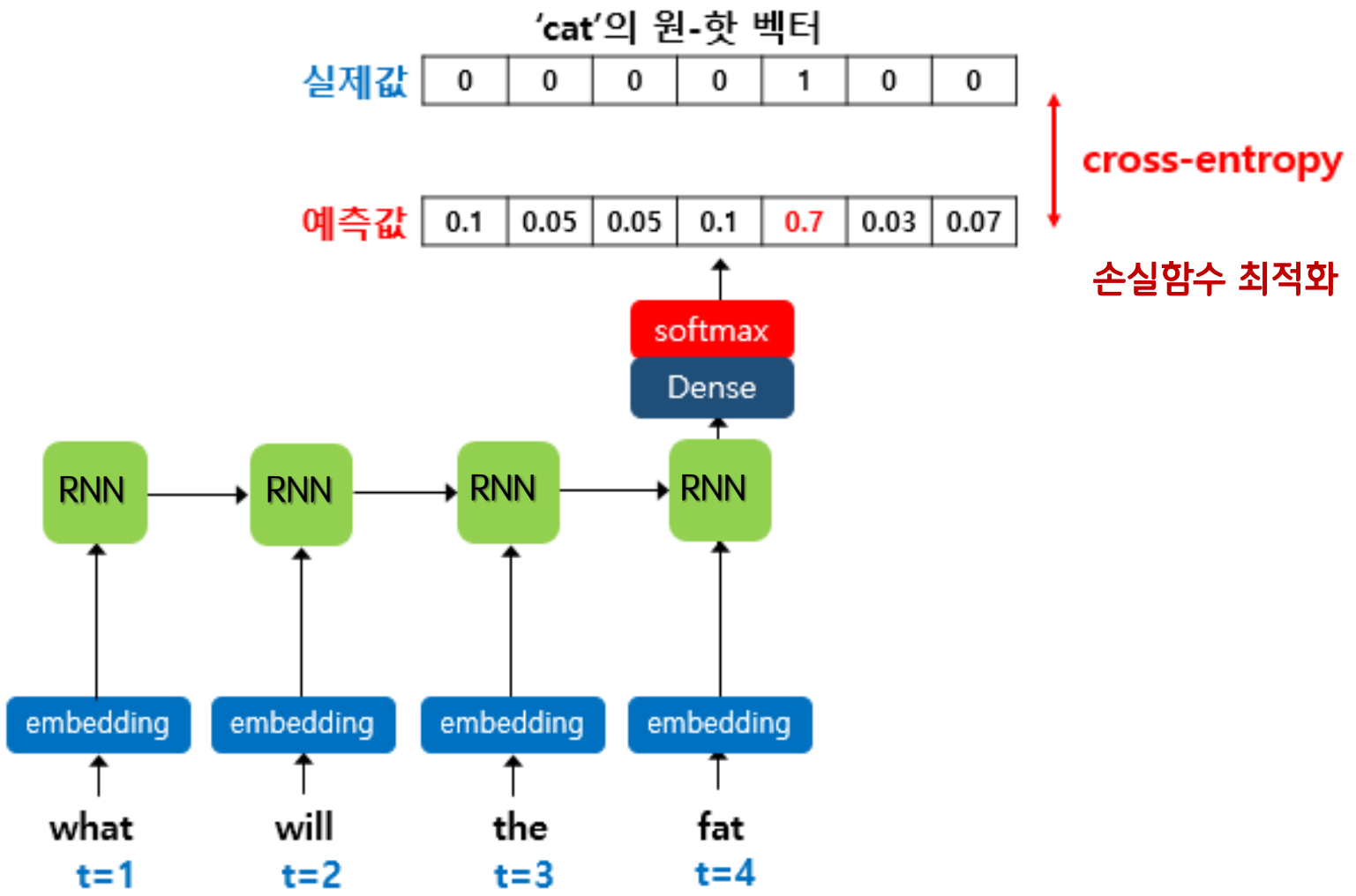
(생성) RNN 언어모델로 다음 단어 예측하기 (1/2)

Softmax 적용

Dense 층 1개
라벨 개수 (3000개)

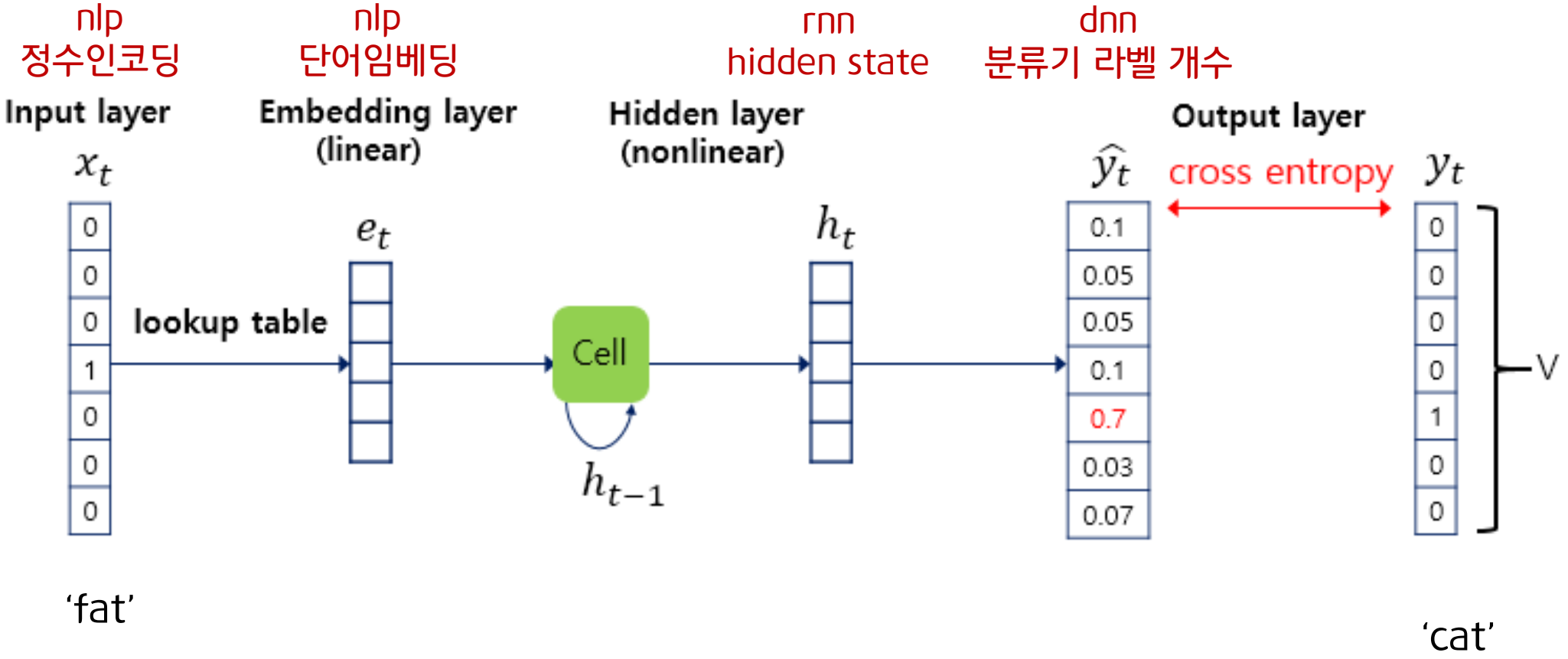
RNN 층 1개
은닉상태 뉴런 (64)

임베딩 층 1개
1차원 벡터 (256)



(생성) RNN 언어모델로 다음 단어 예측하기 (2/2)

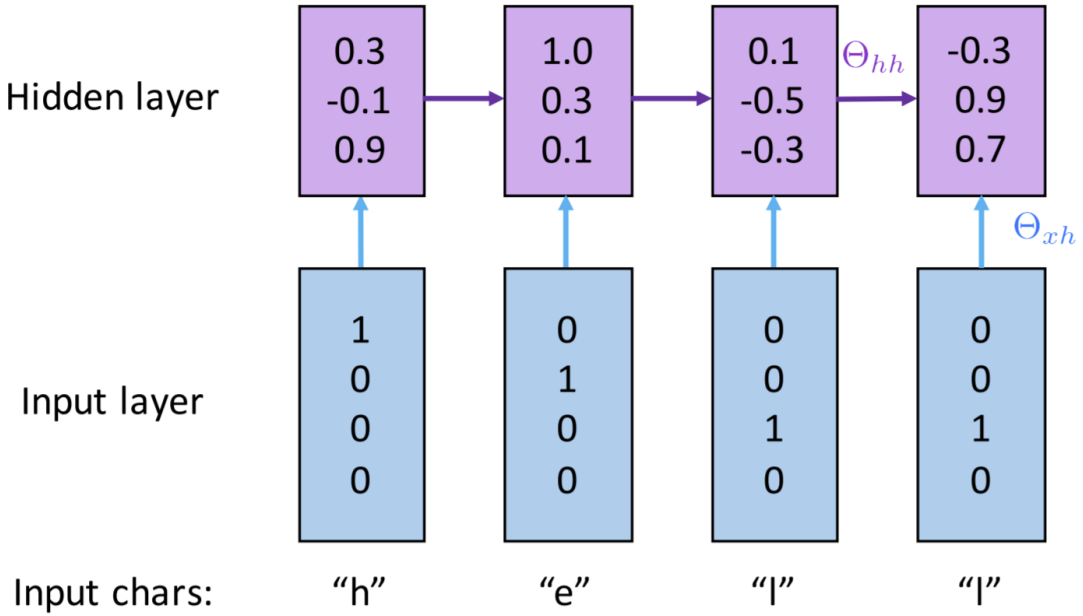
예제: timestep=4일때 입력 단어인 'fat'이 'cat'을 예측하는 과정

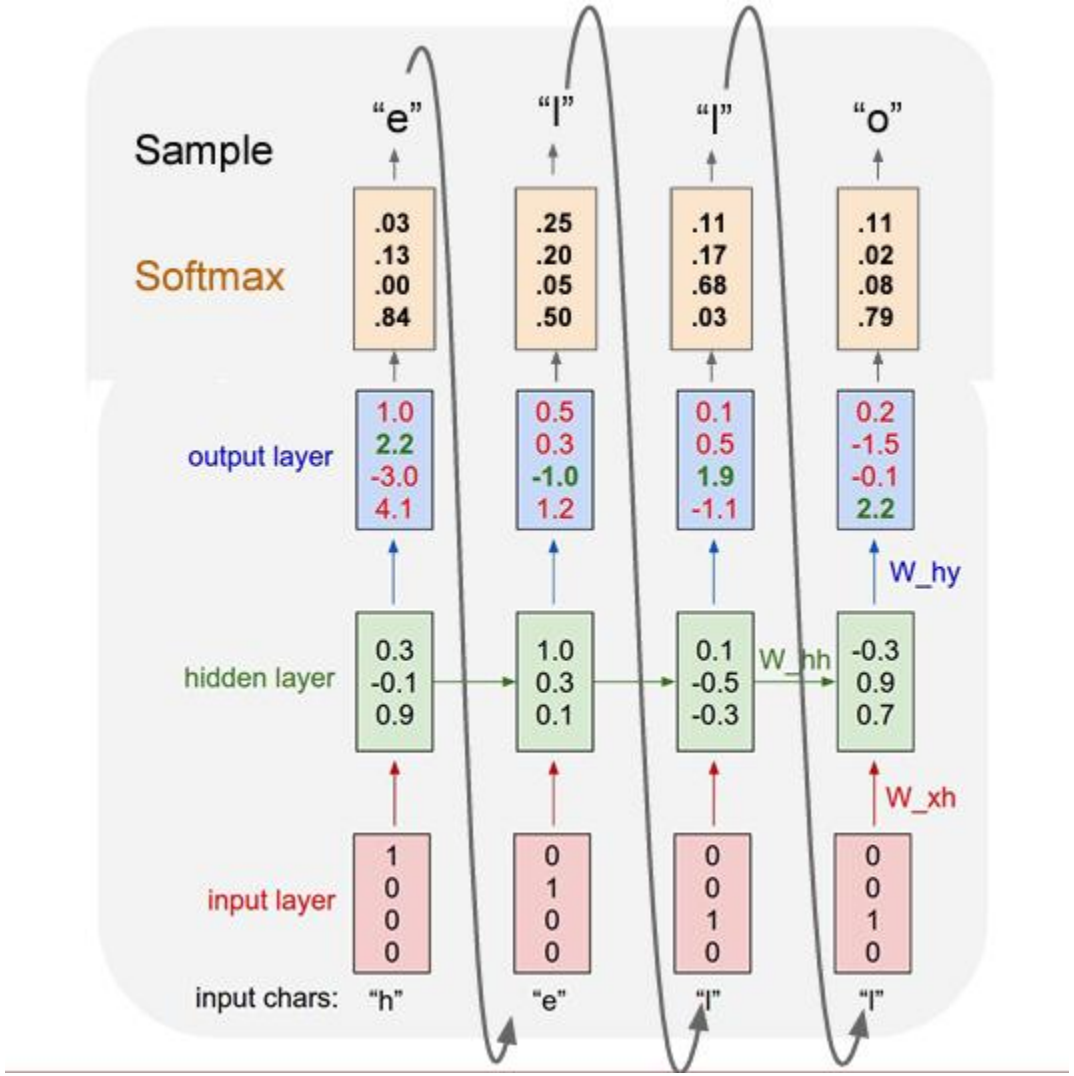


(생성) RNN Character-level language model

- 4개의 알파벳만 있다고 가정하자. “hell”: [h,e,l,o]
  - ✓ One-hot encoding을 글자를 표현

$$\mathbf{h}_t = \tanh(\Theta_{hh}\mathbf{h}_{t-1} + \Theta_{xh}\mathbf{x}_t)$$



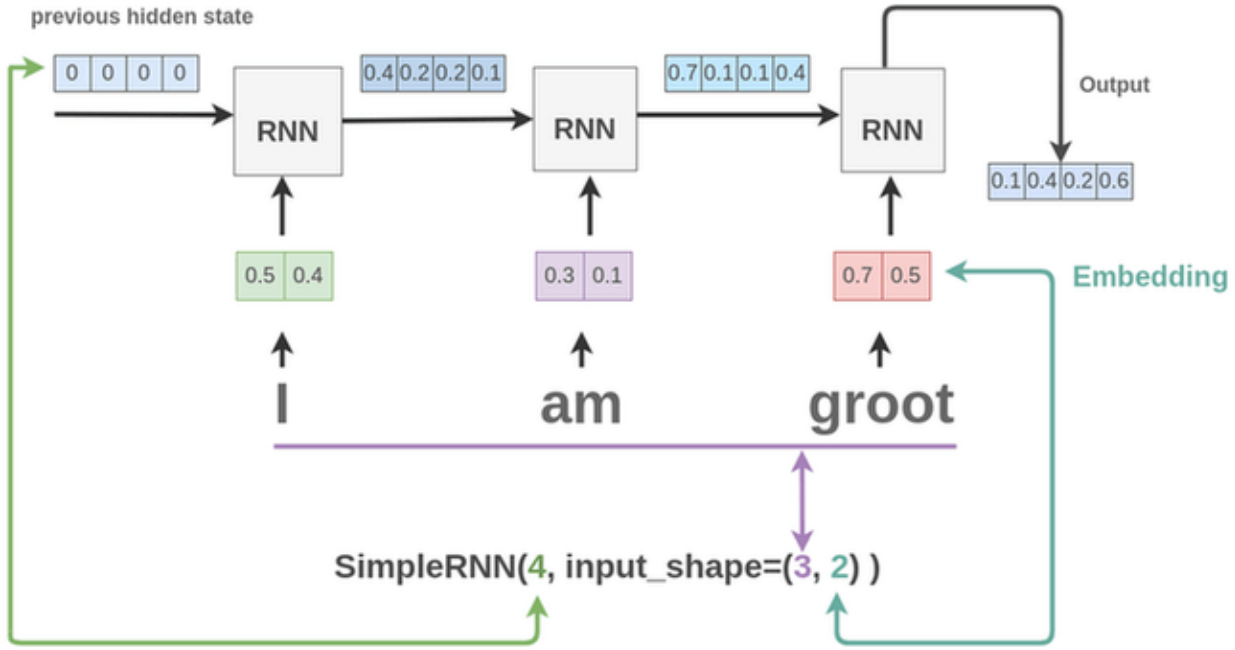


SimpleRNN 적용 예제

I am groot

Credits: Marvel Studios

Word	E1	E2
I	0.5	0.4
am	0.3	0.1
groot	0.7	0.5

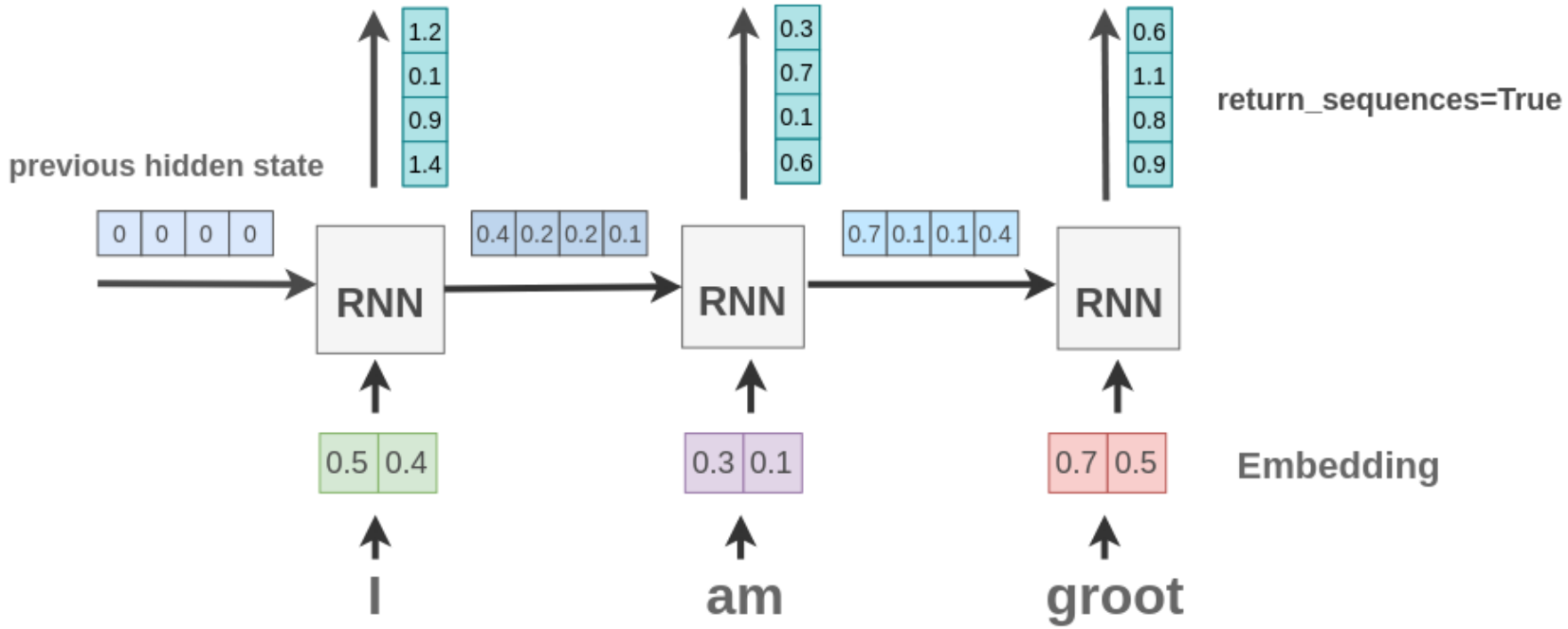



```
import tensorflow as tf
from tensorflow.keras.layers import SimpleRNN
x = tf.random.normal((1, 3, 2))
layer = SimpleRNN(4, input_shape=(3, 2))
output = layer(x)
print(output.shape)
```

```
(1, 4)
```

SimpleRNN의 Many-to-Many로 수정해서 적용하자

model.add(SimpleRNN(4, input_shape=(3, 2), return_sequences=True))




```
# multiple output
layer = SimpleRNN(4, input_shape=(3, 2), return_sequences=True )
output = layer(x)
print(output.shape)
print(output)
```

```
(1, 3, 4)
tf.Tensor(
[[[-0.6854385   0.08265962  0.30888444 -0.30752325]
  [ 0.4584542  -0.1935767  -0.91095936 -0.2416075 ]
  [ 0.7241105  -0.49960855 -0.5059616   0.7261468 ]]], shape=(1, 3, 4), dtype=float32)
```

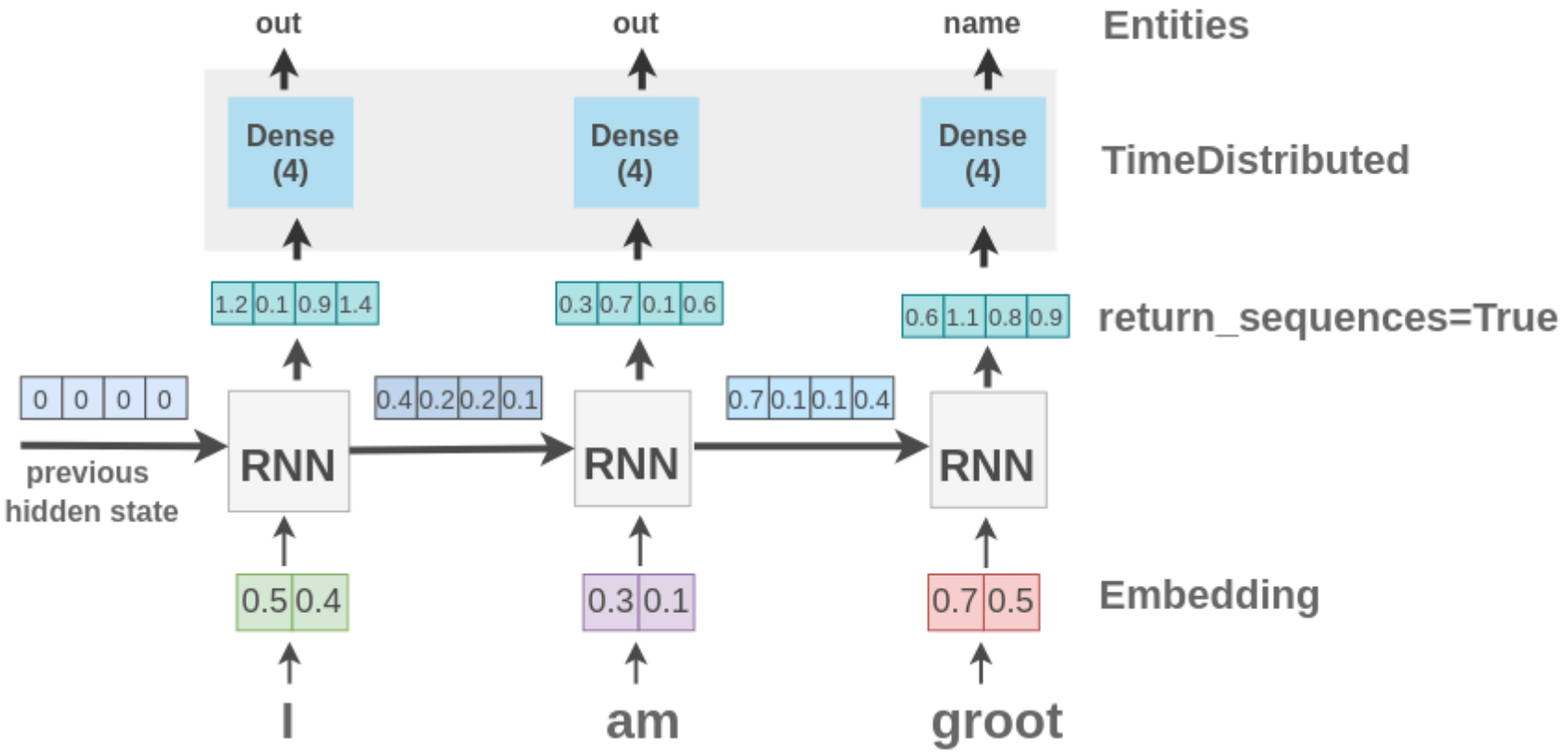
RNN with TimeDistributed : 엔티티 인식

```
model.add(SimpleRNN(4, input_shape=(3, 2), return_sequences=True))
model.add(TimeDistributed(Dense(4, activation='softmax')))
```

Identify entity

I am groot

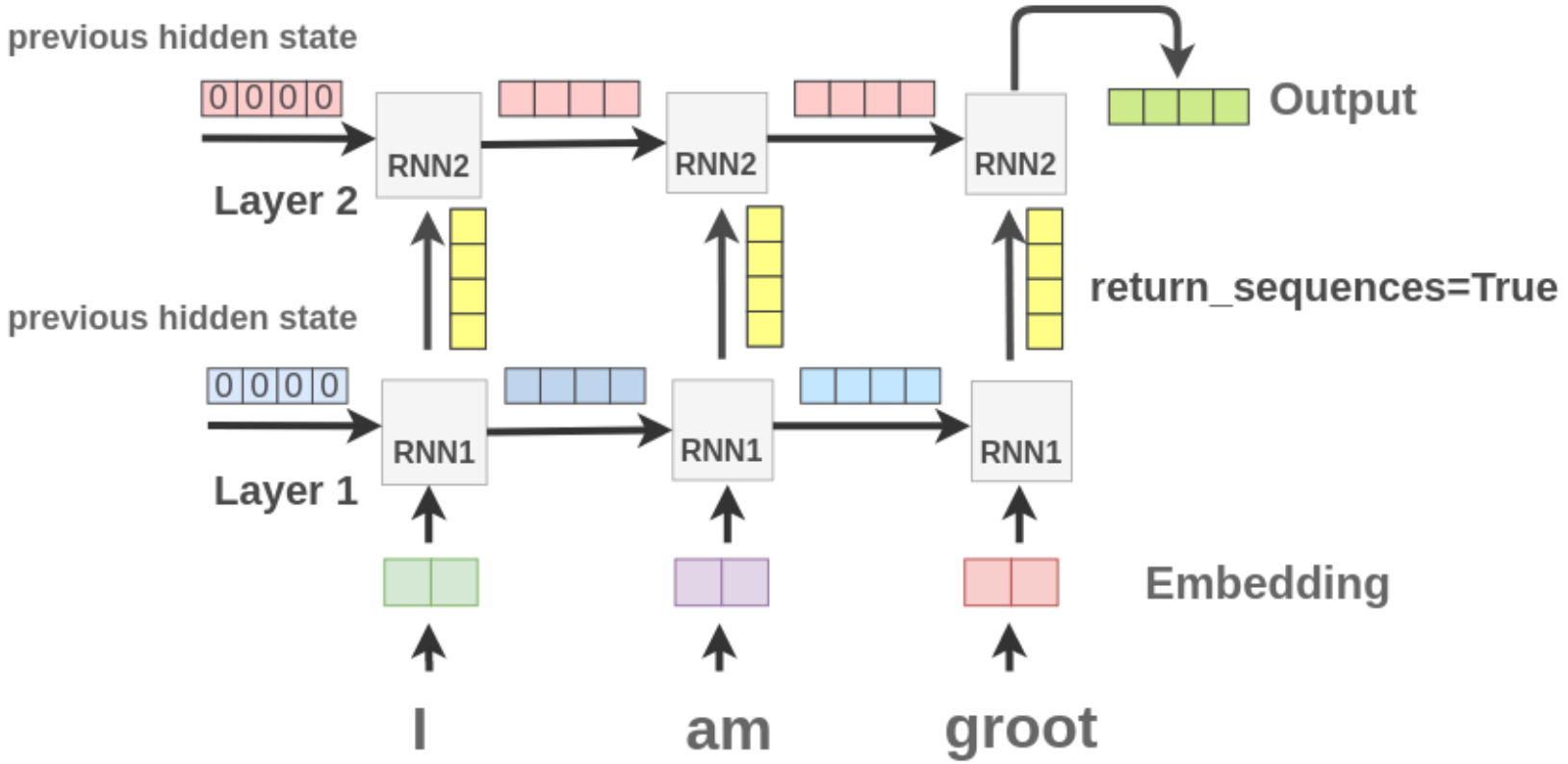
NAME



RNN Stacking Layer : Deep but Many-to-One

```
model.add(SimpleRNN(4, input_shape=(3, 2), return_sequences=True))
model.add(SimpleRNN(4))
```

1개의 출력을 원할 경우 return_sequences=False 를 사용한다.



텍스트 생성 데이터

❖ 다 대 일(many-to-one) 구조의 RNN을 사용하여 문맥을 반영해서 텍스트를 생성

text = """경마장에 있는 말이 뛰고 있다₩n
그의 말이 법이다₩n
가는 말이 고와야 오는 말이 곱다₩n"""

모델이 문맥을 학습할 수 있도록 전체 문장의 앞의 단어들을 전부 고려하여 학습하도록 데이터를 재구성 해보자.

samples	X	y
1	경마장에	있는
2	경마장에 있는	말이
3	경마장에 있는 말이	뛰고
4	경마장에 있는 말이 뛰고	있다
5	그의	말이
6	그의 말이	법이다
7	가는	말이
8	가는 말이	고와야
9	가는 말이 고와야	오는
10	가는 말이 고와야 오는	말이
11	가는 말이 고와야 오는 말이	곱다

케라스 토큰화

```
[1] import numpy as np
    from tensorflow.keras.preprocessing.text import Tokenizer
    from tensorflow.keras.preprocessing.sequence import pad_sequences
    from tensorflow.keras.utils import to_categorical
```

```
[2] text = """경마장에 있는 말이 뛰고 있다₩n
    그의 말이 법이다₩n
    가는 말이 고와야 오는 말이 곱다₩n"""
```

2]예제로 언급한 3개의 한국어 문장을 저장

```
[3] tokenizer = Tokenizer()
    tokenizer.fit_on_texts([text])
    vocab_size = len(tokenizer.word_index) + 1
    print('단어 집합의 크기 : %d' % vocab_size)
```

```
단어 집합의 크기 : 12
```

3]단어 집합의 크기를 저장할 때는 케라스 토크나이저의 정수 인코딩은 인덱스가 1부터 시작하지만, 패딩을 위한 0을 고려하여 +1을 해줍니다.

```
[4] print(tokenizer.word_index)
```

4]각 단어와 단어에 부여된 정수 인덱스를 출력

```
{'말이': 1, '경마장에': 2, '있는': 3, '뛰고': 4, '있다': 5, '그의': 6, '법이다': 7, '가는': 8, '고와야': 9, '오는': 10, '곱다': 11}
```

훈련 데이터를 만들자

```
[5] sequences = list()
    for line in text.split('₩n'): # Wn을 기준으로 문장 토큰화
        encoded = tokenizer.texts_to_sequences([line])[0]
        for i in range(1, len(encoded)):
            sequence = encoded[:i+1]
            sequences.append(sequence)

    print('학습에 사용할 샘플의 개수: %d' % len(sequences))
```

```
학습에 사용할 샘플의 개수: 11
```

5)아직 레이블로 사용될 단어를 분리하지 않은 훈련 데이터

[2, 3]은 [경마장에, 있는]에 해당되며

[2, 3, 1]은 [경마장에, 있는, 말이]에 해당

전체 훈련 데이터에 대해서 맨 우측에 있는 단어에 대해서만 레이블로 분리해야 합니다.

```
[6] print(sequences)
```

```
[[2, 3], [2, 3, 1], [2, 3, 1, 4], [2, 3, 1, 4, 5], [6, 1], [6, 1, 7], [8, 1], [8, 1, 9], [8, 1, 9, 10], [8, 1, 9, 10, 1], [8, 1, 9, 10, 1, 11]]
```

```
[7] max_len = max(len(l) for l in sequences) # 모든 샘플에서 길이가 가장 긴 샘플의 길이 출력
    print('샘플의 최대 길이 : {}'.format(max_len))
```

```
샘플의 최대 길이 : 6
```

7)전체 샘플에 대해서 길이를 일치시켜 줍니다. 가장 긴 샘플의 길이를 기준으로 [8, 1, 9, 10, 1, 11]이고 길이는 6입니다.

패딩 이후 데이터를 훈련과 테스트로 분리

```
[8] sequences = pad_sequences(sequences, maxlen=max_len, padding='pre')
```

```
print(sequences)
```

```
[[ 0  0  0  0  2  3]
 [ 0  0  0  2  3  1]
 [ 0  0  2  3  1  4]
 [ 0  2  3  1  4  5]
 [ 0  0  0  0  6  1]
 [ 0  0  0  6  1  7]
 [ 0  0  0  0  8  1]
 [ 0  0  0  8  1  9]
 [ 0  0  8  1  9 10]
 [ 0  8  1  9 10  1]
 [ 8  1  9 10  1 11]]
```

8)모든 샘플의 길이를 6으로 맞춰주며, 'pre'는 앞에서부터 0으로 채운다.

```
[10] sequences = np.array(sequences)
     X = sequences[:,:-1]
     y = sequences[:,-1]
```

```
[11] print(X)
```

```
[[ 0  0  0  0  2]
 [ 0  0  0  2  3]
 [ 0  0  2  3  1]
 [ 0  2  3  1  4]
 [ 0  0  0  0  6]
 [ 0  0  0  6  1]
 [ 0  0  0  0  8]
 [ 0  0  0  8  1]
 [ 0  0  8  1  9]
 [ 0  8  1  9 10]
 [ 8  1  9 10  1]]
```

10)리스트의 마지막 값을 제외하고 저장한 것은 X, 리스트의 마지막 값만 저장한 것은 y.

```
[12] print(y) # 모든 샘플에 대한 레이
```

```
[ 3  1  4  5  1  7  1  9 10  1 11]
```

12)각 샘플의 마지막 단어를 레이블로 분리

13)RNN 모델 훈련을 위해서 레이블에 대해서 원-핫 인코딩을 수행합니다.

```
[ ] y = to_categorical(y, num_classes=vocab_size)
```

```
[14] print(y)
```

```
[[0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

3에 대한 원-핫 벡터

RNN 모델 설계하기

```
[17] from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Embedding, Dense, SimpleRNN
```

```
embedding_dim = 10
hidden_units = 32

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(SimpleRNN(hidden_units))
model.add(Dense(vocab_size, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history=model.fit(X, y, epochs=200, verbose=2)
```

임베딩 벡터의 차원은 10, 은닉 상태의 크기는 32입니다. 다 대 일 구조의 RNN을 사용합니다.

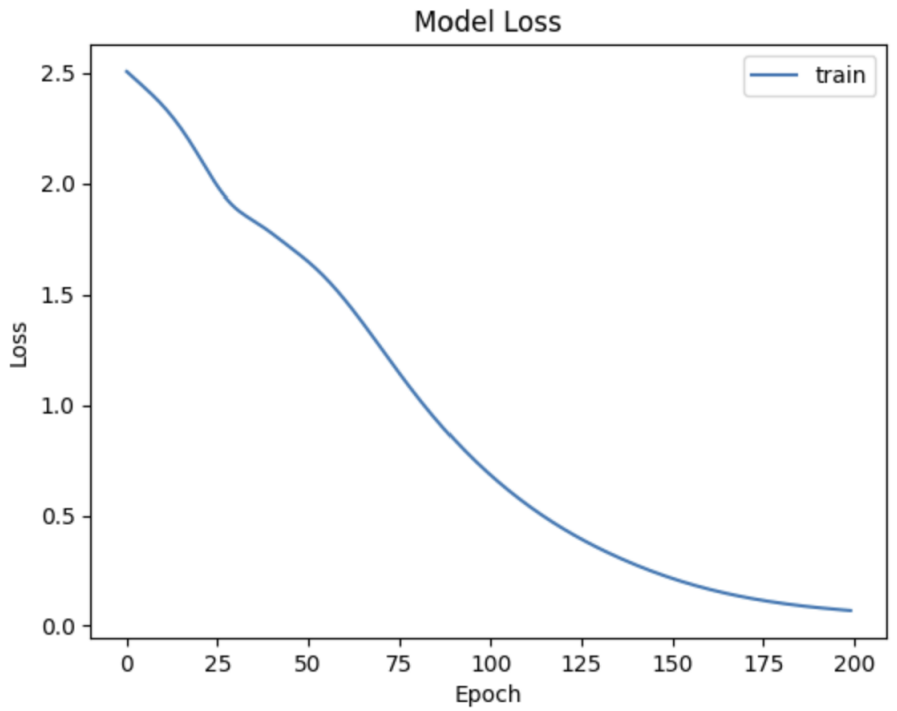
출력층으로 단어 집합 크기만큼의 뉴런을 배치하여 모델을 설계합니다.

다중 클래스 분류 문제로 출력층에 소프트맥스 회귀를 사용

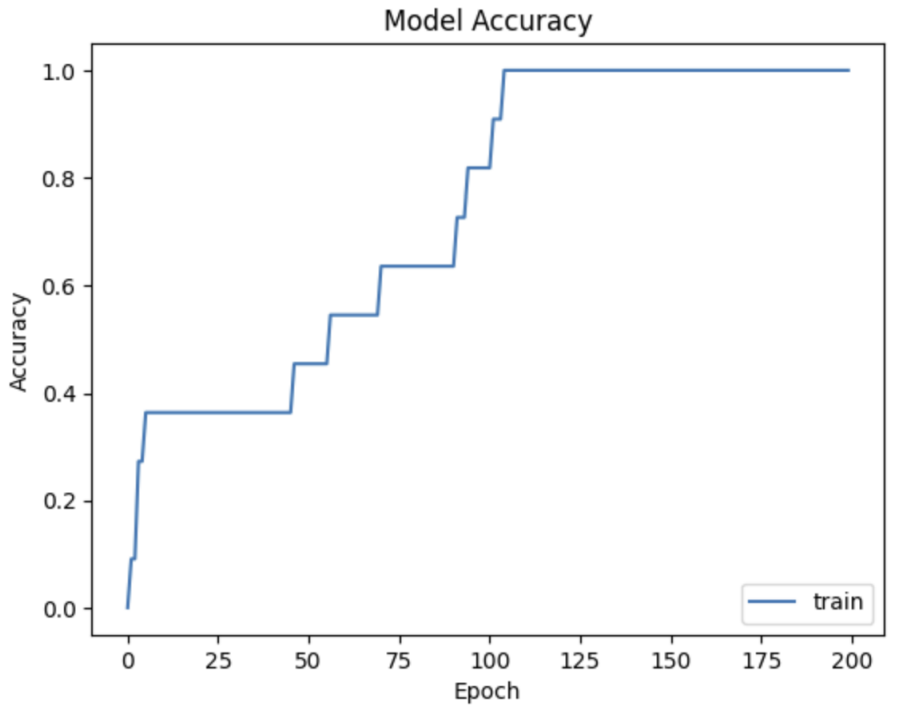
```
Epoch 199/200
1/1 - 0s - 58ms/step - accuracy: 1.0000 - loss: 0.0706
Epoch 200/200
1/1 - 0s - 57ms/step - accuracy: 1.0000 - loss: 0.0693
```

훈련 최적화 과정 가시화

```
[21] # 훈련 손실 그래프
     plt.plot(history.history['loss'])
     plt.title('Model Loss')
     plt.xlabel('Epoch')
     plt.ylabel('Loss')
     plt.legend(['train'], loc='upper right')
     plt.show()
```




```
# 훈련 정확도 그래프
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['train'], loc='lower right')
plt.show()
```



텍스트 생성 및 단어를 예측 RNN 모델

입력된 단어로부터 다음 단어를 예측해서 문장을 생성하는 함수

```
def sentence_generation(model, tokenizer, current_word, n):
    init_word = current_word
    sentence = ''
    for _ in range(n):
        encoded = tokenizer.texts_to_sequences([current_word])[0]
        encoded = pad_sequences([encoded], maxlen=5, padding='pre')
        result = model.predict(encoded, verbose=0)
        result = np.argmax(result, axis=1)
        for word, index in tokenizer.word_index.items():
            if index == result:
                break
        current_word = current_word + ' '  + word
        sentence = sentence + ' ' + word
    sentence = init_word + sentence
    return sentence
```

[24]
```
print(sentence_generation(model, tokenizer, '경마장에', 4))
```
경마장에 있는 말이 뛰고 있다

[25]
```
print(sentence_generation(model, tokenizer, '그의', 2))
```
그의 말이 법이다

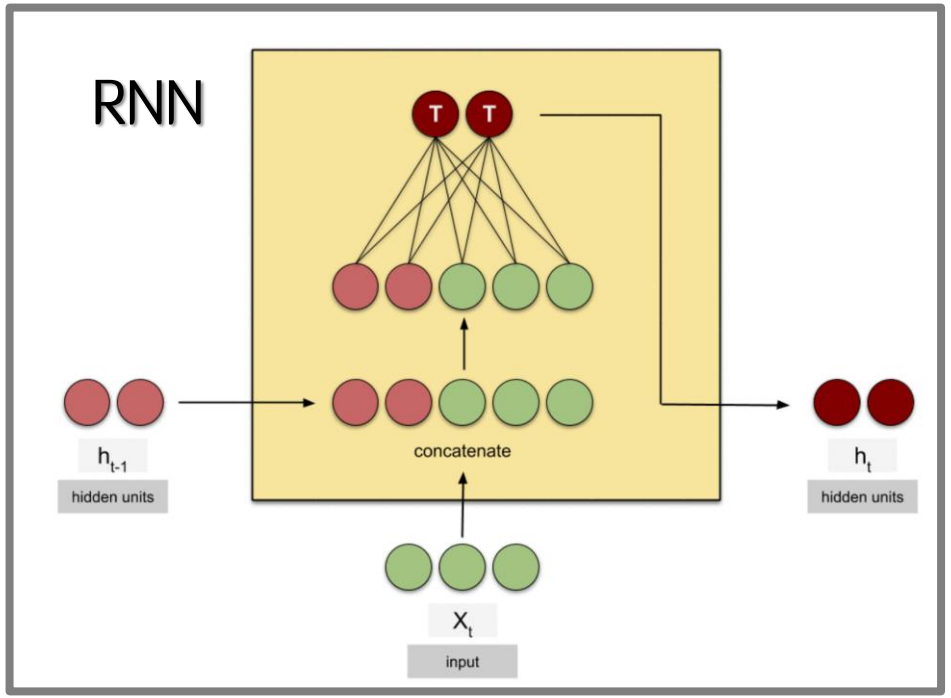
[26]
```
print(sentence_generation(model, tokenizer, '가는', 5))
```
가는 말이 고와야 오는 말이 곱다

Lecture 9

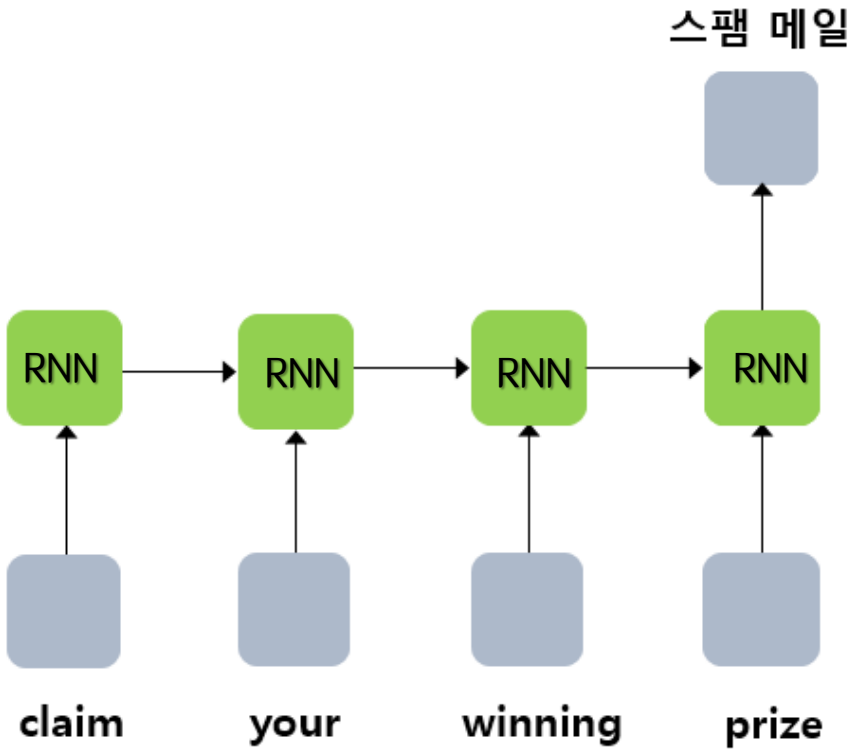
SimpleRNN 네이버 영화 리뷰 감성 분류 (실습)

테스트 입력에 따른 감성분류 모델

- 문서의 단어들을 순차적으로 입력받아 해당 문서의 유형을 판단하는 텍스트 분류에 사용될 수 있다.
  - ✓ 영화 리뷰 감성분류 (IMDB, 네이버 영화)
  - ✓ 쇼핑 리뷰 감성분류 (네이버 쇼핑리뷰)
  - ✓ 스팸메 일 분류



다-대-일(Many-to-One) RNN 모델



Colab에 한글 형태소 Okt 실습하기

```
[2]: import pandas as pd
     df=pd.read_csv("https://raw.githubusercontent.com/hongsukyi/Lectures/main/data/naver_shopping.txt", header=None,sep="\t")
```

```
[3]: df.columns = ['ratings', 'reviews']; df.head()
```

```
[1]: !pip install konlpy
     from konlpy.tag import Okt
```

[3]:

	ratings	reviews
0	5	배공빠르고 굿
1	2	택배가 엉망이네용 저희집 밑에층에 말도없이 놔두고가고
2	5	아주좋아요 바지 정말 좋아서2개 더 구매했어요 이가격에 대박입니다. 바느질이 조금 ...
3	2	선물용으로 빨리 받아서 전달했어야 하는 상품이었는데 머그컵만 와서 당황했습니다. 전...
4	5	민트색상 예뻐요. 옆 손잡이는 거는 용도로도 사용되네요 ㅎㅎ

- 교육용 데이터 크기를 줄이자. 15,000개

```
[4]: df=df[:15000]; len(df)
```

```
[4]: 15000
```

- ratings > 3 긍정(1), 2이하이면 부정(0)
- 넘파이 np.select
- 판다스에 새로운 컬럼 'label'을 추가함

```
[5]: import numpy as np
     df['label'] = np.select([df.ratings > 3], [1], default=0)
     df.head()
```

5]:

	ratings	reviews	label
0	5	배공빠르고 굿	1
1	2	택배가 엉망이네용 저희집 밑에층에 말도없이 놔두고가고	0
2	5	아주좋아요 바지 정말 좋아서2개 더 구매했어요 이가격에 대박입니다. 바느질이 조금 ...	1
3	2	선물용으로 빨리 받아서 전달했어야 하는 상품이었는데 머그컵만 와서 당황했습니다. 전...	0
4	5	민트색상 예뻐요. 옆 손잡이는 거는 용도로도 사용되네요 ㅎㅎ	1

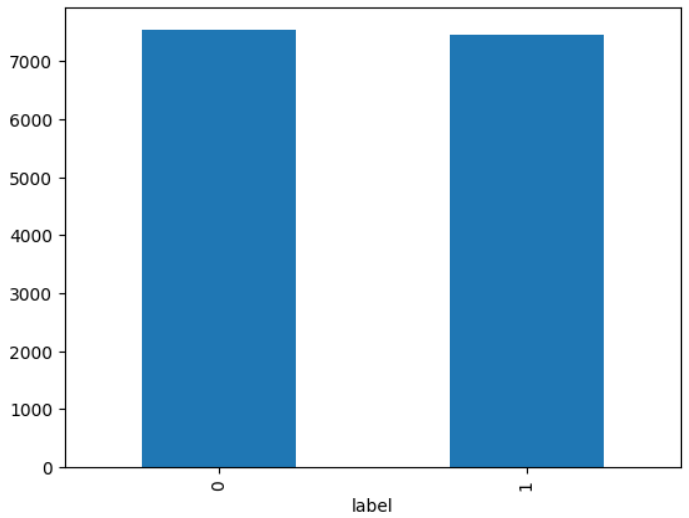
한글 OKT 형태소 분석기로 토큰화

```
[6]: df['ratings'].nunique(), df['reviews'].nunique(), df['label'].nunique()
```

```
[6]: (4, 14999, 2)
```

```
[7]: df['label'].value_counts().plot(kind = 'bar')
```

```
[7]: <Axes: xlabel='label'>
```



한글과 공백을 제외하고 모두 제거

```
[8]: df['reviews'] = df['reviews'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
```

2. 한글 형태소 Okt 활용하기

```
[9]: okt = Okt()
     print(okt.morphs('택배가 엉망이네용 저희집 밑에층에 말도없이 놔두고가고'))
```

```
['택배', '가', '엉망', '이네', '용', '저희', '집', '밑', '에', '층', '에', '말',
```

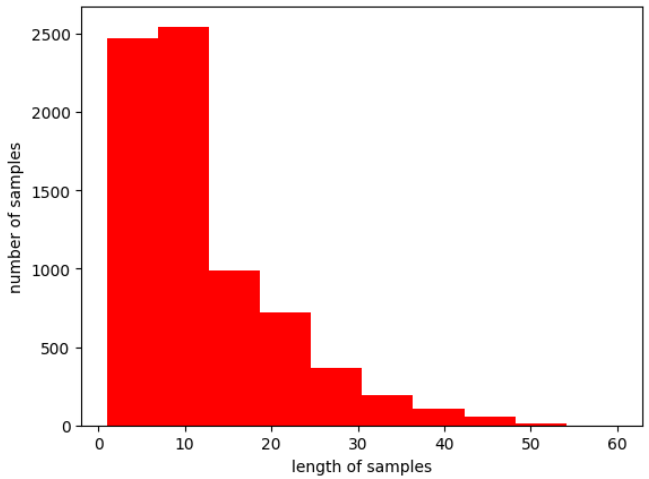
```
[10]: stopwords = ['도','는','다','의','가','이','은','한','에','하','고','을','를','인','듯','과','와','
```

```
[11]: df['tokenized'] = df['reviews'].apply(okt.morphs)
      df['tokenized'] = df['tokenized'].apply(lambda x: [item for item in x if item not in stopwords])
```

```
[12]: df['tokenized'].head()
```

```
[12]: 0                                  [배공, 빠르고, 굿]
      1    [택배, 엉망, 이네, 용, 저희, 집, 밑, 층, 말, 없이, 놔두고가고]
```

긍정 리뷰의 평균 길이 : 11.890289699570815



케라스를 이용한 리뷰 단어를 정수인코딩 및 토큰화

KISTI SINCE 1962

[14]:
```
X_train = df['tokenized'].values
y_train = df['label'].values
```

[15]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
```

[16]:
```
vocab_size = 8000
tokenizer = Tokenizer(vocab_size, oov_token = 'OOV')
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
```

[17]:
```
print(X_train[:3])
```

```
[[5843, 55, 281], [187, 612, 116, 99, 661, 166, 459, 1986, 320, 167, 1], [76, 8, 624, 49, 231, 94, 44, 34, 12, 56, 21, 802,
27, 2, 1280, 71, 702, 1, 269, 225, 158, 109, 253, 124, 2]]
```

4. 토큰화된 텍스트를 패딩(padding)하기

[18]:
```
plt.hist([len(s) for s in X_train], bins=50)
plt.show()
```

케라스 패딩: 모든 입력 데이터의 리뷰 단어 개수를 같게한다.

KISTI SINCE 1962

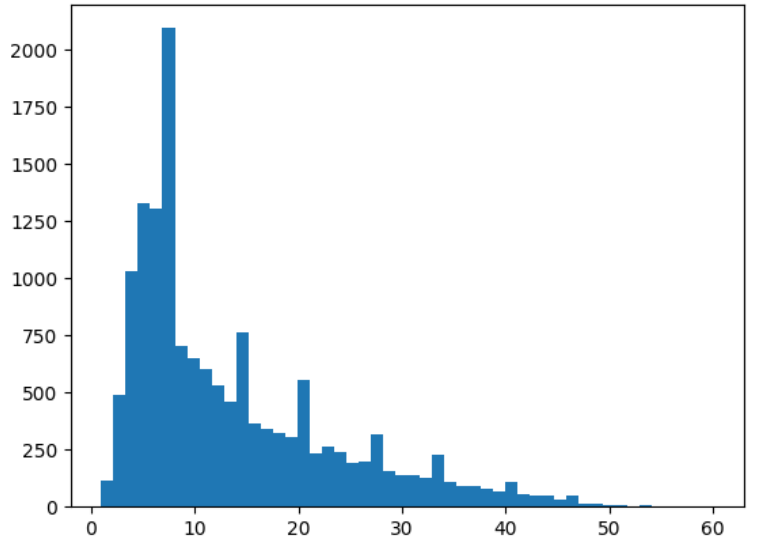
[19]:
```
max_len = 60   # 한 문장에서 리뷰 길이(개수)는 60이면 충분하다.
X_train = pad_sequences(X_train, maxlen = max_len)
```

- max_len = 50 : 리뷰 문장 1개의 고정 길이(패딩)
- vocab_size = 5,000 : BoW의 길이
- 샘플의 길이 = 2,000 -X_train.shape의 크기는 (2000, 50)이다.

[20]:
```
X_train.shape
```

[20]:
```
(15000, 60)
```

[18]:
```
plt.hist([len(s) for s in X_train], bins=50)
plt.show()
```



[21]:
```
X_train[:1], y_train[:1]
```

[21]:
```
(array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0, 5843,   55,  281]]),
 array([1]))
```

[3] SimpleRNN을 이용한 네이버 쇼핑 리뷰 분류하기

[22]:
```
from tensorflow.keras.layers import Embedding, Dense, SimpleRNN, GRU, LSTM, Flatten
from tensorflow.keras.models import Sequential
```

[23]:
```
emb_dim = 128
rnn_hiddens = 64
```

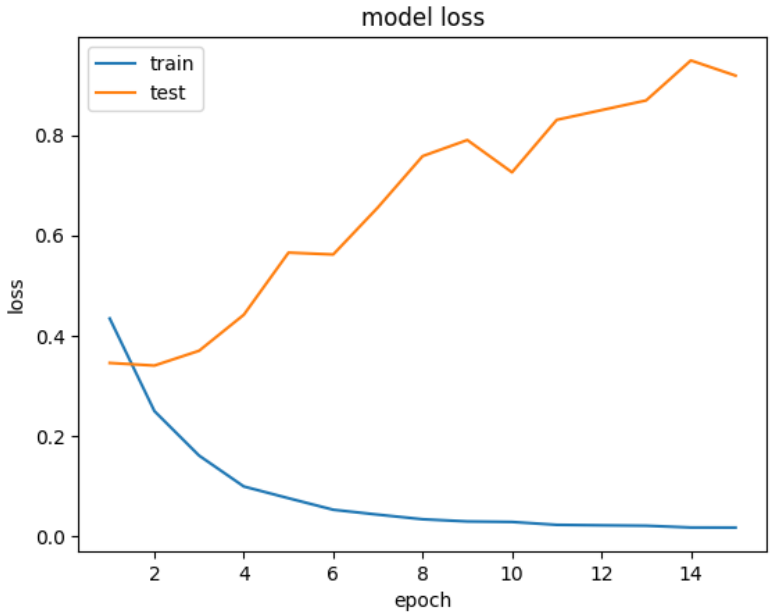
[24]:
```
model = Sequential()
model.add(Embedding(vocab_size, emb_dim))
model.add(SimpleRNN(rnn_hiddens))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
```

[25]:
```
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=15, batch_size=32, validation_split=0.2)
```

```
Epoch 1/15
375/375 ━━━━━━━━━━━━━━━━━━━━ 9s 18ms/step - acc: 0.7223 - loss: 0.5349 - val_acc: 0.8587 - val_loss: 0.3453
Epoch 2/15
375/375 ━━━━━━━━━━━━━━━━━━━━ 7s 18ms/step - acc: 0.9064 - loss: 0.2503 - val_acc: 0.8700 - val_loss: 0.3401
Epoch 3/15
375/375 ━━━━━━━━━━━━━━━━━━━━ 7s 18ms/step - acc: 0.9446 - loss: 0.1612 - val_acc: 0.8637 - val_loss: 0.3697
Epoch 4/15
375/375 ━━━━━━━━━━━━━━━━━━━━ 7s 18ms/step - acc: 0.9706 - loss: 0.0952 - val_acc: 0.8547 - val_loss: 0.4415
```

[26]:
```
model.summary()
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(32, 60, 128)	1,024,000
simple_rnn (SimpleRNN)	(32, 64)	12,352
flatten (Flatten)	(32, 64)	0
dense (Dense)	(32, 1)	65



테스트 과정 및 모델 평가

KISTI SINCE 1962

[50]:
```
import re
def sentiment_predict(new_sentence):
  new_sentence = re.sub(r'[^ㄱ-ㅎㅏ-ㅣ가-힣 ]','', new_sentence)
  new_sentence = okt.morphs(new_sentence) # 토큰화
  new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
  encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
  pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩

  score = float(model.predict(pad_new)) # 예측
  if(score > 0.5):
    print("{:.2f}% 확률로 긍정 리뷰입니다.".format(score * 100))
  else:
    print("{:.2f}% 확률로 부정 리뷰입니다.".format((1 - score) * 100))
```

[53]:
```
sentiment_predict('판매자님... 너무 짱이에요.. 대박나삼')
```
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
95.45% 확률로 부정 리뷰입니다.
```

[54]:
```
sentiment_predict('ㅁㄴㅇㄹㄴㅇㄹㄴㅇ리뷰쓰기도 귀찮아')
```
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
92.45% 확률로 긍정 리뷰입니다.
```

[51]:
```
sentiment_predict('이 상품 진짜 좋아요... 저는 강추합니다. 대박')
```
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step
97.80% 확률로 긍정 리뷰입니다.
```

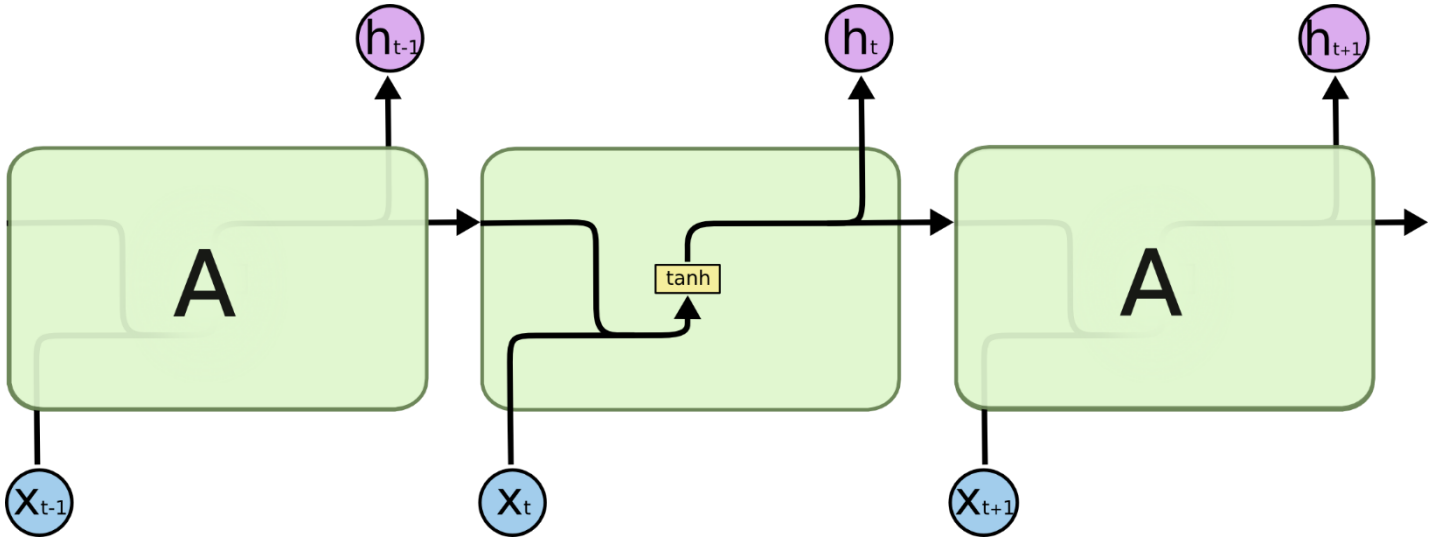
[52]:
```
sentiment_predict('진짜 배송도 늦고 개짜증나네요. 뭐 이런 걸 상품이라고 만듬?')
```
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
98.20% 확률로 부정 리뷰입니다.
```

Lecture 10

LSTM (Long Short-Term Memory) 모델 소개

Long Short-Term Memory, LSTM

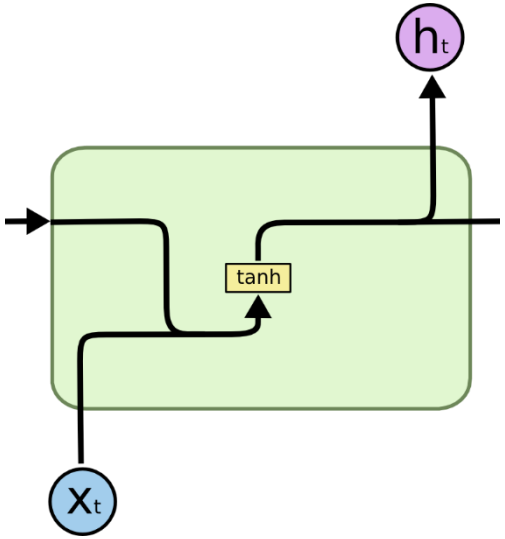
- 기존 RNN(바닐라 RNN)의 장기 의존성 문제를 개선하여 기억력을 높인 RNN의 명칭.
- 앞으로 나오는 설명에서 RNN을 사용한다고 하면 기본적으로 LSTM(또는 GRU)를 사용한다고 가정한다.



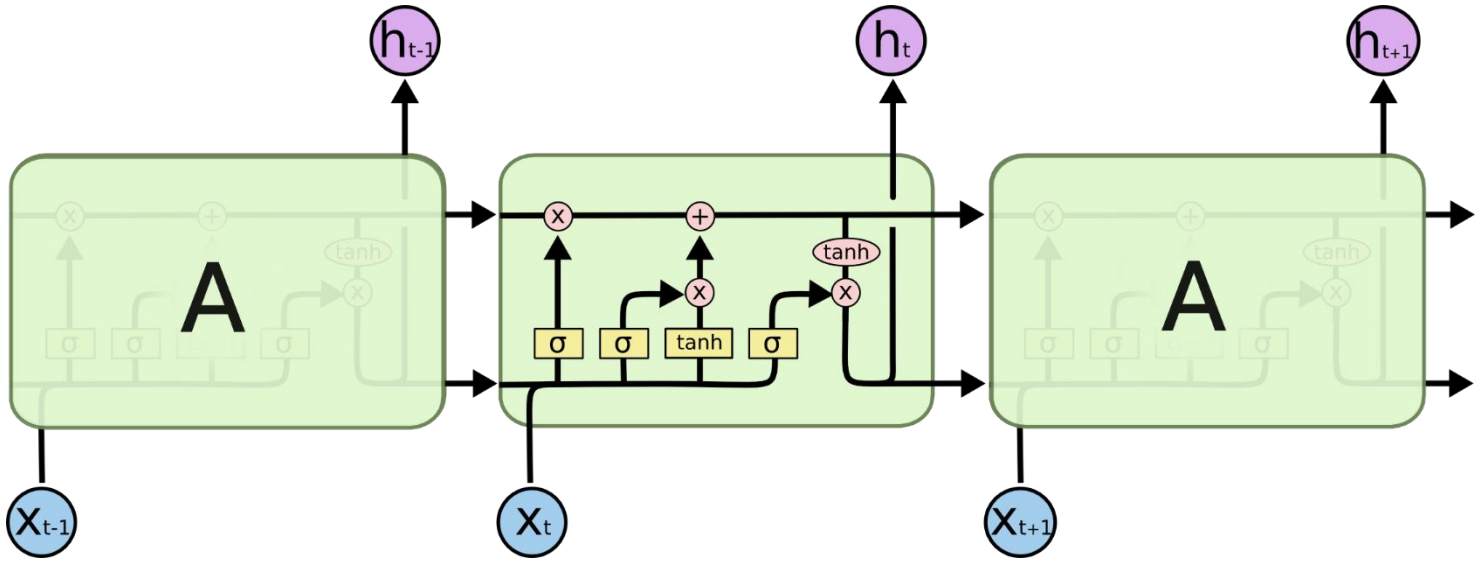
Vanilla RNN

Long Short-Term Memory, LSTM

- 기존 RNN(바닐라 RNN)의 장기 의존성 문제를 개선하여 기억력을 높인 RNN의 명칭.
- 앞으로 나오는 설명에서 RNN을 사용한다고 하면 기본적으로 LSTM(또는 GRU)를 사용한다고 가정한다.



Vanilla RNN



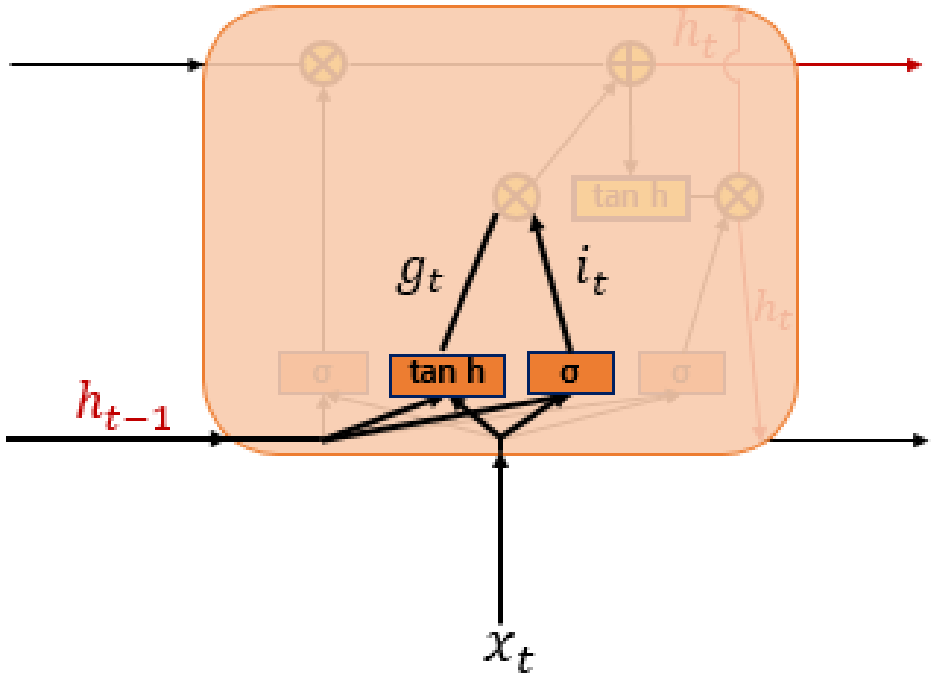
LSTM(Long Short-Term Memory)

입력 게이트(Input Gate)

❖ 입력 게이트는 현재 정보를 기억하기 위한 게이트이다.

✓ 시그모이드 함수를 지나 0과 1 사이의 값 and 하이퍼볼릭탄젠트 함수를 지나 -1과 1사이의 값이 두 개의 값

✓ 이 두 가지 값을 가지고 Cell state에서 이번에 선택된 기억할 값을 정한다.



Input Gate

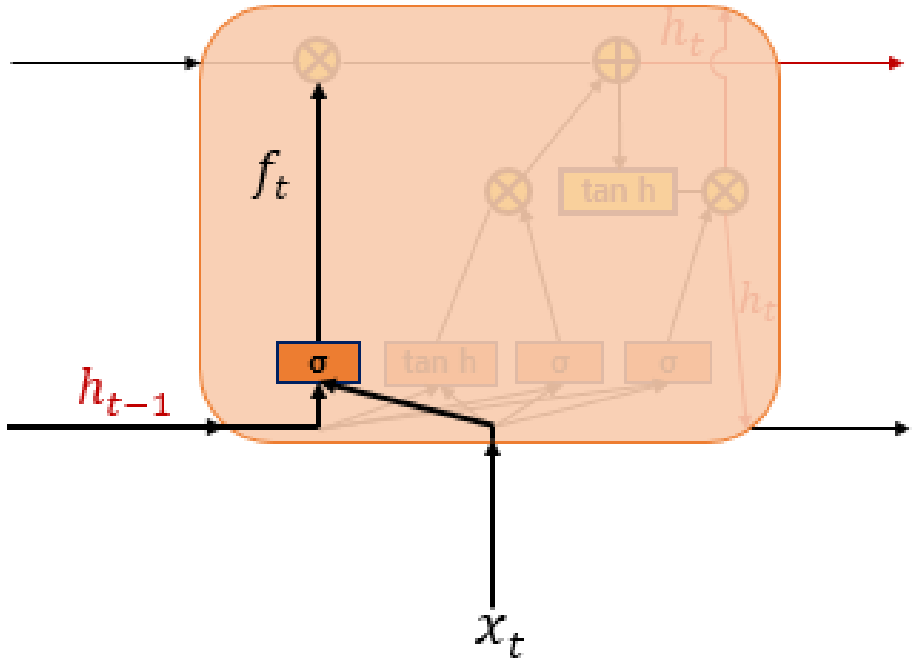
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$g_t = tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

삭제 게이트(Forget Gate)

❖ 삭제 게이트는 기억을 삭제하기 위한 게이트이다.

✓ 시그모이드 함수를 지나 0과 1 사이의 값이 나온다.

✓ 0에 가까울수록 정보가 많이 삭제된 것이며, 1에 가까울수록 정보를 온전히 기억한 셈이다.

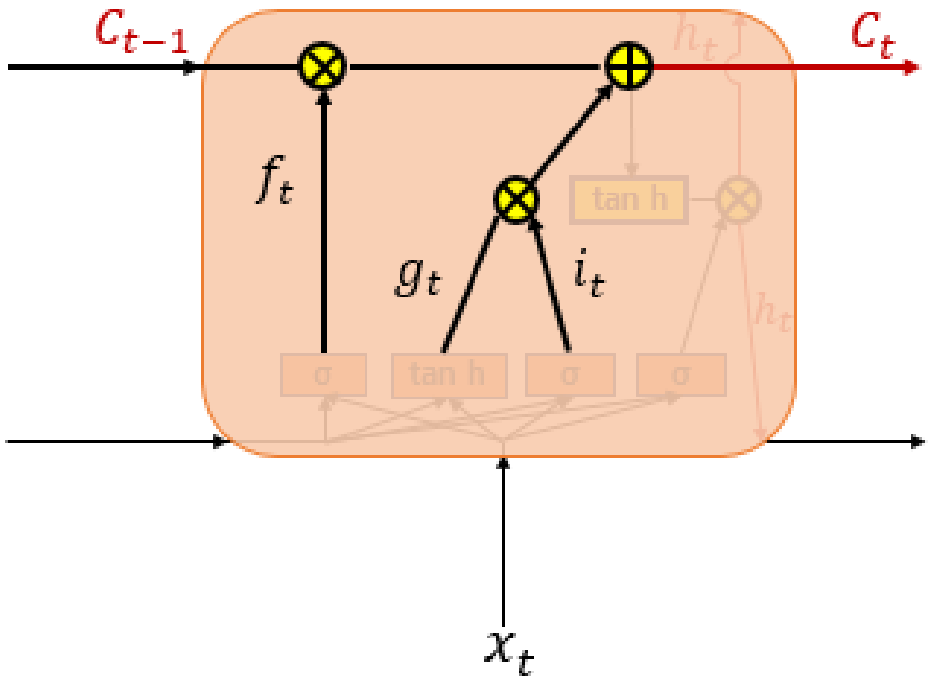


Forget Gate

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

셀 상태(Cell state) or 장기 상태

- **삭제 게이트에서 일부 기억을 소실.**
  - ✓ 입력 게이트의 i_t와 g_t를 가지고 elementiwise product를 수행 : 이번에 기억할 값.



Cell State

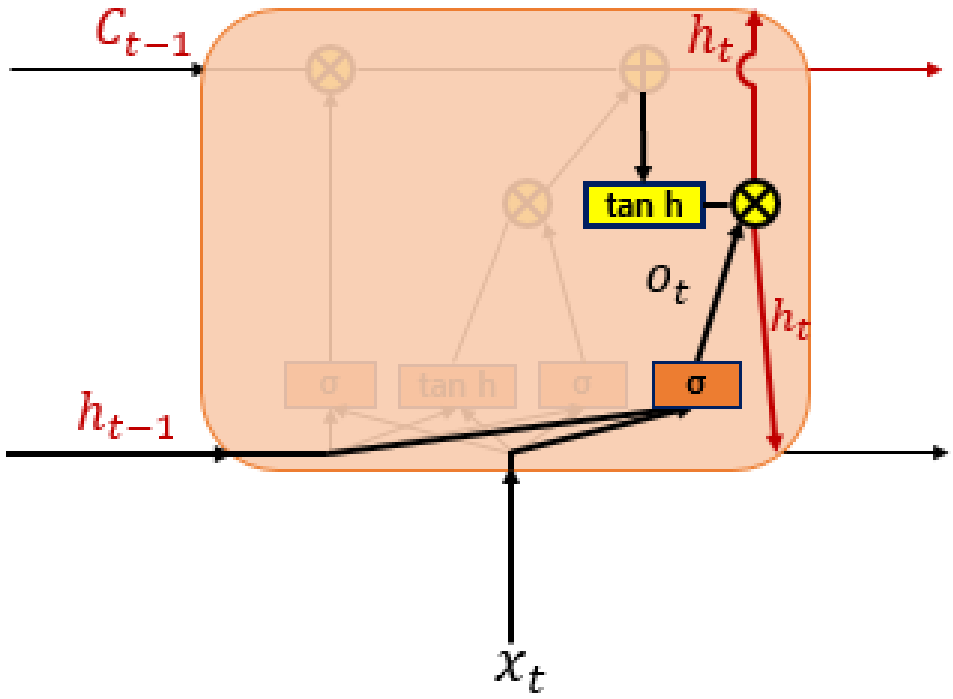
$$C_t = f_t \circ C_{t-1} + i_t \circ g_t$$

i_t가 0이 된다면 이전 시점의 Cell state값으로 현재 시점의 Cell state값을 결정한다.

f_t가 0이 된다면 오직 입력 게이트만이 현재 시점의 Cell state값을 결정한다.

출력 게이트(Output Gate)

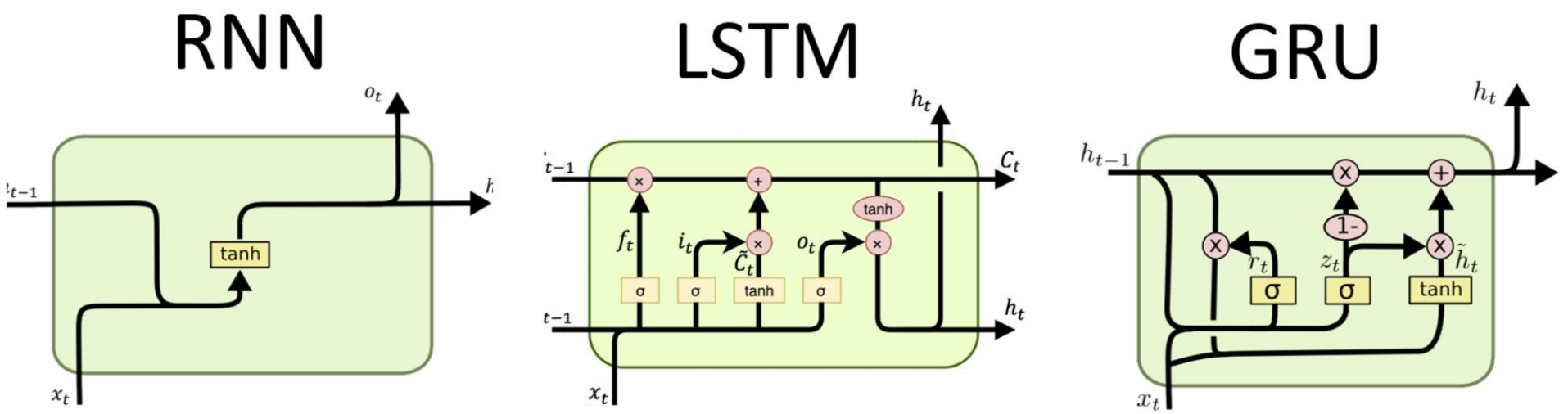
- 출력 게이트는 Hidden State를 연산하는 일에 쓰인다.
- Hidden State는 Cell State와 비교하여 단기 상태라고도 부른다.



$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$

Output Gate / Hidden State

- RNN과 유사 RNN
  - ✓ Vanilar RNN
  - ✓ Long Short-Term Memory (LSTM)
  - ✓ Gated Recurrent Unit (GRU) they handle the state updates differently



Lecture 11

LSTM을 이용한 네이버 쇼핑 리뷰 분류

LSTM을 이용한 모델 훈련

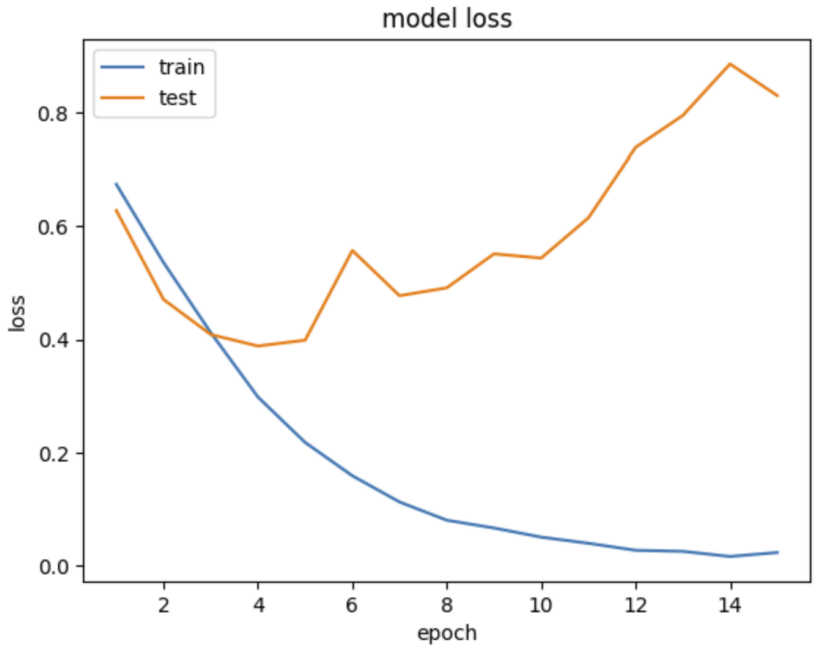
```
[33] from tensorflow.keras.layers import Embedding, Dense, GRU, LSTM
     from tensorflow.keras.models import Sequential
```

```
[34] model = Sequential()
     model.add(Embedding(vocab_size, 100))
     model.add(GRU(128))
     model.add(Dense(1, activation='sigmoid'))
```

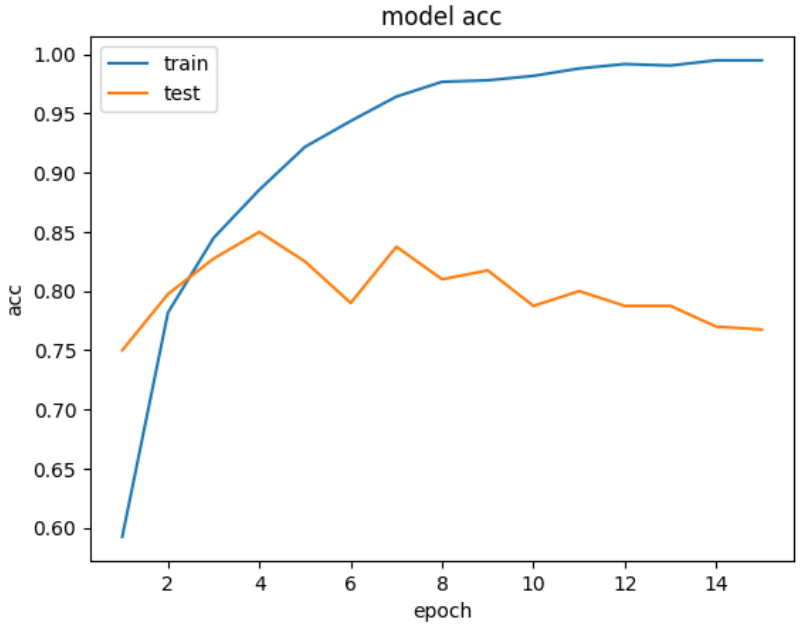
```
[35] model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
     history = model.fit(X_train, y_train, epochs=15, batch_size=60, validation_split=0.2)
```

훈련과정 Loss와 Accuracy (샘플 2,000개)

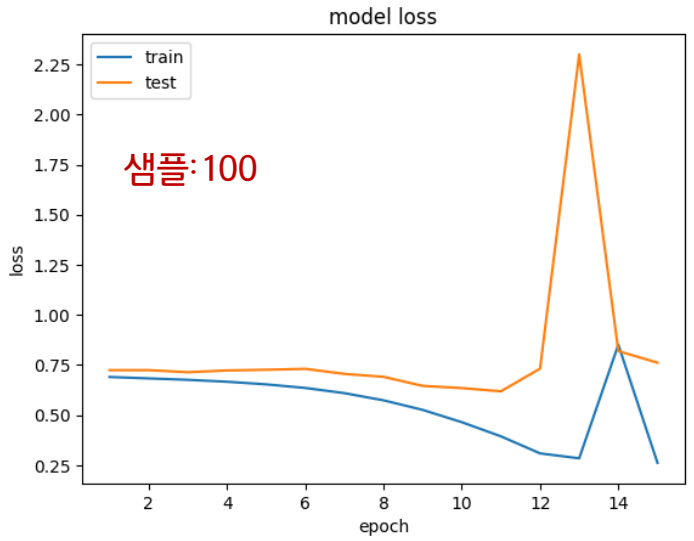
```
epochs = range(1, len(history.history['acc']) + 1)
plt.plot(epochs, history.history['loss'])
plt.plot(epochs, history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

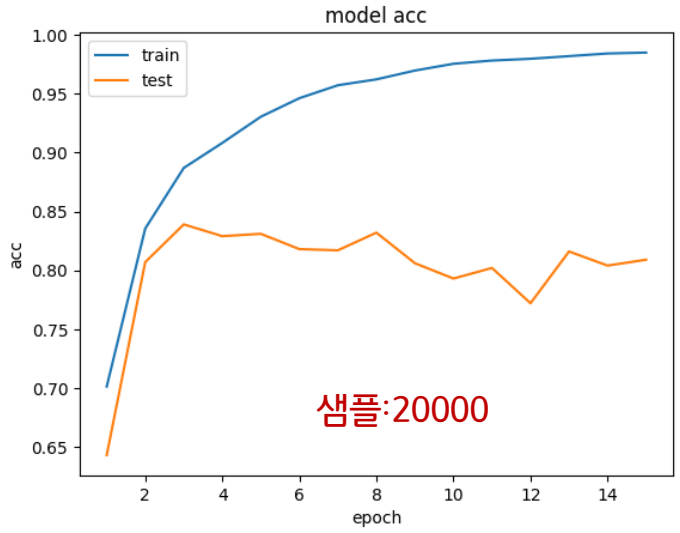


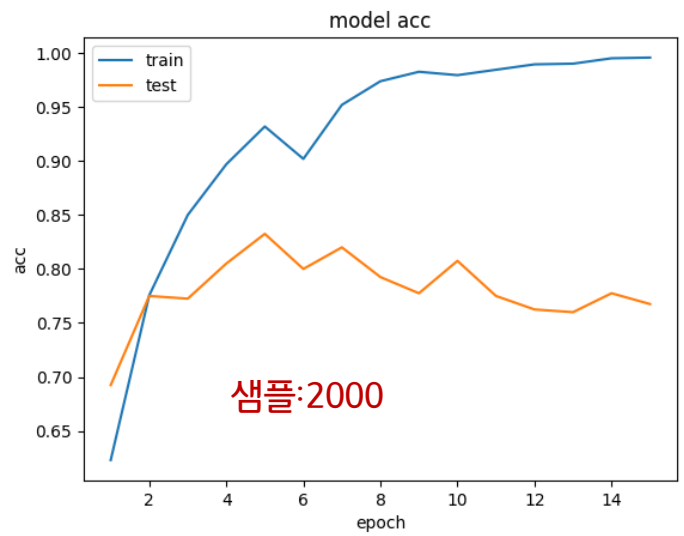
[84]
```
epochs = range(1, len(history.history['acc']) + 1)
plt.plot(epochs, history.history['acc'])
plt.plot(epochs, history.history['val_acc'])
plt.title('model acc')
plt.ylabel('acc')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

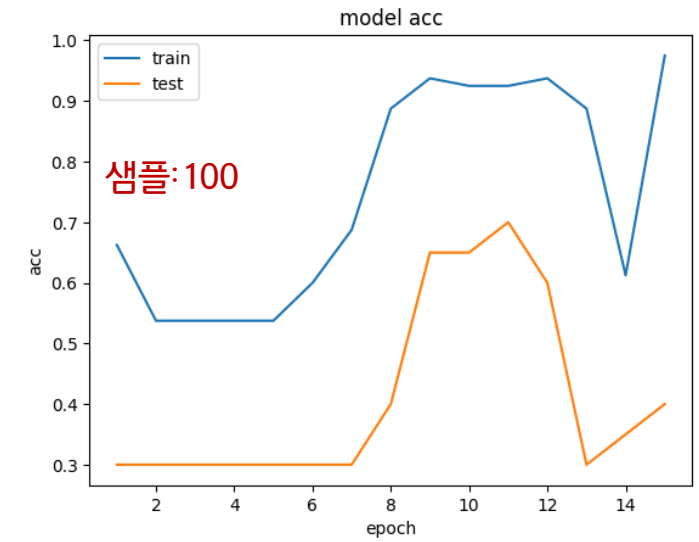


LSTM 학습 결과 : 손실 함수와 정확도









리뷰 예측해보기

데이터를 훈련과 테스트 데이터를 나눔.
테스트 데이터의 NLP 전처리가 필요하다. sentiment_predict() 함수에 전처리가 있다.

```
[36] import re
     def sentiment_predict(new_sentence):
       new_sentence = re.sub(r'[^ㄱ-ㅎㅏ-ㅣ가-힣 ]','', new_sentence)
       new_sentence = okt.morphs(new_sentence) # 토큰화
       new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
       encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
       pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩

       score = float(model.predict(pad_new)) # 예측
       if(score > 0.5):
         print("{:.2f}% 확률로 긍정 리뷰입니다.".format(score * 100))
       else:
         print("{:.2f}% 확률로 부정 리뷰입니다.".format((1 - score) * 100))
```

모델 평가하기

```
[79] sentiment_predict('이 상품 진짜 좋아요... 저는 강추합니다. 대박')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 180ms/step
99.95% 확률로 긍정 리뷰입니다.
```

```
[80] sentiment_predict('진짜 배송도 늦고 개짜증나네요. 뭐 이런 걸 상품이라고 만듬?')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
53.55% 확률로 부정 리뷰입니다.
```

```
sentiment_predict('판매자님... 너무 짱이에요.. 대박나삼')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
92.75% 확률로 긍정 리뷰입니다.
```

```
[82] sentiment_predict('ㅁㄴㅇㄹㅁㄴㅇㄹㅁㄴㅇ리뷰쓰기도 귀찮아')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
95.56% 확률로 부정 리뷰입니다.
```

KISTI SINCE 1962

감사합니다.

TRUST

KISTI

Korea Institute of Science
and Technology Information