

딥러닝 #4 RNN

오늘

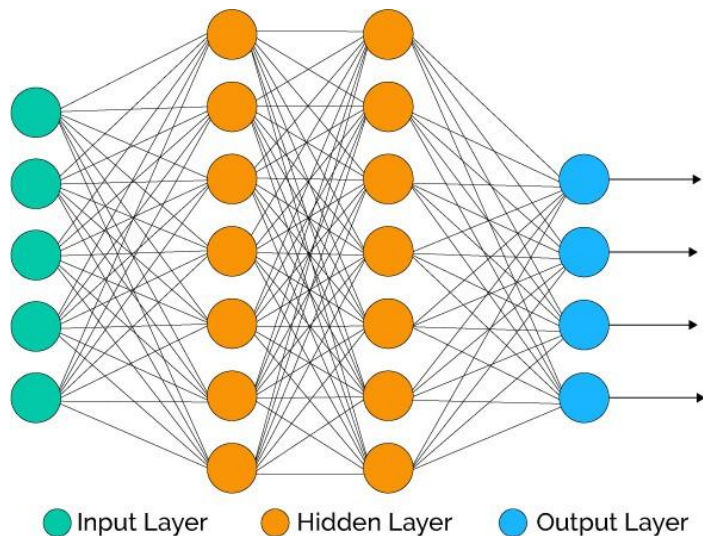
1. Sequential Data를 이해한다.
2. RNN의 기본틀과 기본흐름을 이해한다.
3. 수식, 데이터, 다이어그램, 코드 사이의 표현을 이해한다.
4. Sigmoid 활성화 함수를 재발견한다.

Sequential Data

- sequential data – 순서가 의미가 있는 데이터
 - temporal sequence – 시간적 의미가 있는 데이터
 - time series – 일정한 주기가 존재하는 데이터
- resampling을 통해 temporal sequence를 time series로 변환할 수 있다.

신경망의 한계

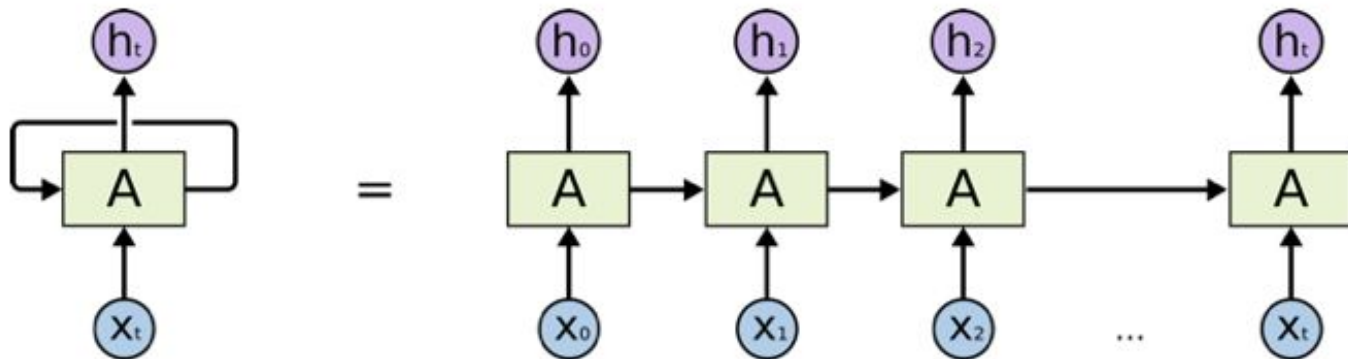
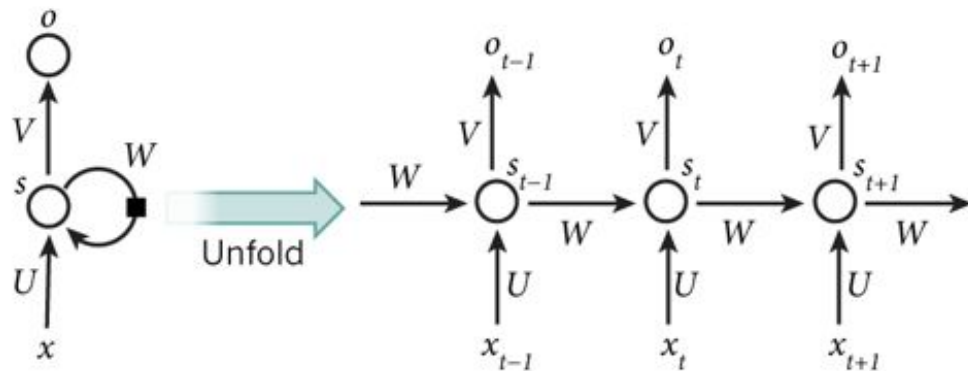
- 신경망은 자기가 ‘이전에 뭘 했는지’에 관심이 없다
- 입력 데이터 길이가 매번 바뀌는 경우에 대응할 수 없다.
- 분류가 무수히 많은 경우 매핑이 어렵다.



What makes RNN so special?

- CNN은 제한적이다.
 - 고정된 크기의 인풋 (이미지 등)
 - 고정된 크기의 아웃풋
 - 고정된 크기의 계산 절차 (layer 갯수가 고정)
- ‘안’ 고정된 것을 다뤄보고 싶다
 - 쓰다 만 ‘문장’ 이라든지
 - 그리다 만 ‘그림’ 이라든지
 - 연주하다 만 ‘노래’ 라든지

Vanilla RNN

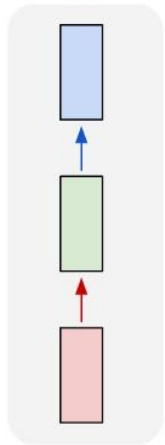


RNN code

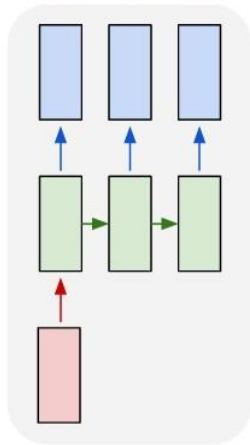
```
1 import tensorflow as tf
2
3 # vanilla RNN
4 model = tf.keras.models.Sequential()
5 model.add(tf.keras.layers.Input(shape = (7, 5)))
6
7 cell = tf.keras.layers.SimpleRNNCell(15)
8 model.add(tf.keras.layers.RNN(cell))
9
10 model.add(tf.keras.layers.Dense(1))
11
12 model.compile(
13     loss=tf.keras.losses.mse,
14     optimizer=tf.keras.optimizers.Adam(lr=0.01)
15 )
```

RNN 구성사례

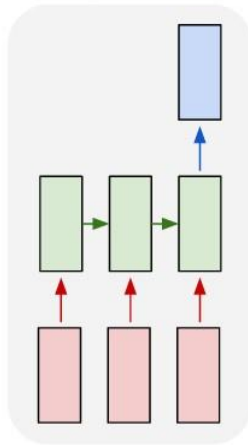
one to one



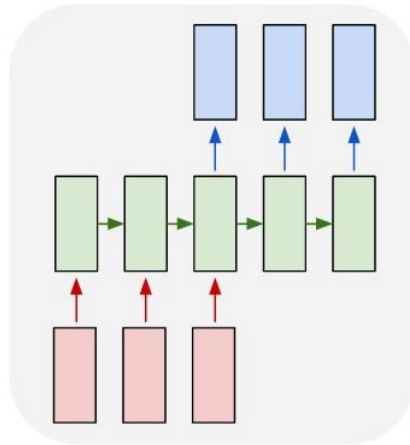
one to many



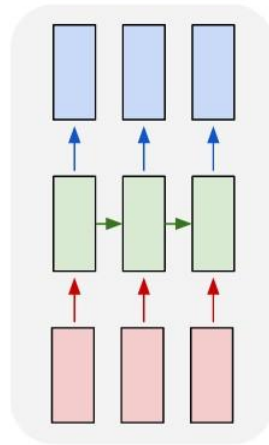
many to one



many to many



many to many



고정된 갯수의 layer를 재사용하므로 입출력단의 순열(Sequence) 길이를 미리 제한할 필요가 없다

RNN 구성사례

- one to one: 이미지 입력→이미지 분류 (개 사진/고양이 사진)
- one to many: 이미지 입력→이미지 Captioning (문장)
- many to one: 문장 입력→감정분석 결과 (긍정/부정)
- many to many: 문장 입력→문장 출력 (기계 번역)
- many to many: 비디오 입력→(동기화된)결과 출력 (비디오 프레임별 분류)

Vanilla RNN 기본 개념

We can process a sequence of vectors \mathbf{x} by applying a recurrence formula at every time step:

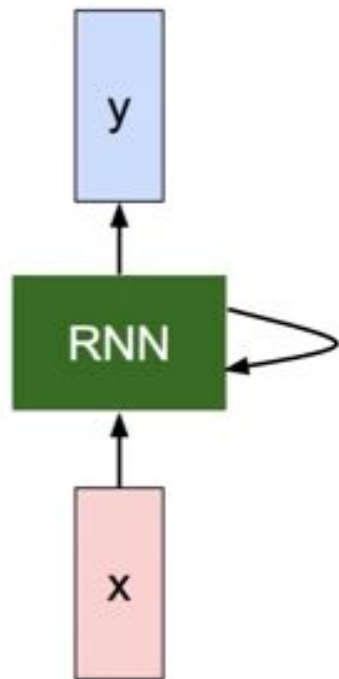
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

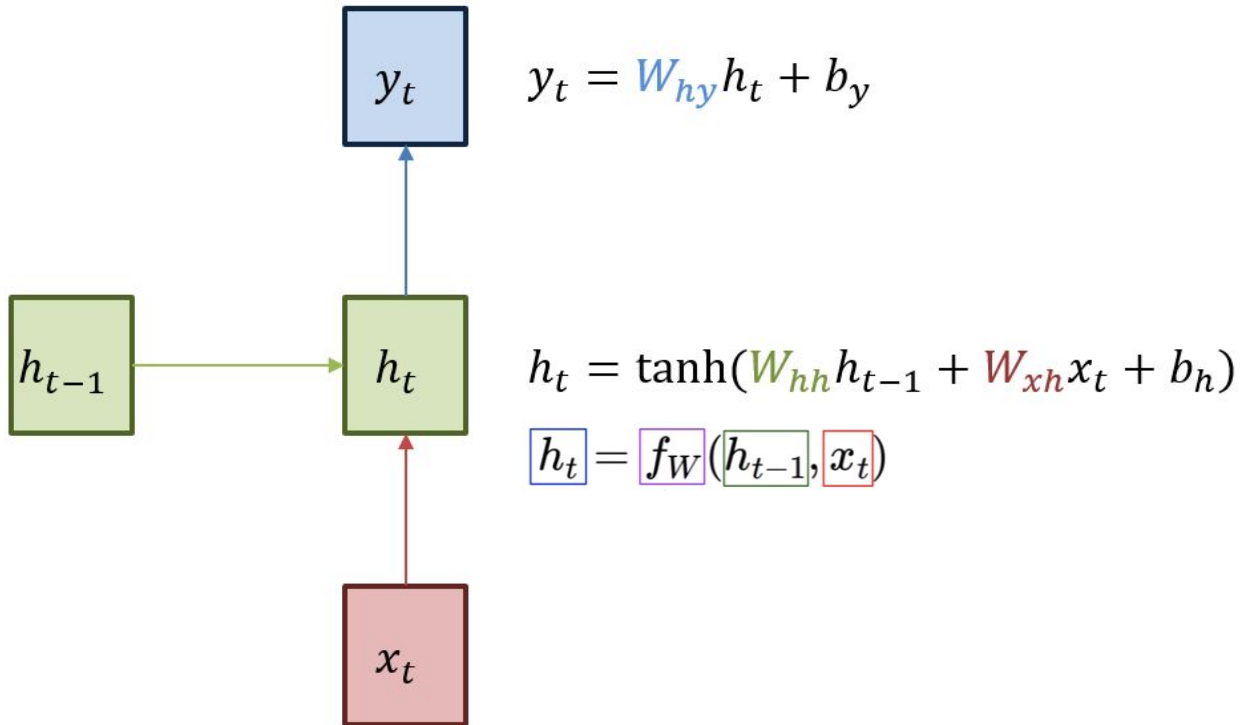
some function with parameters W

old state

input vector at some time step

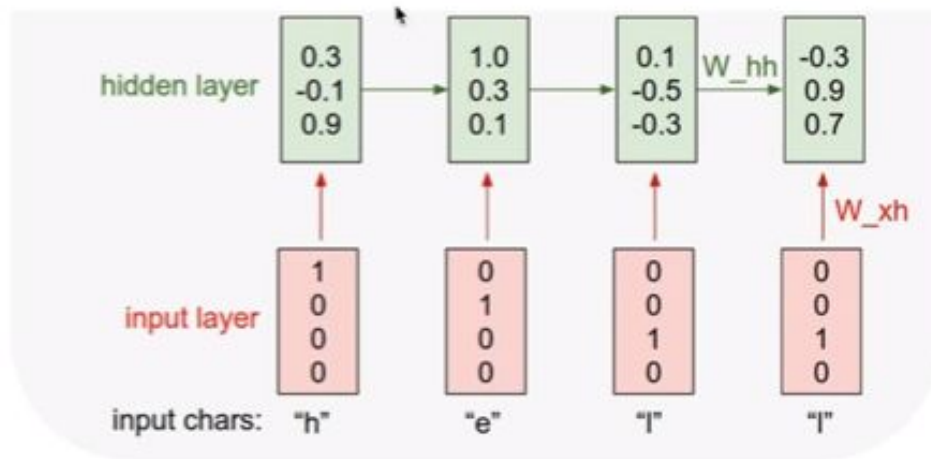


RNN 기본 개념



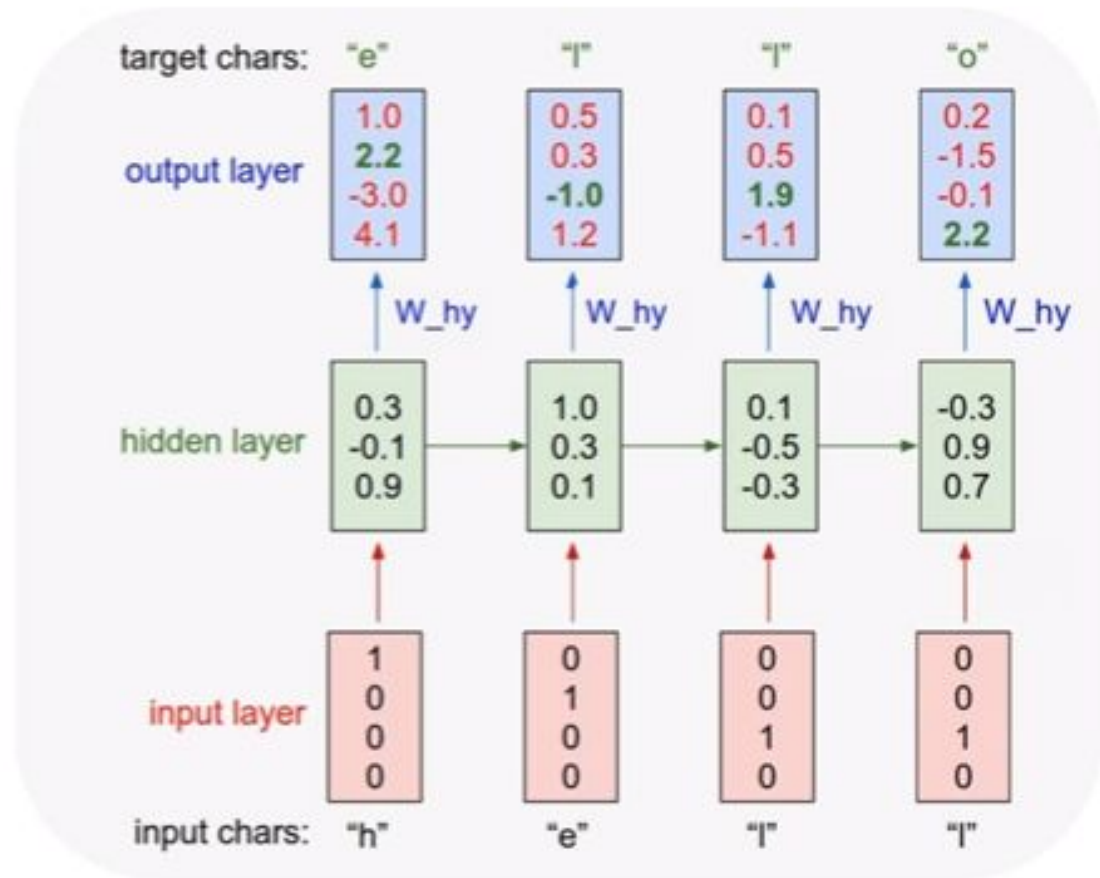
RNN 입력

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



RNN 출력

$$y_t = W_{hy}h_t$$

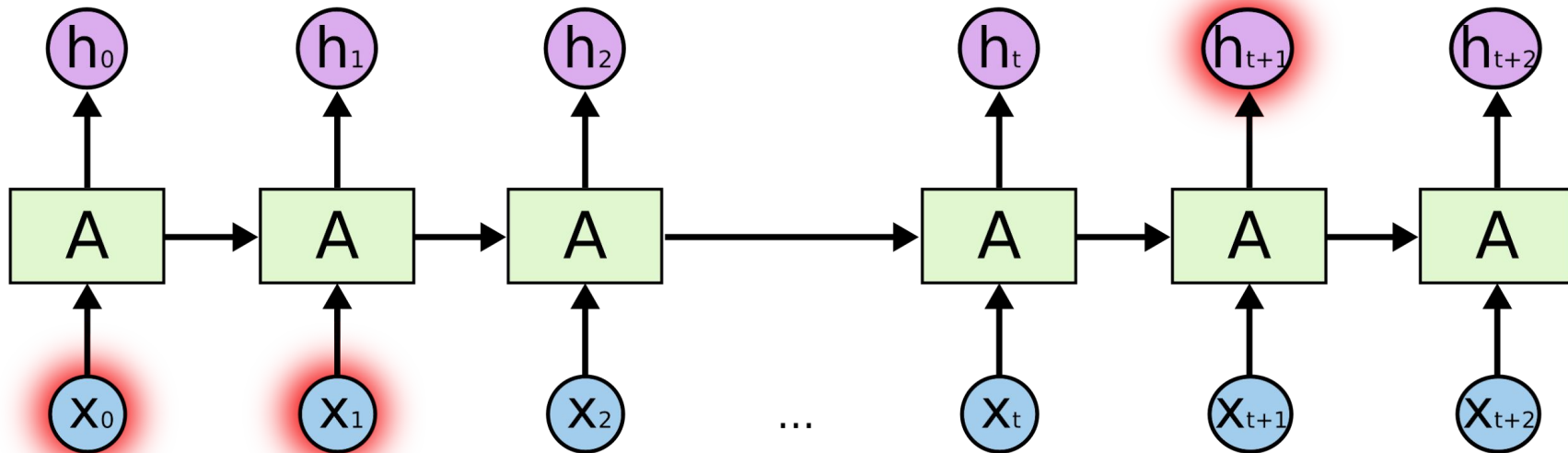


RNN의 기울기 소실 문제



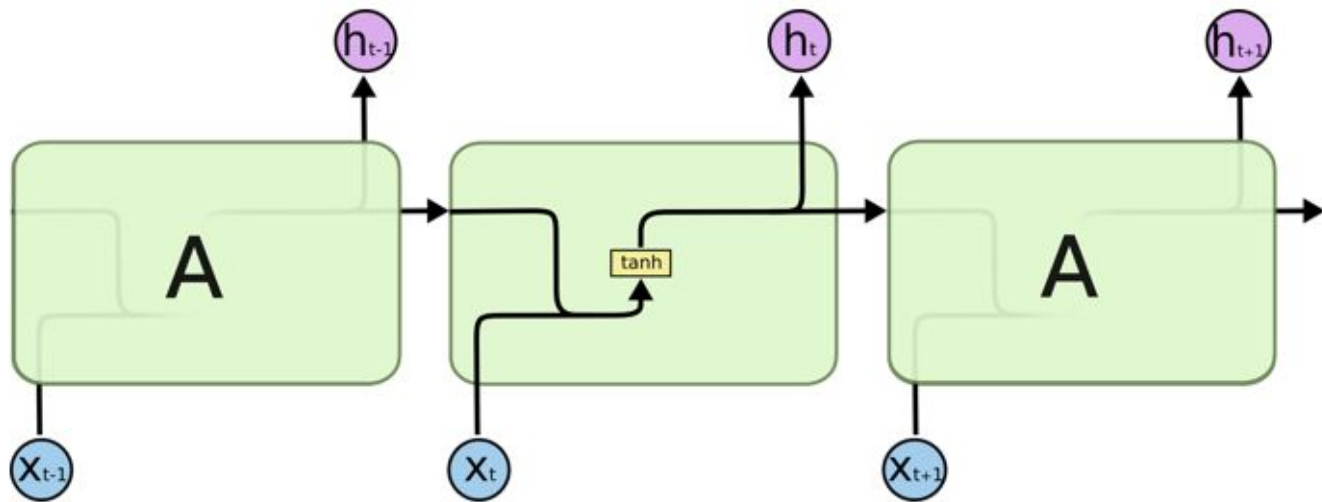
바로 잠깐 전에 했던 말만 기억한다

‘시간’이 오래 지나면 예전 정보가 ‘희석’되어 사라지는 문제



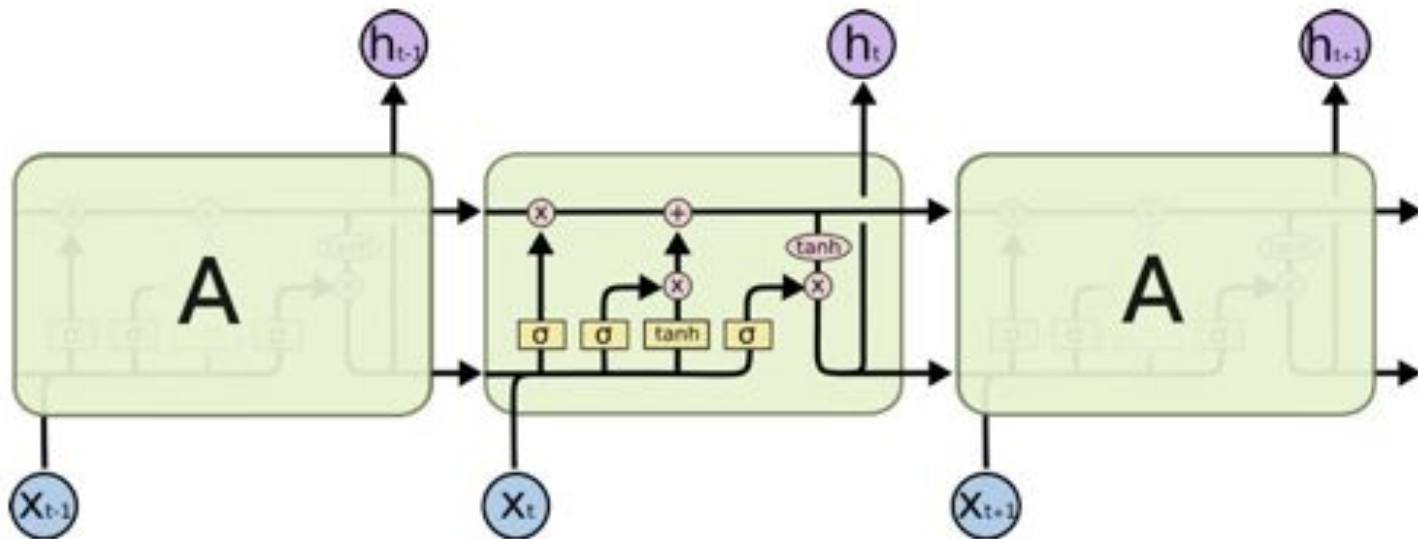
RNN의 기울기 소실 문제

원인은 \tanh , 중첩되면 0에 수렴한다

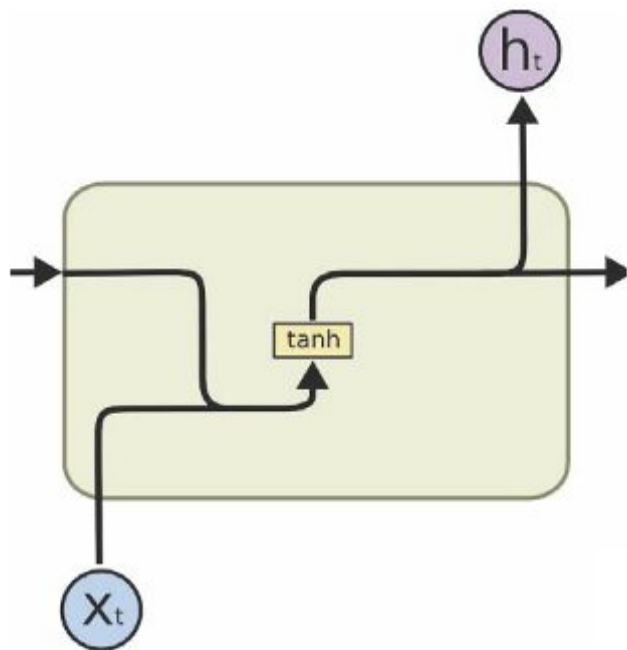


LSTM (long short term memory)

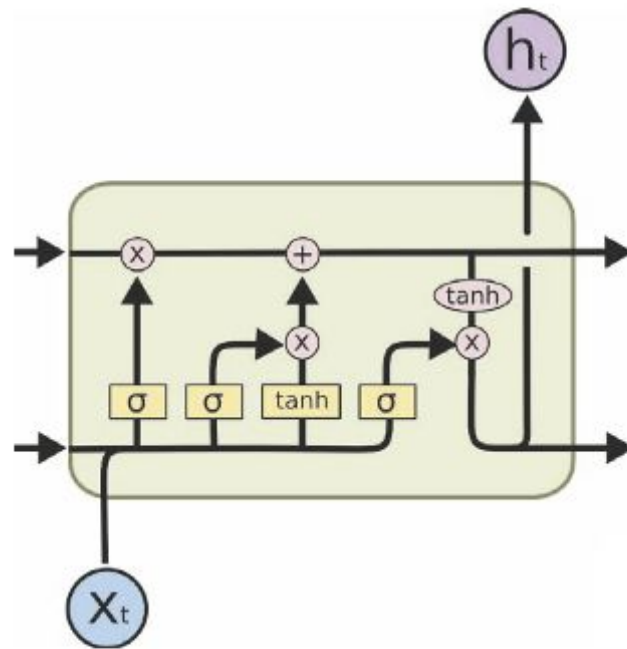
잊을 것은 잊고, 기억할 것만 기억한다.



LSTM (long short term memory)




(a) RNN





(b) LSTM


기억력을 만드는 도구들

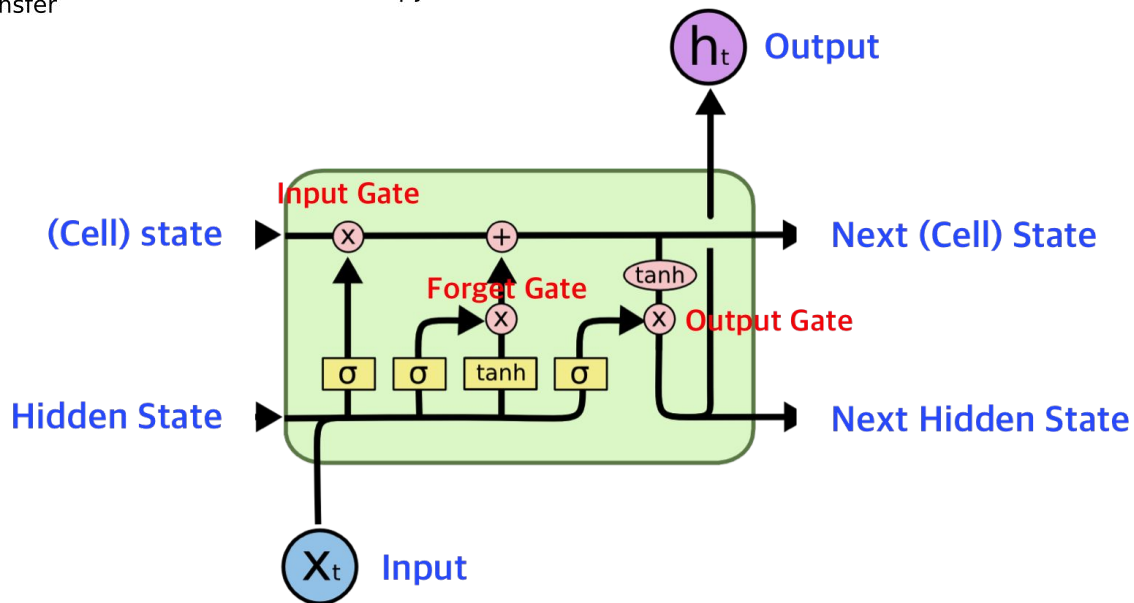

Neural Network
Layer


Pointwise
Operation


Vector
Transfer

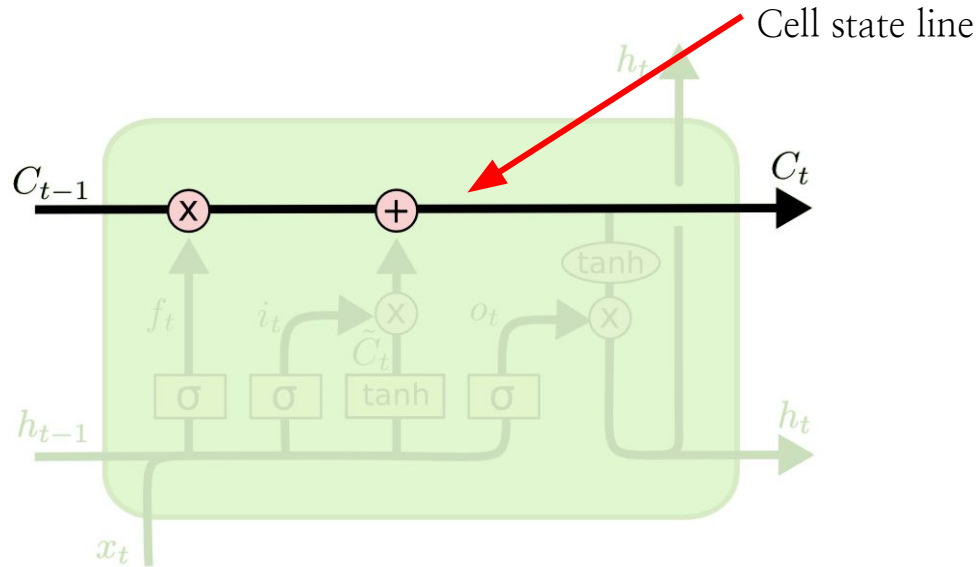

Concatenate


Copy

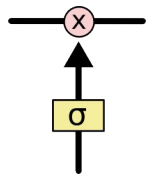


Cell State (기억장치)

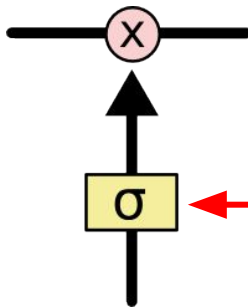
- 기억에 관한 부분을 담당
- \tanh 의 영향을 받지 않는다.



Gate



- 정보를 ‘선택적으로’ 통과시키는 관문
- Sigmoid 활성화함수와 곱셈으로 이루어짐
- cell state를 관리하기 위함
- 3종류의 gate가 있음



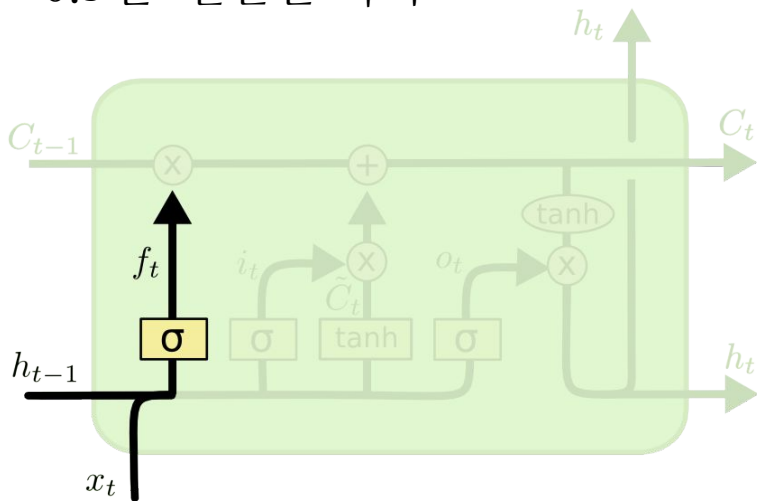
● \times 가 통과시키는 정보의 비율을 정해서 0에서 1 사이의 숫자를 보낸다.

- 0이면 ‘차단’
- 1이면 ‘통과’
- 0.5면 ‘절반만 통과’

Forget Gate

Cell state의 예전 값을 얼마나 ‘잊을지’ 결정해 줌

- 1이면 ‘전부 기억’
- 0이면 ‘전부 망각’
- 0.5면 ‘절반만 기억’

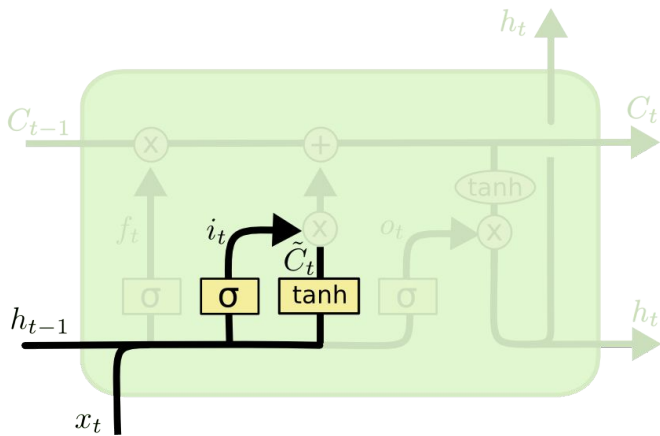


$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input Gate

Cell state의 예전 값을 얼마나 ‘잊을지’ 결정해 줌

- 1이면 ‘전부 기억’
- 0이면 ‘전부 망각’
- 0.5면 ‘절반만 기억’

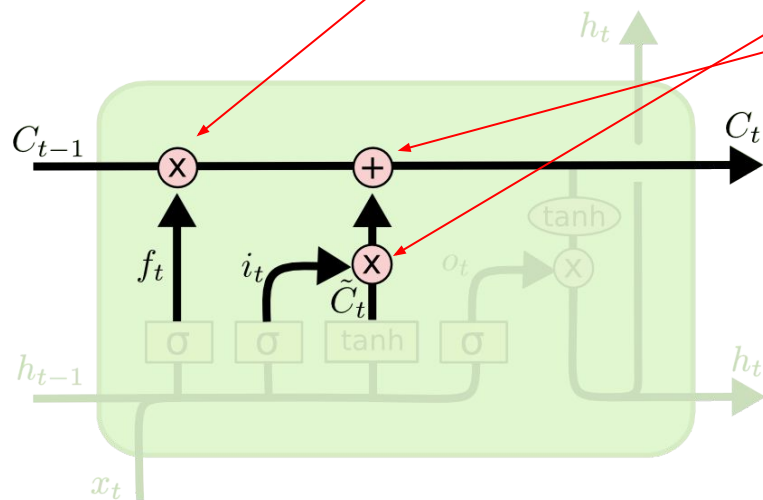
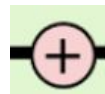


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

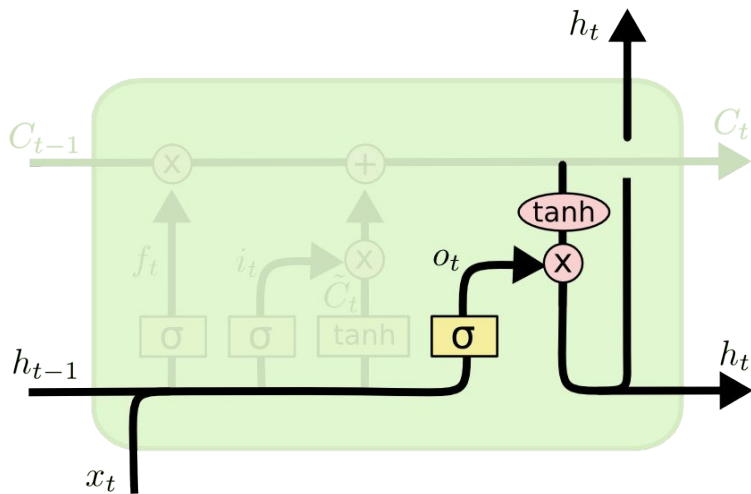
Update Cell State

예전 값을 부분적으로 ‘잊고’, 새 값을 부분적으로 ‘기억하고’, 둘을 합쳐줌



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

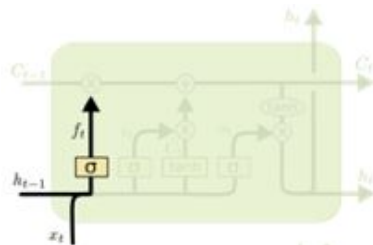
Output Gate - 결과내기



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

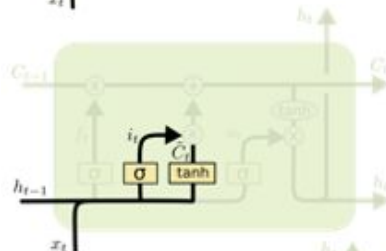
$$h_t = o_t * \tanh (C_t)$$

forget gate



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

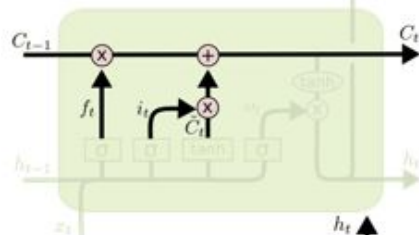
input gate



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

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

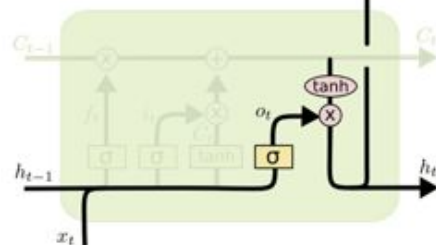
i_t decides what component
is to be updated.
 \tilde{C}_t provides change contents



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Updating the cell state

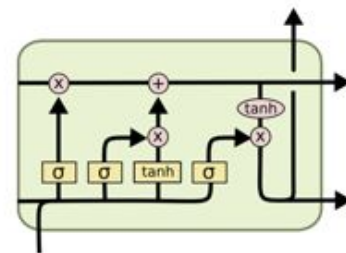
output gate



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

$$h_t = o_t * \tanh(C_t)$$

Decide what part of the cell
state to output



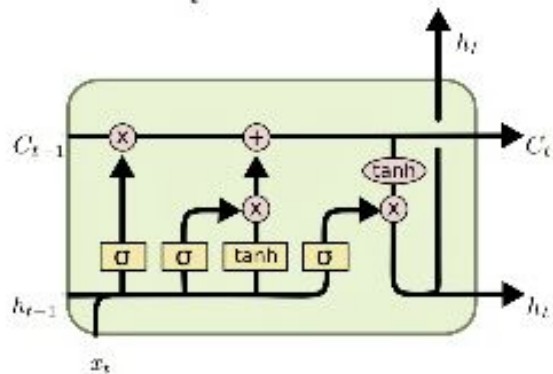
LSTM code

```
1 import tensorflow as tf
2
3 # LSTM
4 model = tf.keras.models.Sequential()
5 model.add(tf.keras.layers.Input(shape = (7, 5)))
6
7 cell = tf.keras.layers.LSTMCell(15)
8 model.add(tf.keras.layers.RNN(cell))
9
10 model.add(tf.keras.layers.Dense(1))
11
12 model.compile(
13     loss=tf.keras.losses.mse,
14     optimizer=tf.keras.optimizers.Adam(lr=0.01)
15 )
```

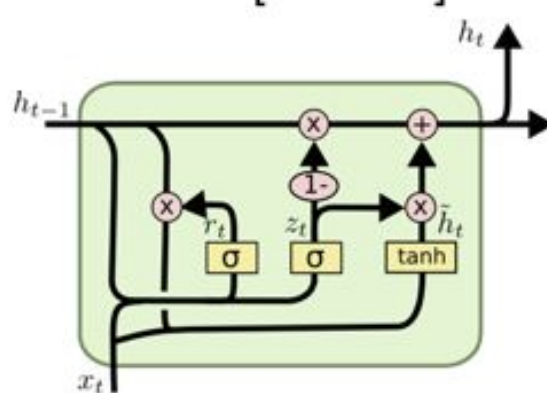
GRU (Gated Recurrent Unit)

- LSTM의 간소화 버전
- hidden state only

- **LSTM** [Hochreiter&Schmidhuber97]

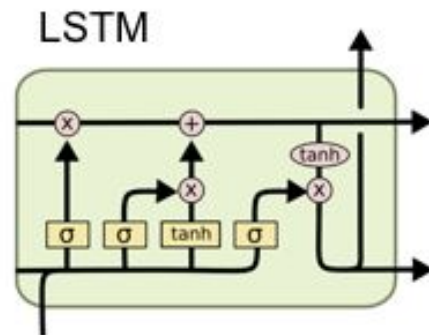
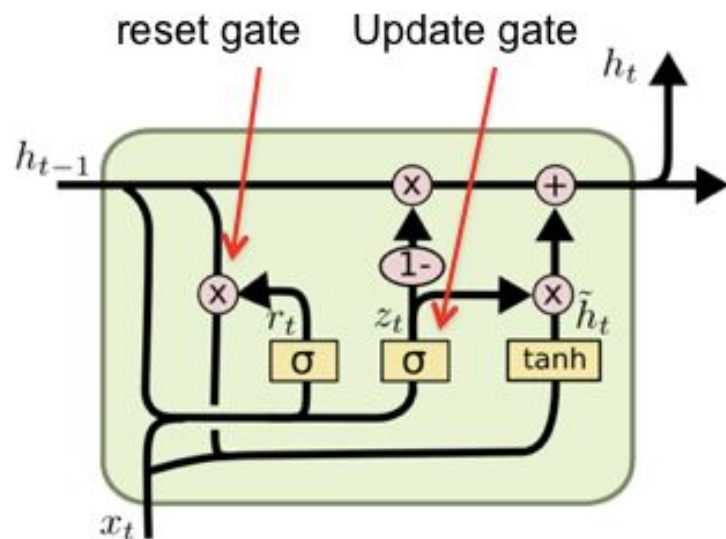


- **GRU** [Cho+14]



GRU – gated recurrent unit

(more compression)



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

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

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

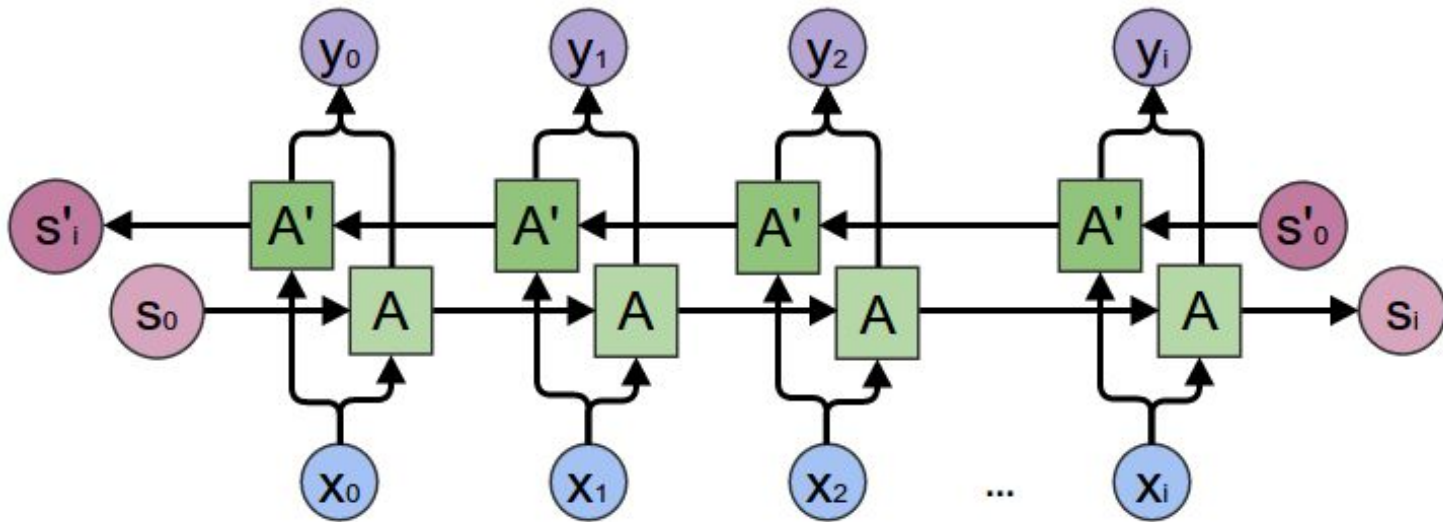
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU code

```
1 import tensorflow as tf
2
3 # GRU
4 model = tf.keras.models.Sequential()
5 model.add(tf.keras.layers.Input(shape = (7, 5)))
6
7 cell = tf.keras.layers.GRUCell(15)
8 model.add(tf.keras.layers.RNN(cell))
9
10 model.add(tf.keras.layers.Dense(1))
11
12 model.compile(
13     loss=tf.keras.losses.mse,
14     optimizer=tf.keras.optimizers.Adam(lr=0.01)
15 )
```

bidirectional

- 이전의 정보뿐 아니라 이후의 정보도 활용해보자.
- 전방향 후방향 각 1개의 cell을 가진다.



bidirectional code

```
1 import tensorflow as tf
2
3 # lstm + bidirectional
4 model = tf.keras.models.Sequential()
5 model.add(tf.keras.layers.Input(shape = (7, 5)))
6
7 cell = tf.keras.layers.LSTMCell(15)
8 lstm = tf.keras.layers.RNN(cell)
9 model.add(tf.keras.layers.Bidirectional(lstm))
10
11 model.add(tf.keras.layers.Dense(1))
12
13 model.compile(
14     loss=tf.keras.losses.mse,
15     optimizer=tf.keras.optimizers.Adam(lr=0.01)
16 )
```

Embedding

- ‘차원의 저주’ 해법
 - 원핫 인코딩으로 만든 벡터는 고차원의 희소벡터
- 밀집 벡터의 형태로 변환
- Embedding 두가지 방법
 - 랜덤한 벡터로 시작해서 신경망의 가중치를 학습하는 것과 같은 방식으로 벡터를 학습합니다.
 - 미리 계산(훈련)된 임베딩을 통해 벡터화합니다.

Embedding code

```
1 import tensorflow as tf
2
3 model = tf.keras.models.Sequential()
4 model.add(tf.keras.layers.Embedding(input_dim=256, output_dim=128))
5 model.add(tf.keras.layers.GlobalAveragePooling1D())
6 model.add(tf.keras.layers.Dense(128))
7 model.add(tf.keras.layers.Activation('relu'))
8 model.add(tf.keras.layers.Dense(10))
9 model.add(tf.keras.layers.Activation('softmax'))
10
11 # 모델 컴파일
12 model.compile(
13     loss=keras.losses.categorical_crossentropy,
14     optimizer=keras.optimizers.Adam(lr=0.01),
15     metrics=[keras.metrics.categorical_accuracy]
16 )
```

NLP 용어 미리보기

- Document : 문서
- Corpus : 말뭉치
- Token : 토큰 : text내의 의미있는 단위
- Morphemes : 형태소 : 언어에서 의미있는 최소 단위
- POS (Part of Speech) : 품사
- Lemmatization : 표제어 추출
- Stemming : 어간추출
- Encoding : 인코딩
- Stopword : 불용어 : 분석에 사용하지 않으려는 용어