6주차 - 책 발표

트리케라톱스

INDEX

-

1. Embedding

텍스트 데이터 다루기 ⇒ 수치형 데이터로 변환 = 텍스트 벡터화

텍스트 ⇒ 단어, 문자, ngram으로 나누기 = 텍스트 토큰화

토큰 – 수치형 벡터

1. 원-핫 인코딩
2. 임베딩

1. Embedding

원-핫 인코딩

단어 별로 인덱스 부여, 이 인덱스를 1 또는 0 값을 가지는 벡터로 변환

Document1 : We are SAI　　Document2 : We are a student

	We	are	SAI	a	student
Document1	1	1	1	0	0
Document2	1	1	0	1	1

간단하고 모든 벡터가 독립적이라 헷갈리지 않음

But, 단어 수가 늘어나는 경우 차원이 커지고
단어 사이의 관계를 알 수 없음

1. Embedding

임베딩

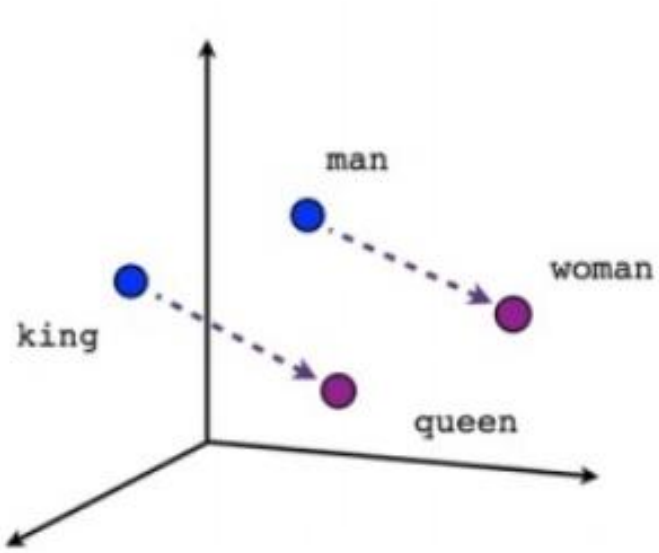
단어를 실수 벡터로 변환함

Document : He is a nice king

	male	Female	Positive	authoritative
He	0.9	0.1	0.2	0.4
is	0.3	0.3	0.2	0.2
a	0.1	0.1	0.1	0.2
nice	0.3	0.3	0.8	0.4
king	0.8	0.2	0.5	0.7

저차원의 실수형 벡터, 단어 간 관계 알 수 있음

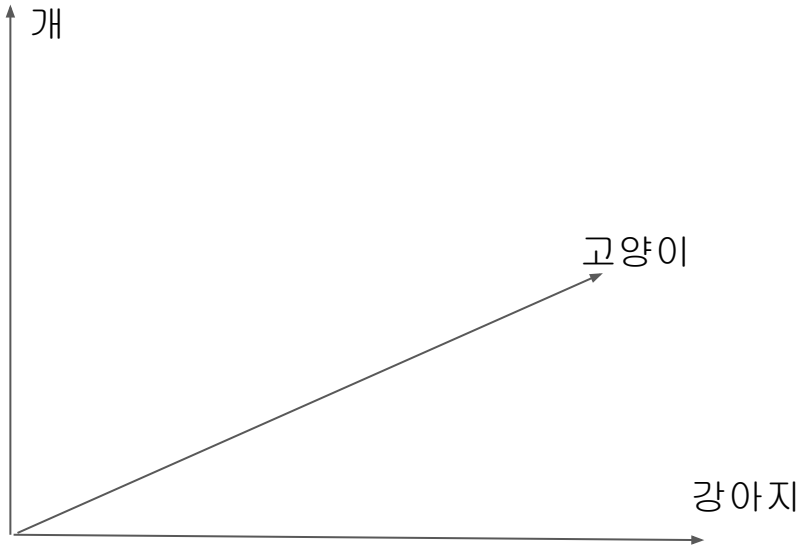
King – male + female = queen

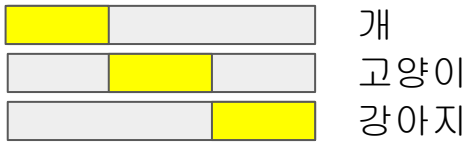


Male-Female

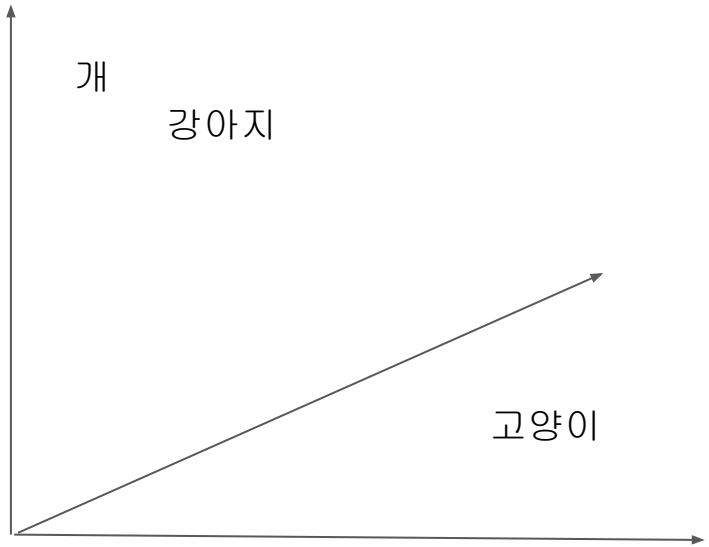
king 벡터에 female 벡터를 더하면 queen
king 벡터에 plural 벡터 더하면 kings

one hot encodeing



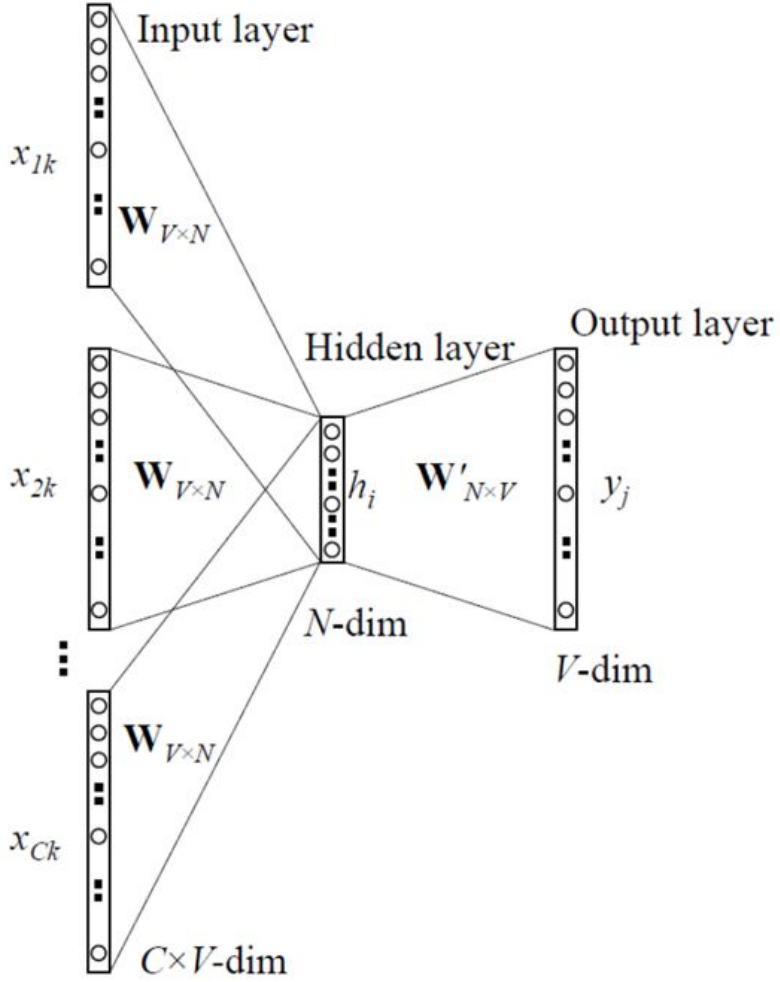


embedding



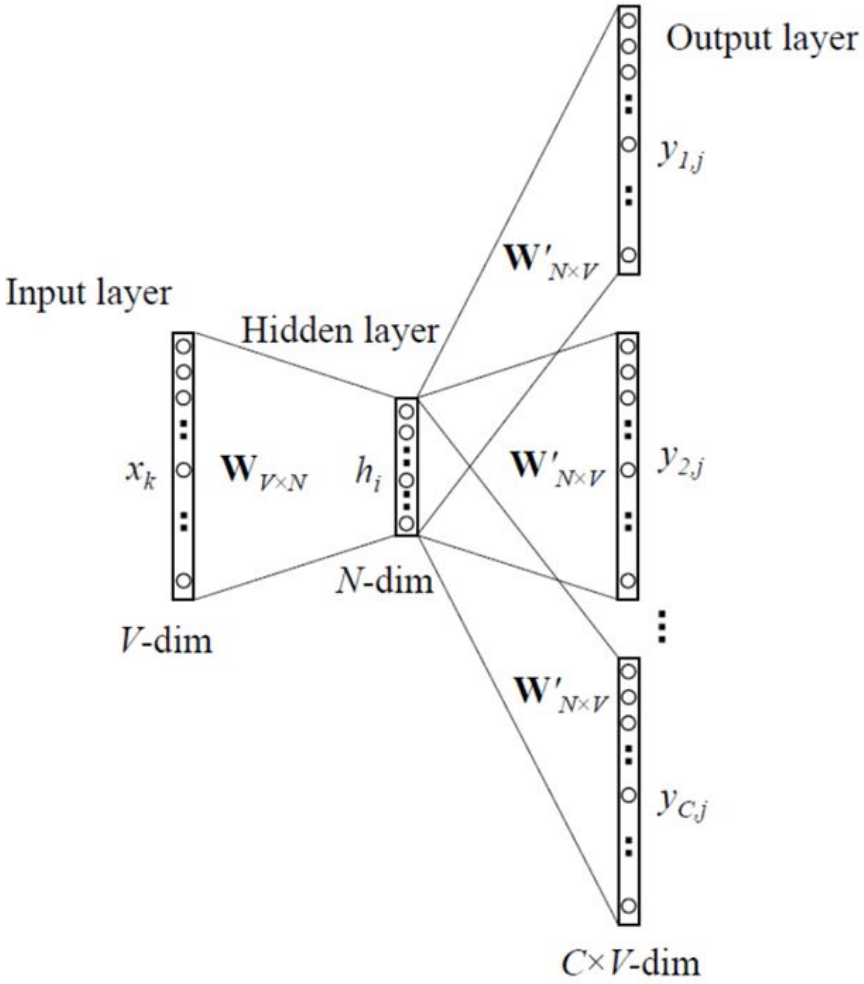
1. Embedding

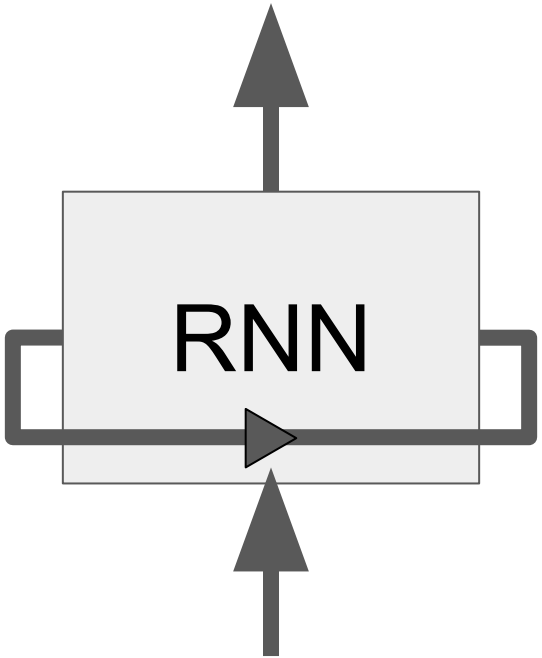
1. CBOW 방식

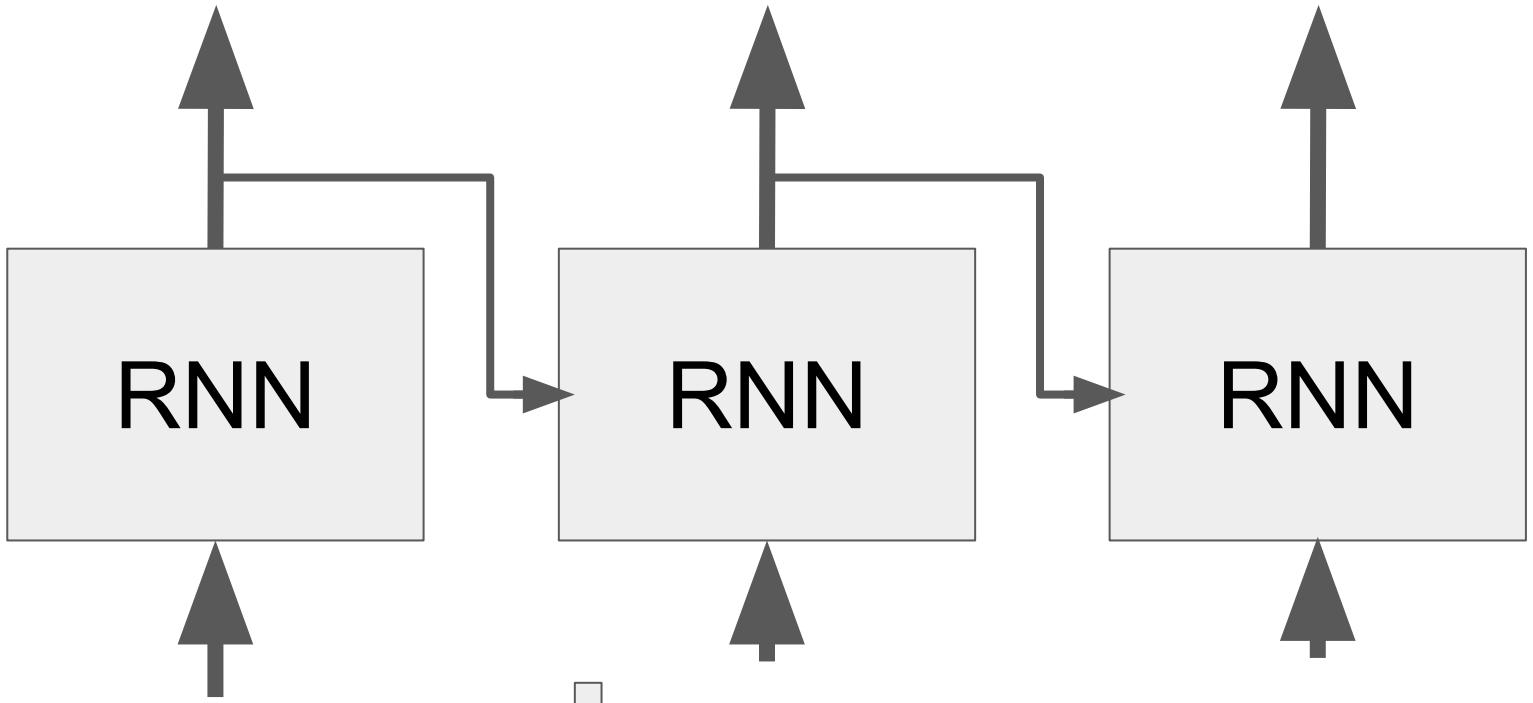


1. Embedding

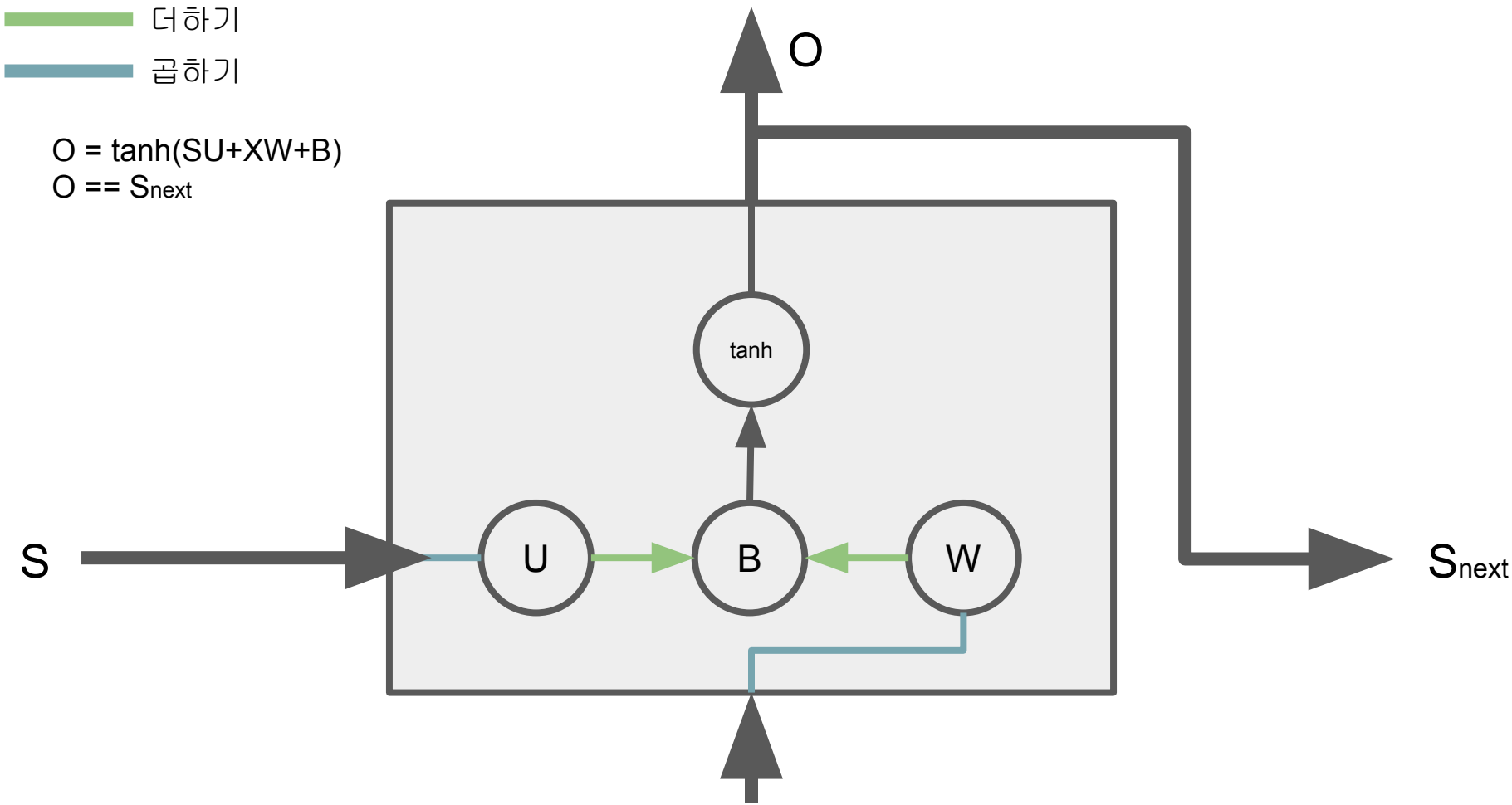
2. Skip-Gram 방식

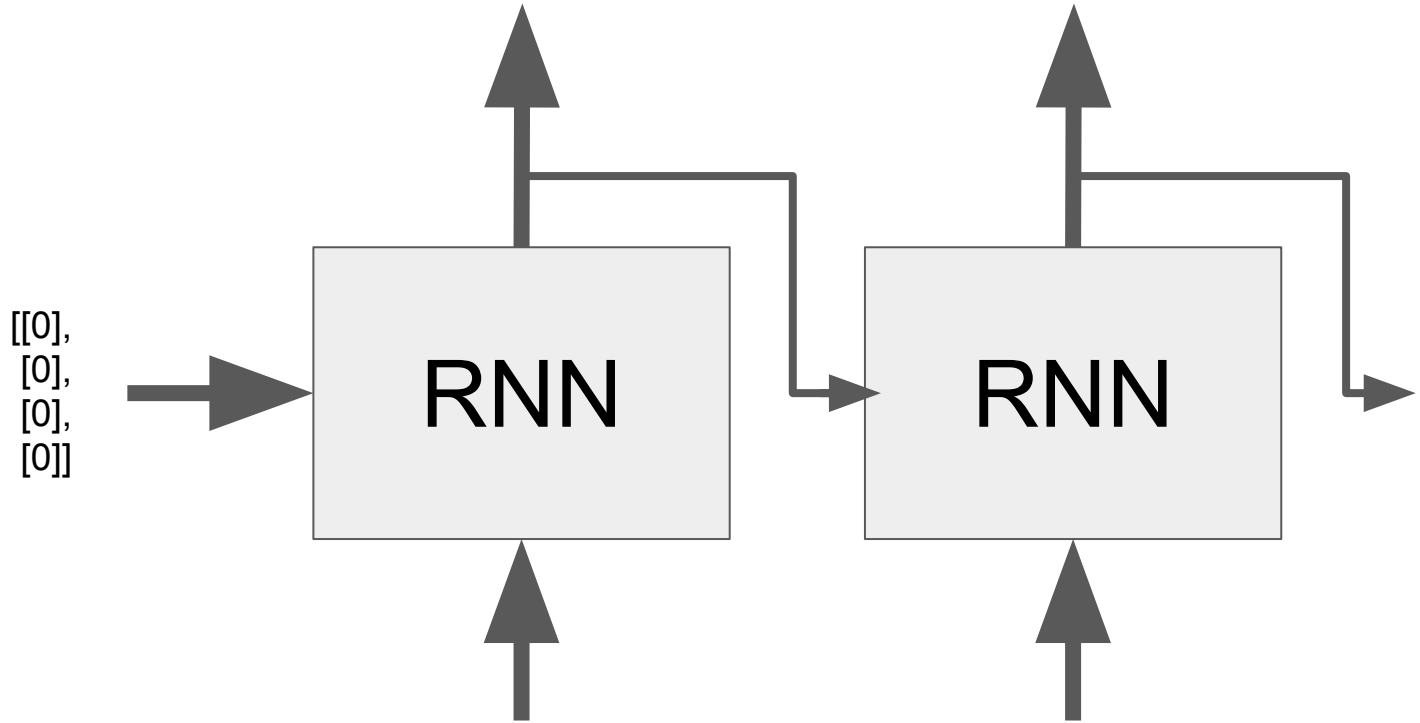


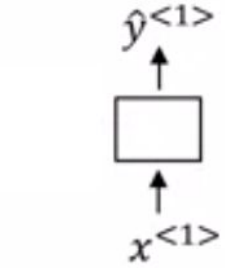

RNN



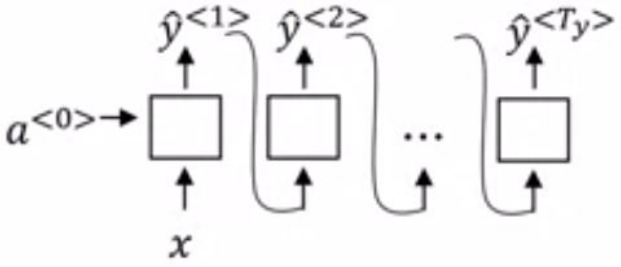
위 모든 RNN 상자는 파라미터를 공유합니다.
= 같은 상자라는 뜻입니다.


더하기
곱하기
O = tanh(SU+XW+B)
O == Snext
O
tanh
S
U
B
W
Snext

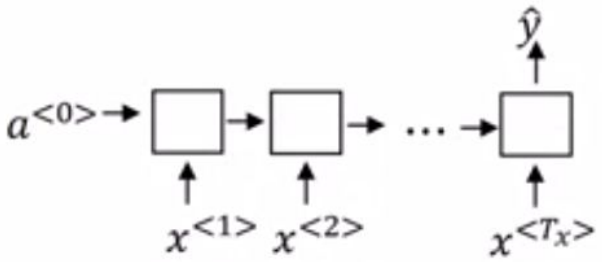

[[0],
[0],
[0],
[0]]
RNN
RNN



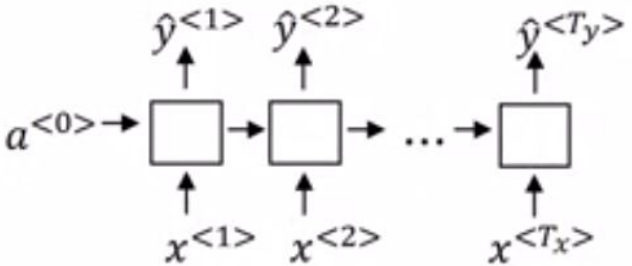
One to one



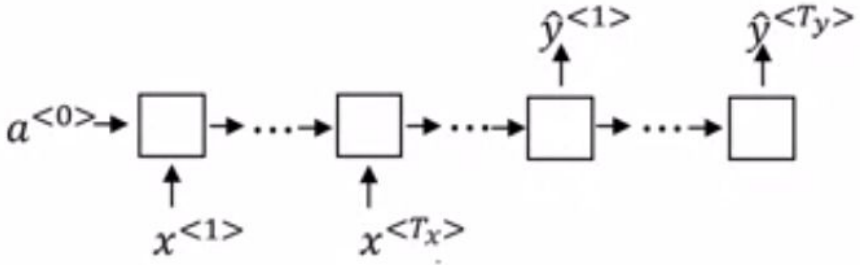
One to many



Many to one



Many to many



Many to many

input word vector

I	love	you	her
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

output word vector

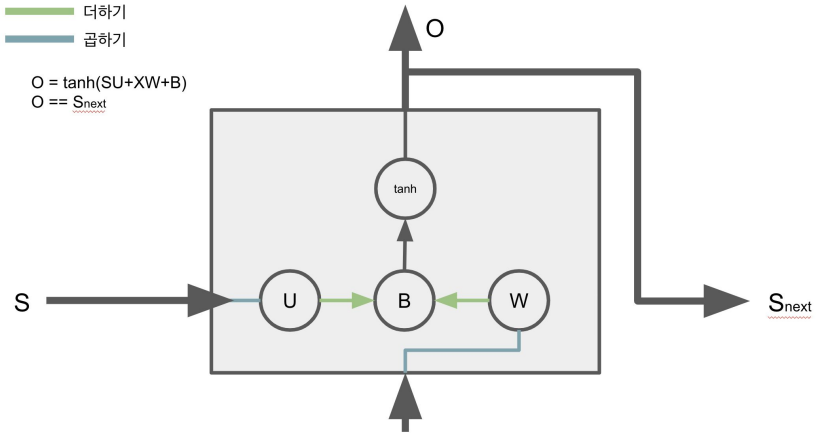
난	널	사랑해
1	0	0
0	1	0
0	0	1

I love you 를 넣어서 난 널 사랑해가 나오게 해보자!

	weight(W)			
	0.5344613383	0.2340135574	0.6833450257	0.134806233
	0.005400196945	0.3850176626	0.9667685167	0.7758002355
output feature	0.3359488417	0.7829360944	0.1415861303	0.423786741
	input feature			

	state weight(U)		
	0.1524336548	0.8001560108	0.159796803
	0.8479080448	0.6059535281	0.2086256969
output feature	0.2940066967	0.710345277	0.6300805231
	output feature		

	bias(B)
	0.6397506561
	0.948082303
output feature	0.644066256



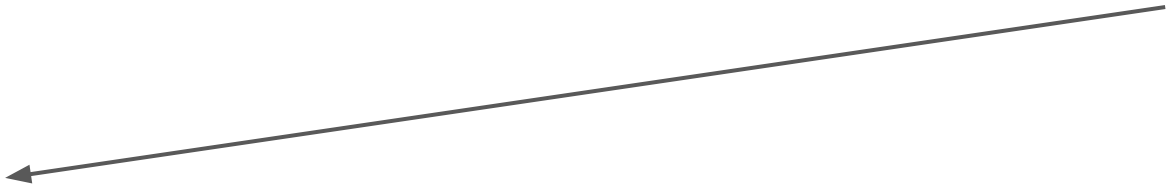
	input vector(X)		
	I	love	you
	1	0	0
	0	1	0
	0	0	1
input feature	0	0	0

첫번째 루프

state(S)	W * X(I)	U * S	WX + US + B	tanh(WX+US+B) (O)	softmax(O[0])	
0	0.5344613383	0	1.174211994	0.8256177525	0.3509740057	난
0	0.005400196945	0	0.9534825	0.7413555946	0.3226118845	널
0	0.3359488417	0	0.9800150977	0.7530724405	0.3264141098	사랑해

첫번째 루프

state(S)	W * X(I)	U * S	WX + US + B	tanh(WX+US+B) (O)	softmax(O[0])
0	0.5344613383	0	1.174211994	0.8256177525	0.3509740057 난
0	0.005400196945	0	0.9534825	0.7413555946	0.3226118845 널
0	0.3359488417	0	0.9800150977	0.7530724405	0.3264141098 사랑해



두번째 루프

state(S)	W * X(I)	U * S	WX + US + B	tanh(WX+US+B) (O)	softmax(O[0])
0.8256177525	0.5344613383	0.8393906351	2.013602629	0.9649761201	0.3305215335 난
0.7413555946	0.005400196945	1.306385235	2.259867735	0.9784509017	0.3350053806 널
0.7530724405	0.3359488417	1.243851871	2.223866968	0.9768607239	0.3344730858 사랑해

첫번째 루프

state(S)	W * X(I)	U * S	WX + US + B	tanh(WX+US+B) (O)	softmax(O[0])	
0	0.5344613383	0	1.174211994	0.8256177525	0.3509740057	난
0	0.005400196945	0	0.9534825	0.7413555946	0.3226118845	널
0	0.3359488417	0	0.9800150977	0.7530724405	0.3264141098	사랑해

두번째 루프

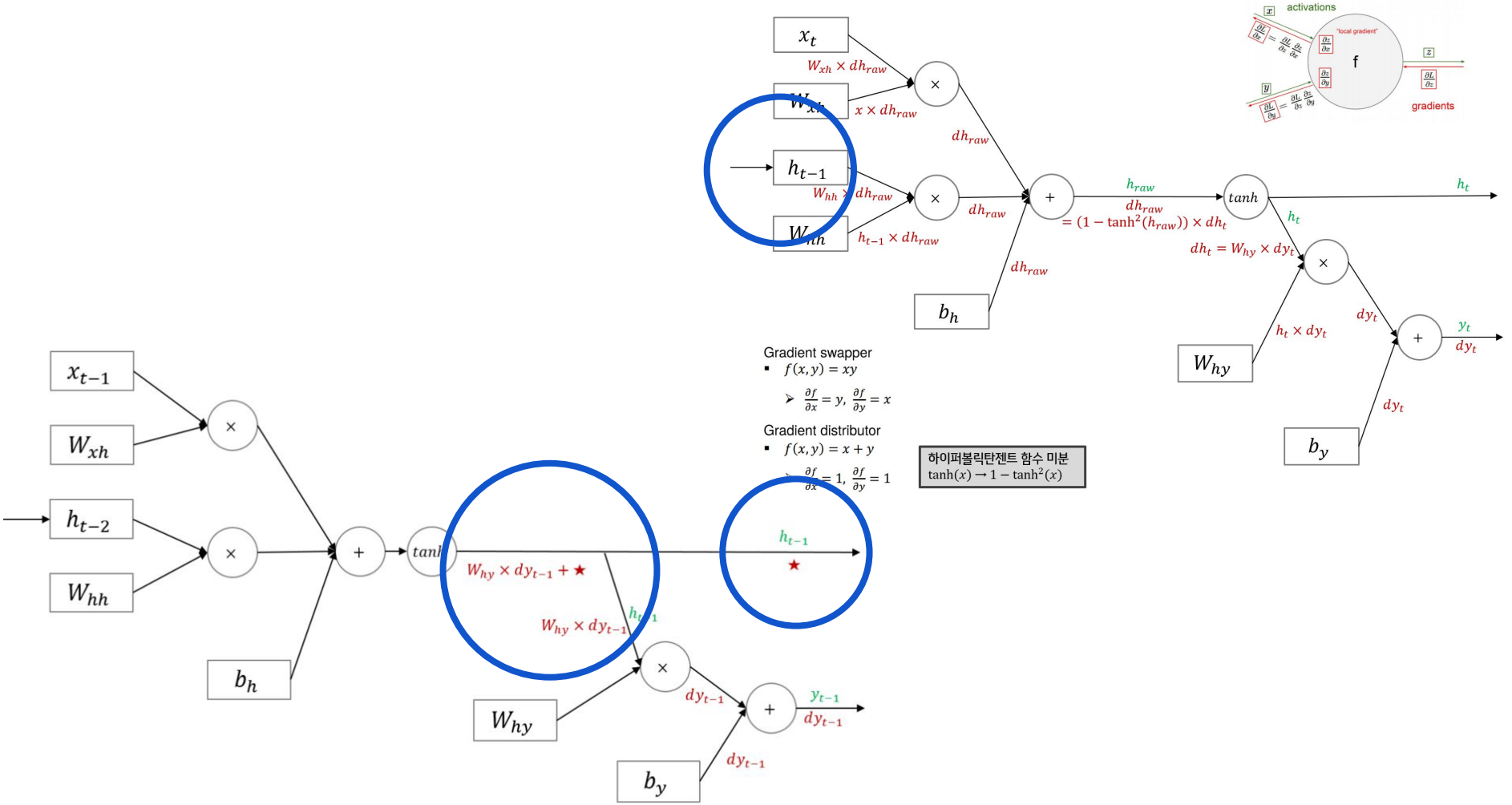
state(S)	W * X(I)	U * S	WX + US + B	tanh(WX+US+B) (O)	softmax(O[0])	
0.8256177525	0.5344613383	0.8393906351	2.013602629	0.9649761201	0.3305215335	난
0.7413555946	0.005400196945	1.306385235	2.259867735	0.9784509017	0.3350053806	널
0.7530724405	0.3359488417	1.243851871	2.223866968	0.9768607239	0.3344730858	사랑해

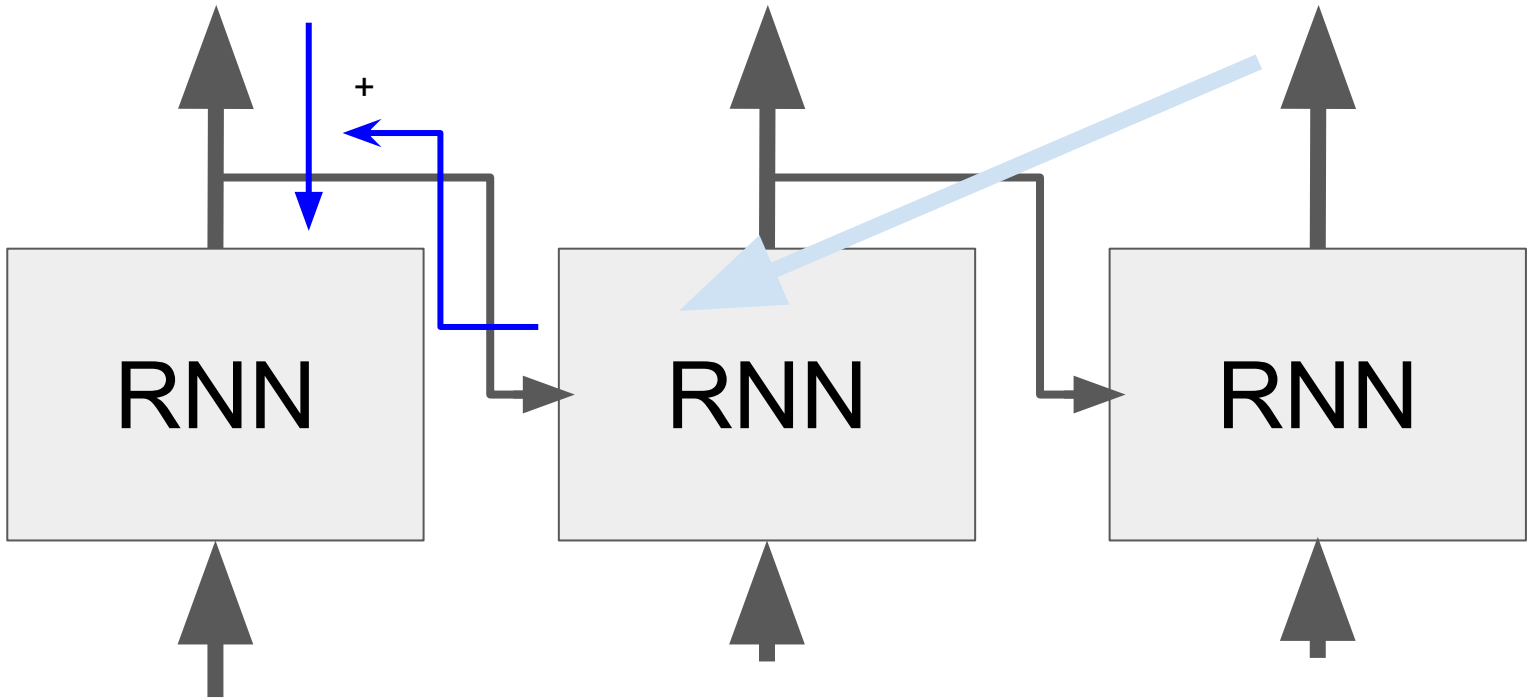
세번째 루프

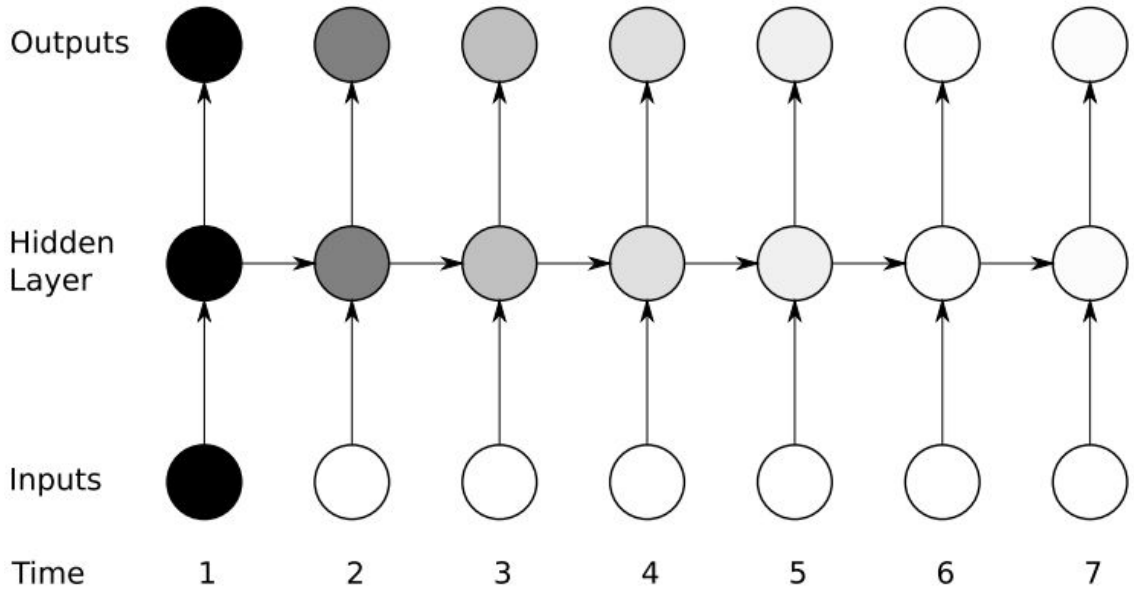
state(S)	W * X(I)	U * S	WX + US + B	tanh(WX+US+B) (O)	softmax(O[0])	
0.9649761201	0.5344613383	1.086107428	2.260319422	0.9784701503	0.3311340611	난
0.9784509017	0.005400196945	1.61490504	2.56838754	0.988315444	0.3344102744	널
0.9768607239	0.3359488417	1.594248334	2.574263432	0.9884511664	0.3344556645	사랑해

결과

입력값	I	love	you
목표결과	난	널	사랑해
실제결과	난	널	사랑해


x_t
$W_{xh} \times dh_{raw}$
$x \times dh_{raw}$
h_{t-1}
$W_{hh} \times dh_{raw}$
$h_{t-1} \times dh_{raw}$
dh_{raw}
dh_{raw}
dh_{raw}
b_h
h_{raw}
dh_{raw}
$= (1 - \tanh^2(h_{raw})) \times dh_t$
tanh
h_t
h_t
$dh_t = W_{hy} \times dy_t$
W_{hy}
$h_t \times dy_t$
dy_t
y_t
dy_t
dy_t
b_y
activations
"local gradient"
f
gradients
Gradient swapper
$f(x, y) = xy$
$\frac{\partial f}{\partial x} = y, \frac{\partial f}{\partial y} = x$
Gradient distributor
$f(x, y) = x + y$
$\frac{\partial f}{\partial x} = 1, \frac{\partial f}{\partial y} = 1$
하이퍼볼릭탄젠트 함수 미분
$\tanh(x) \rightarrow 1 - \tanh^2(x)$
x_{t-1}
W_{xh}
h_{t-2}
W_{hh}
b_h
tanh
$W_{hy} \times dy_{t-1} + \bigstar$
h_{t-1}
$\bigstar$
$W_{hy} \times dy_{t-1}$
h_{t-1}
W_{hy}
dy_{t-1}
y_{t-1}
dy_{t-1}
dy_{t-1}
b_y


+
RNN
RNN
RNN



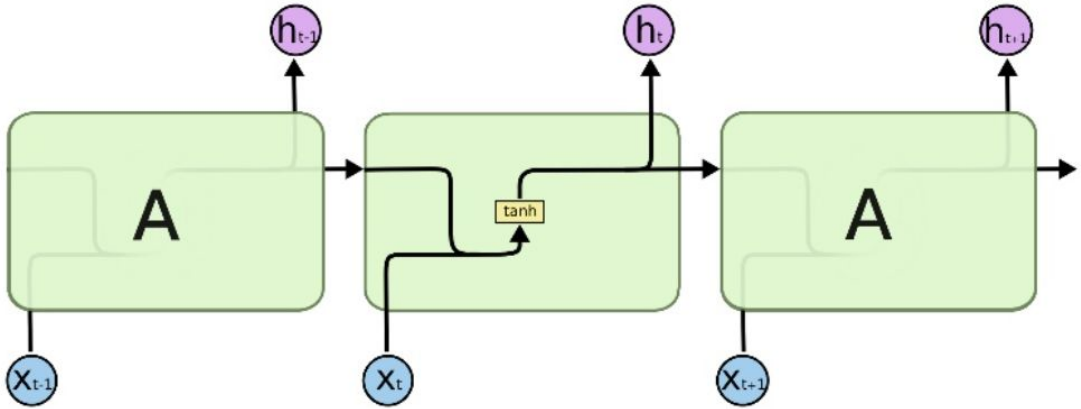
Vanishing Gradient와 Exploding Gradient

끝

https://docs.google.com/spreadsheets/d/1Jai6wNcBnccWn_XJBu6bXoe_4tkcXbwyL3QnuNBiUuM/edit?usp=sharing

LSTM(Long Short-Term Memory)

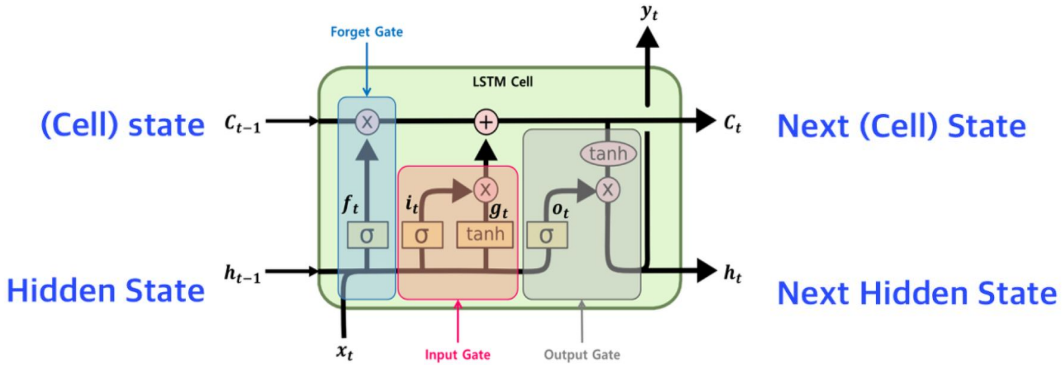
- 기존의 RNN 반복 모듈



단 하나의 layer를 갖고 있는 표준적인 모습

👉 **긴 시간에 걸친 의존성은 학습 불가! (Gradient Vanishing 문제)**

LSTM

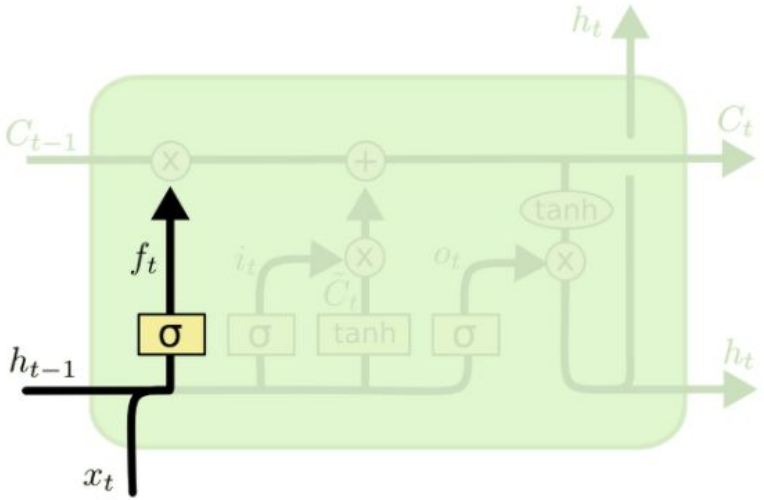


LSTM의 반복 모듈에는 4개의 상호작용하는 layer가 들어있다.

RNN은 수도꼭지가 하나 👉 흐르는 물(기억)의 양은 조절 Ok, 물의 온도는 조절 Fail
LSTM은 밸브가 두개 👉 물의 양, 물의 온도도 조절 가능!

forget gate f_t 는 '과거 정보를 잊기'를 위한 게이트

$$f_t = \sigma(W_{xh\_f}x_t + W_{hh\_f}h_{t-1} + b_{h\_f})$$



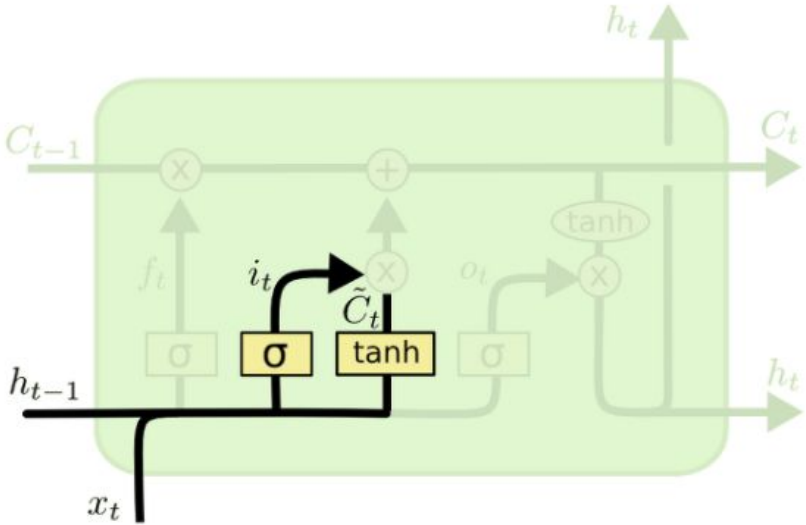
- ht−1 과 xt를 받아 시그모이드를 취해준 값
- 시그모이드 함수의 출력 범위는 0에서 1 사이이기 때문에..

"값이 0이라면 이전 상태는 잊고,
1이라면 이전 상태를 온전히 기억해라!"

ex) 주어가 현영에서 영채로 바뀌었을때 현재 성별 "여성"을 잊어버리는 것

input gate $i_t \odot g_t$ 는 '현재 정보 기억하기'를 위한 게이트

$$i_t = \sigma(W_{xh\_i}x_t + W_{hh\_i}h_{t-1} + b_{h\_i})$$
$$g_t = \tanh(W_{xh\_g}x_t + W_{hh\_g}h_{t-1} + b_{h\_g})$$



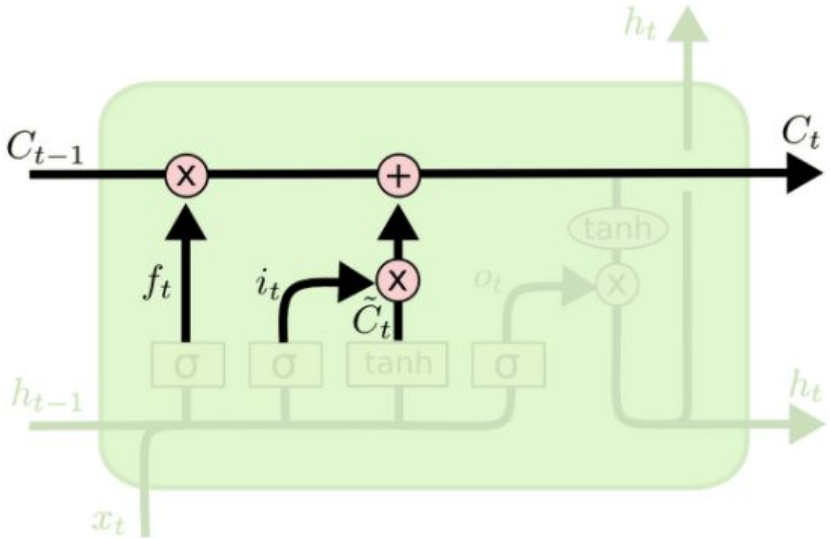
- ht−1 과 xt를 받아 시그모이드 취한 것과 하이퍼볼릭탄젠트를 취해준 것을 요소별 곱셈 연산 한 값이 input gate가 내보내는 값
- 지금 입력과 이전 출력으로 얻어진 값을 얼마나 cell state에 반영할지 정하는 역할

ex) "남성" 성별을 cell state에 더하기 위해 준비

Cell State 업데이트

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

- 이렇게 두 단계에서 나온 정보를 합쳐서 state를 업데이트할 재료를 만들게 된다.

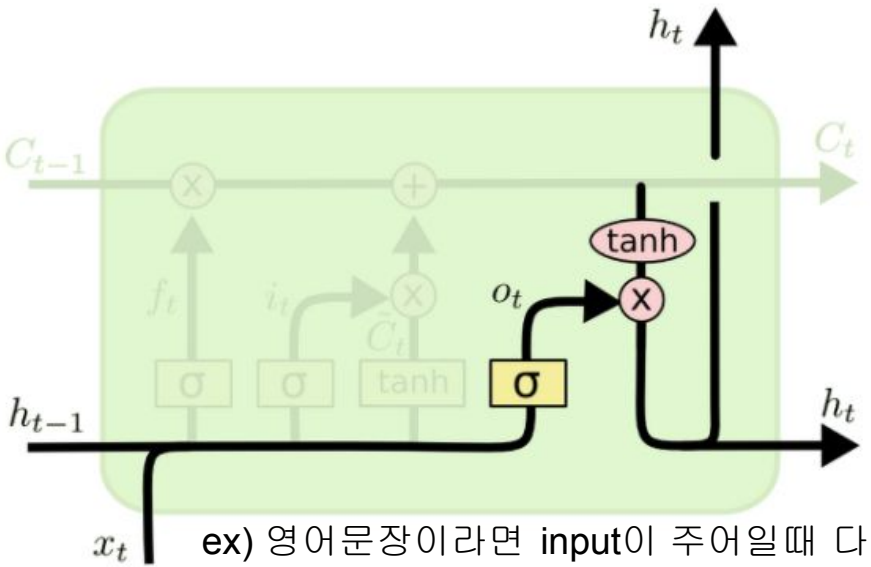


ex) “여성” 성별을 잊고, “남성” 성별로 업데이트 실천

무엇을 output으로 내보낼까?

$$o_t = \sigma(W_{xh\_o}x_t + W_{hh\_o}h_{t-1} + b_{h\_o})$$

$$h_t = o_t \odot \tanh(c_t)$$

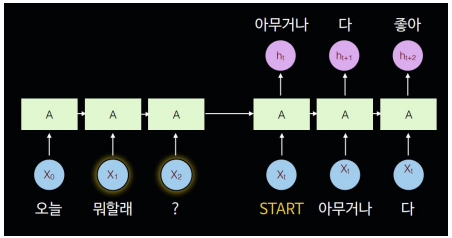


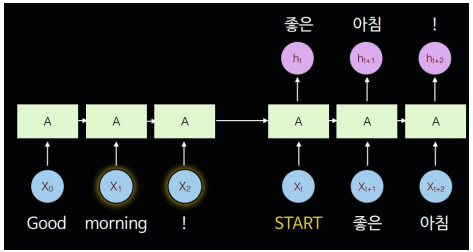
- sigmoid layer에 input 데이터를 태워서 cell state의 어느 부분을 output으로 내보낼지를 정한다
- 그리고나서 cell state를 tanh layer에 태워서 -1과 1 사이의 값을 받은 뒤에 방금 전에 계산한 sigmoid gate의 output과 곱해준다.
- 그렇게 하면 우리가 output으로 보내고자 하는 부분만 내보낼 수 있게 된다!!

ex) 영어문장이라면 input이 주어일때 다음 예측값 ouput은 동사일 것. 앞의 주어가 단수인지 복수인지에 따라 형태 달리하게 됨

LSTM이 잘하는 것

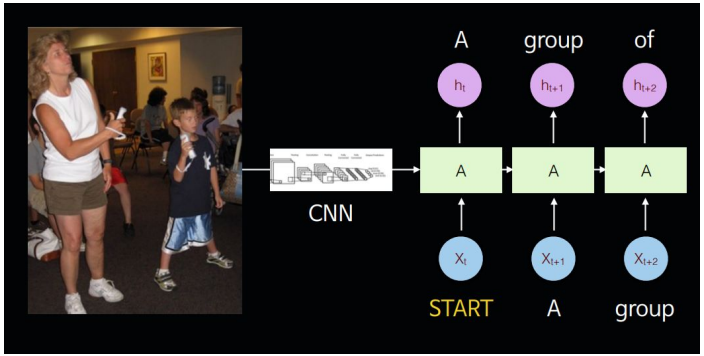
- 중기 기억을 요구하는 패턴 추출 문제
- 질문 - 응답 분야
- 기계 번역 분야





응용

- CNN과 합쳐지면 -> 이미지 설명달기도 가능!

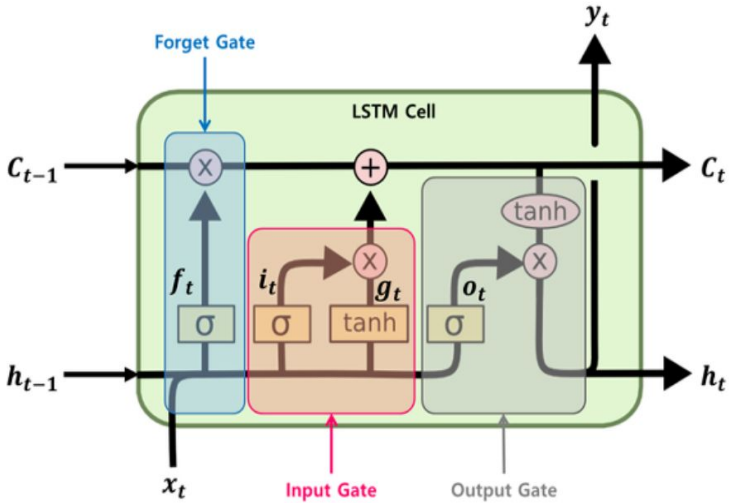


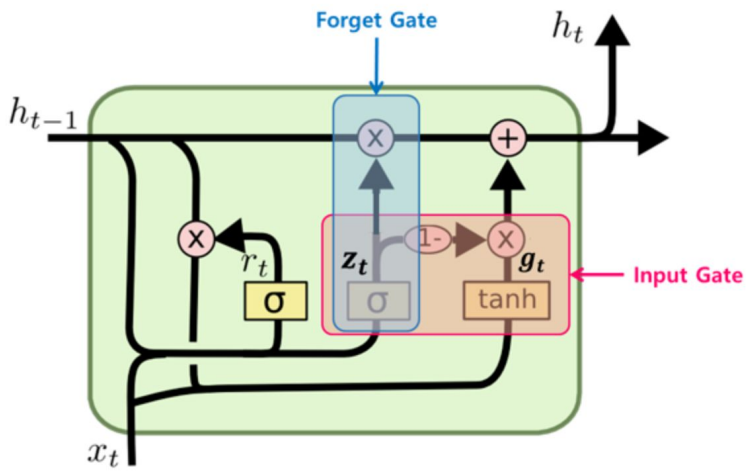
GRU(Gated Recurrent Unit)

LSTM셀의 **간소화된 버전**

- LSTM Cell에서의 두 상태 벡터 $\mathbf{c}_t$와 $\mathbf{h}_t$가 하나의 벡터 $\mathbf{h}_t$로 합쳐짐
- 하나의 gate controller은 $\mathbf{z}_t$ 가 **forget**과 **input** 게이트(gate)를 모두 제어

$$
\begin{aligned}
\mathbf{r}_t &= \sigma\left(\mathbf{W}_{xr}^T \cdot \mathbf{x}_t + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_r\right) \\
\mathbf{z}_t &= \sigma\left(\mathbf{W}_{xz}^T \cdot \mathbf{x}_t + \mathbf{W}_{hz}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_z\right) \\
\mathbf{g}_t &= \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_g\right) \\
\mathbf{h}_t &= \mathbf{z}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \otimes \mathbf{g}_t
\end{aligned}
$$

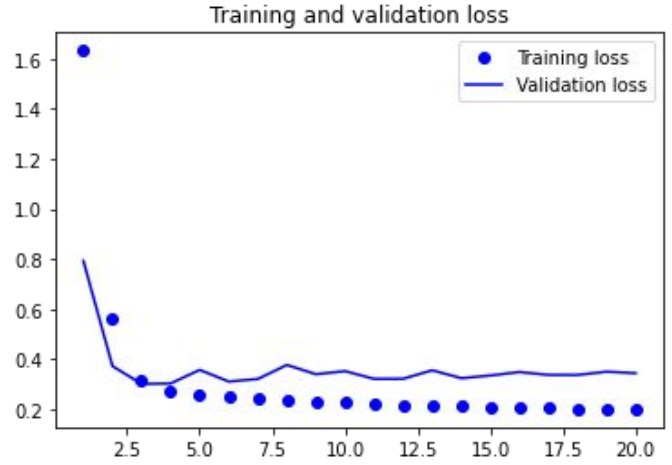


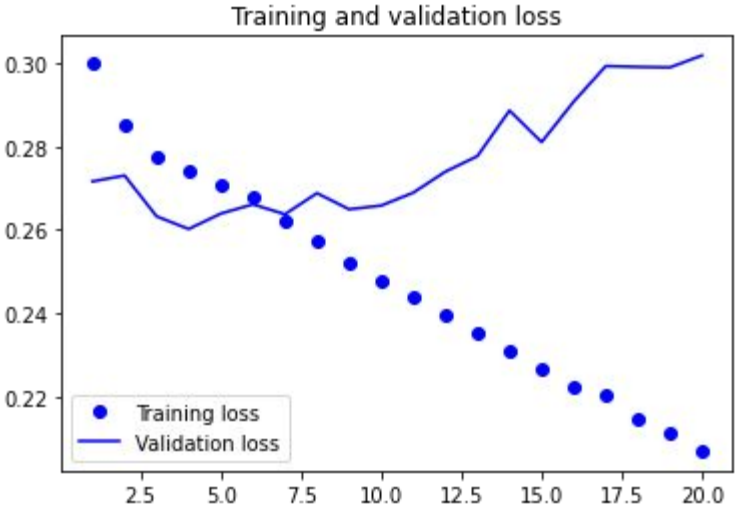


LSTM **GRU**

순환 신경망 고급 사용법

1. 순환 드롭아웃(dropout)
2. 스태킹 순환 층
3. 양방향 순환 층





순환 드롭아웃(dropout)

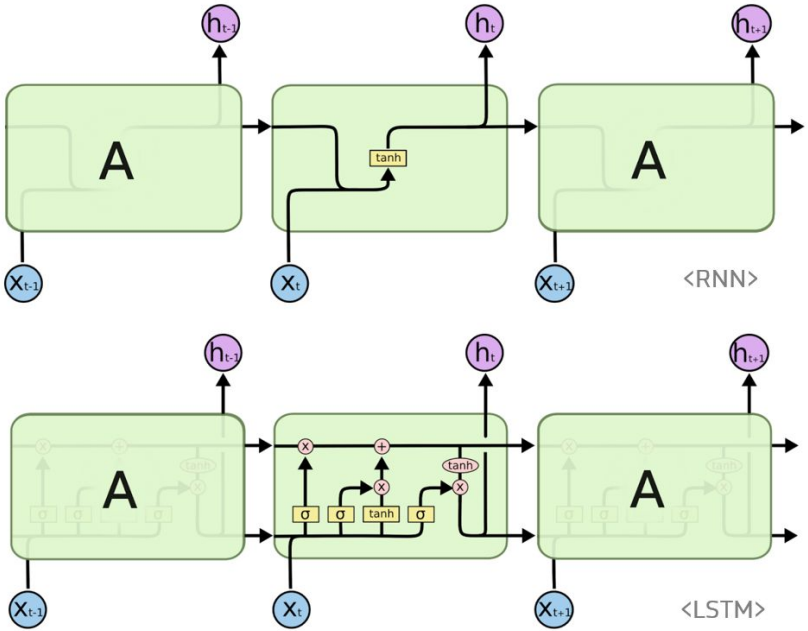
순환 층에서 과대적합을 방지하기 위해 케라스에 내장되어 있는 드롭아웃을 사용

```
6 model.add(layers.GRU(32,
7                       dropout=0.2,
8                       recurrent_dropout=0.2,
9                       input_shape=(None, float_data.shape[-1])))
```

Dropout Layer를 사용하게 되면 중요한 과거 정보를 잃어버릴 확률이 높아지게 되고 따라서 Model의 성능이 나빠진다??

RNN의 구조

순차적 정보의 연관성을 모델링하기 위한 RNN
이전 셀의 정보 Drop은 앞의 셀에 영향을 미친다.



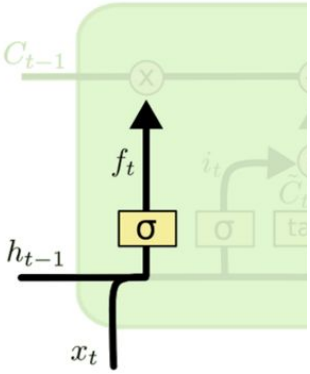
LSTM에서의 Drop

$$\begin{pmatrix} i \\ f \\ o \\ \tanh \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} \mathbf{D}(h_t^{l-1}) \\ h_{t-1}^l \end{pmatrix}$$

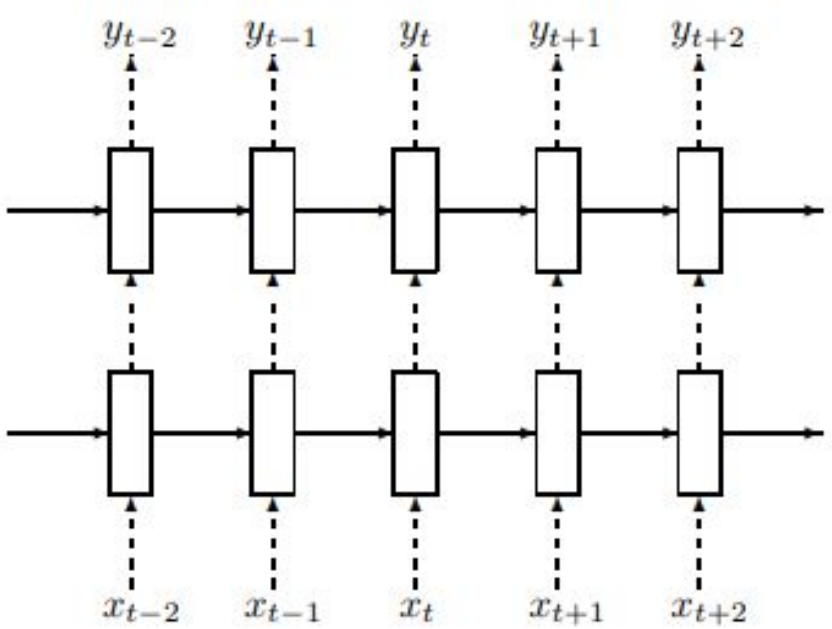
$$_1 + i \odot g$$

$$\mathrm{h}(c_t^l)$$

현재 time ste

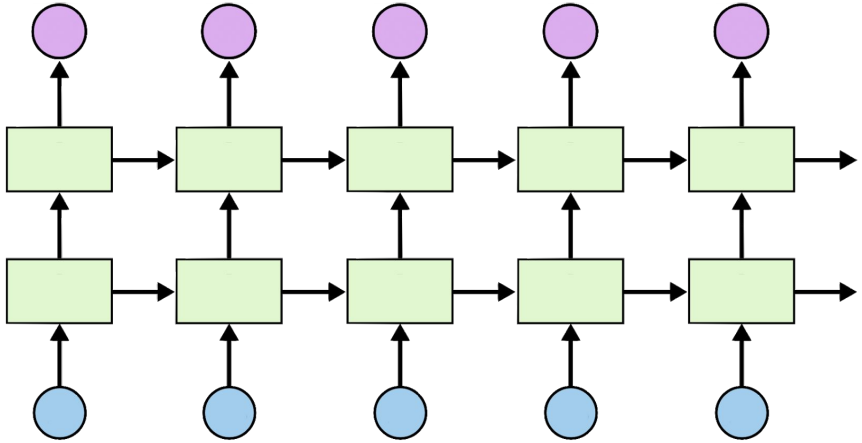


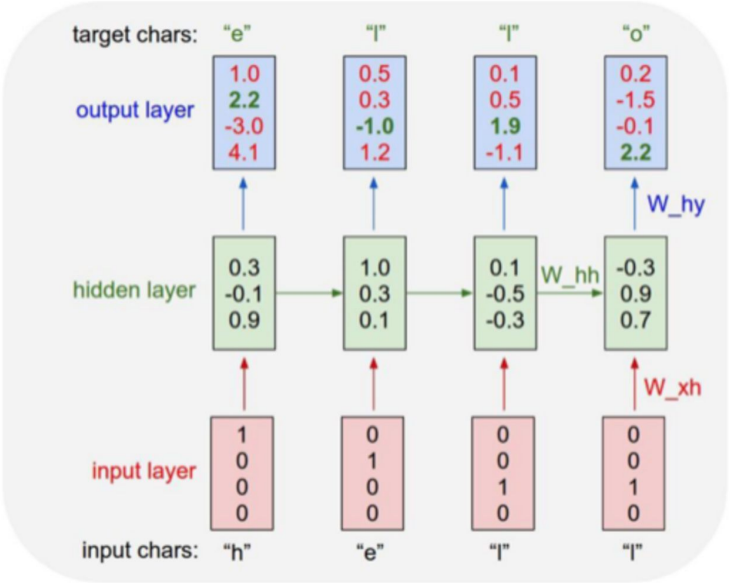
<for



스태킹 순환 층

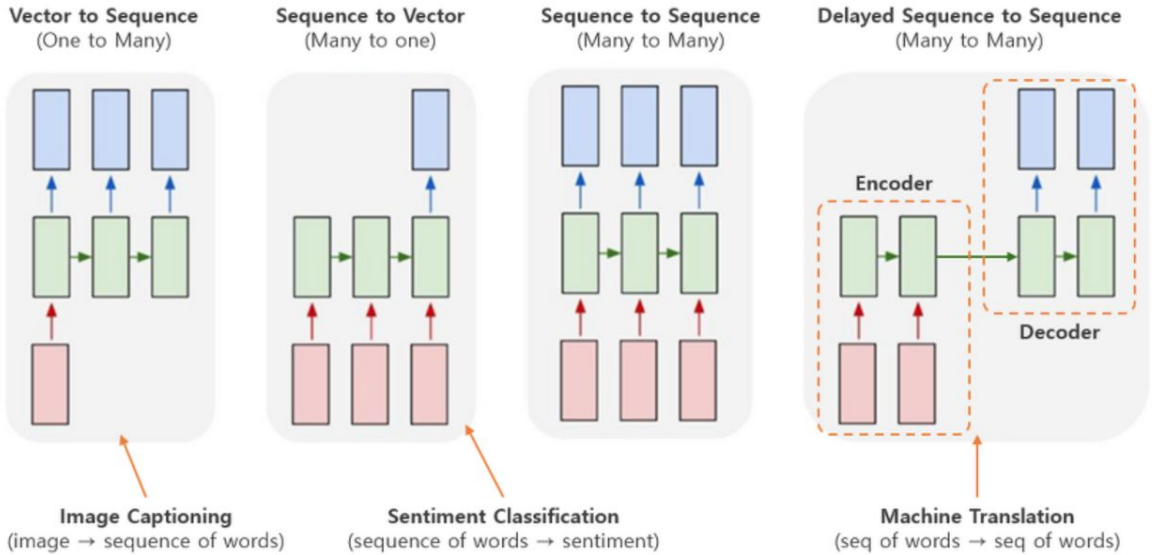
모델 용량을 증가시켜 복잡한 데이터의 표현 능력 제공





RNN 모델의 형태

```
model.add(layers.GRU(32,
                     dropout=0.1,
                     recurrent_dropout=0.5,
                     return_sequences=True,
                     input_shape=(None, float_data.shape[-1])))
model.add(layers.GRU(64, activation='relu',
                     dropout=0.1,
                     recurrent_dropout=0.5))
```

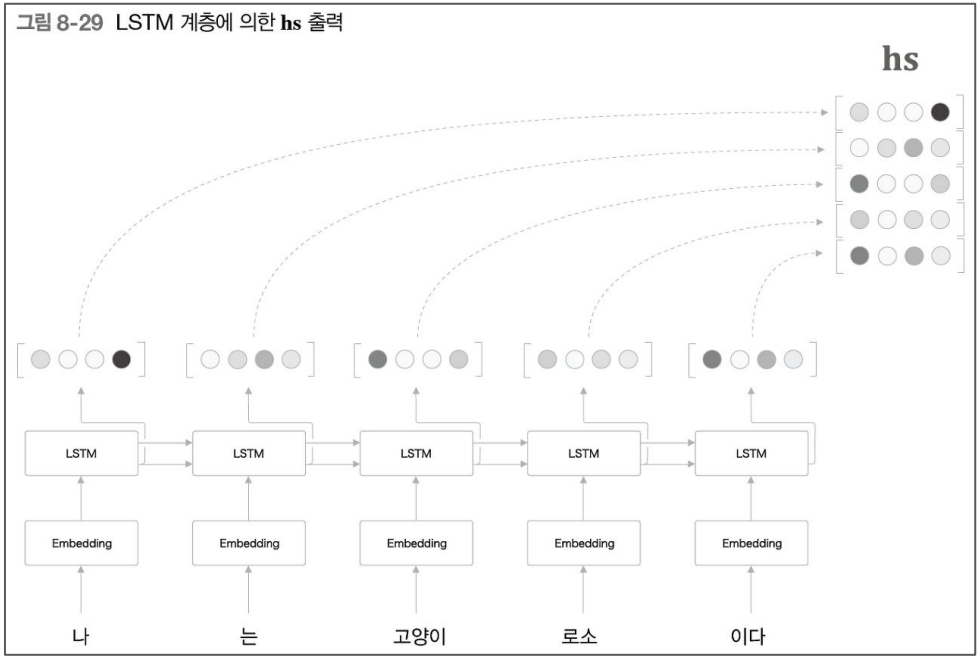


양방향 순환 층

지금의 RNN은
글을 왼쪽에서 오른쪽으로 읽는 것과 같다.

문제점: 오래된 정보의 상실

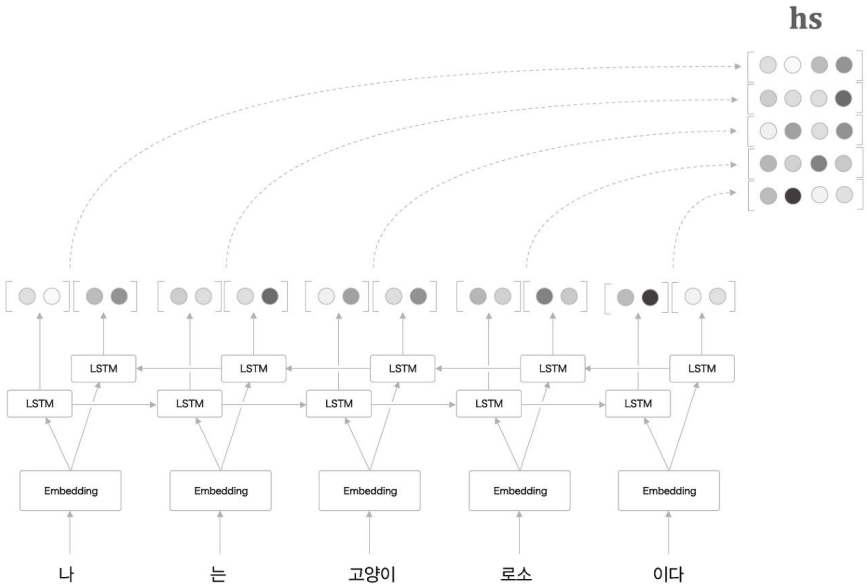
1443년에 집현전 학자들의 도움을 얻어 창제(**創製**)한 **세종**.
세종은 1443년에 집현전 학자들의 도움을 얻어 창제(**創製**)함.



역방향 레이어

언어를 이해하는 데 단어의 순서가 중요하지만 결정적이지 않다는 가정을 뒷받침합니다.

그림 8-30 양방향 LSTM으로 인코딩하는 예(LSTM 계층을 간략화하여 그림)



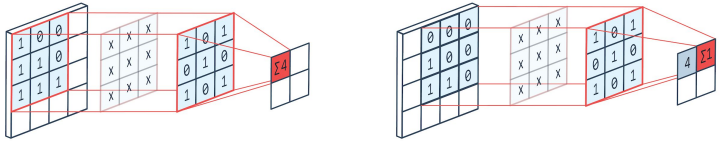
양방향 LSTM 특징

양방향 LSTM은 최근 머신러닝 분야에서 좋은 성과에 적용된 모델일 정도로 높은 성능의 알고리즘 중 하나이다. (Ko et al., 2018)

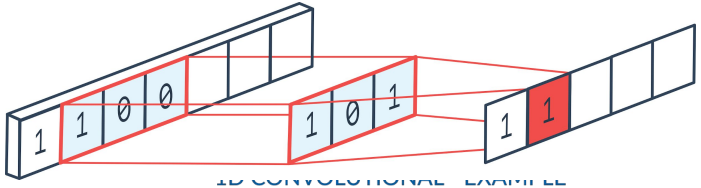
1. 출력값에 대한 손실을 최소화하는 과정에서 모든 파라미터를 동시에 학습되는 종단간 학습 가능
2. 단어와 구(Phrase)간 유사성을 입력벡터에 내재화하여 성능 개선
3. 데이터 길이가 길어도 성능이 저하되지 않음
    - LSTM의 기본 성능과 Attention 매커니즘을 도입함

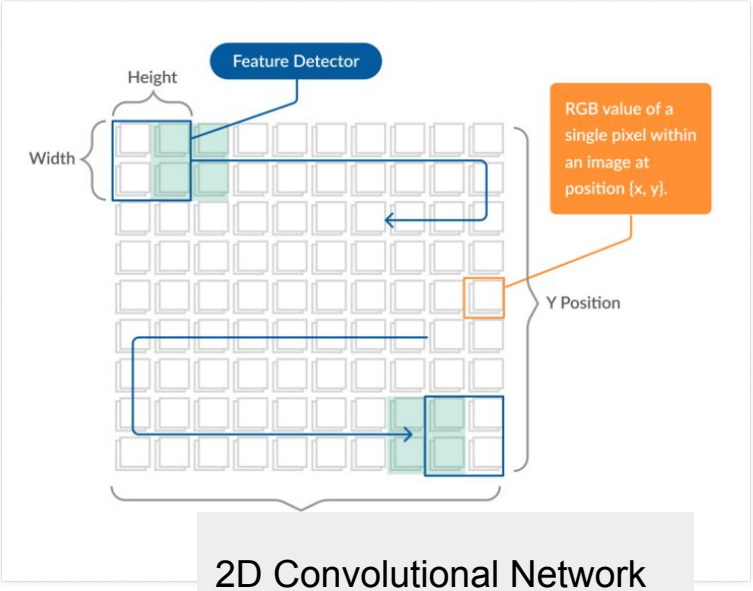
Using ‘ConvNet’ on Sequential data

ConvNet을 이미지 말고도 적용할 수 있다고?

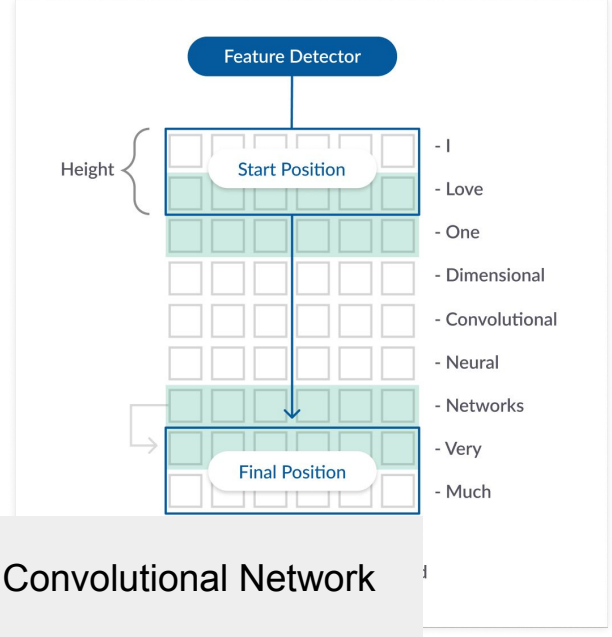


2D CONVOLUTIONAL - EXAMPLE





2D Convolutional Network



1D Convolutional Network

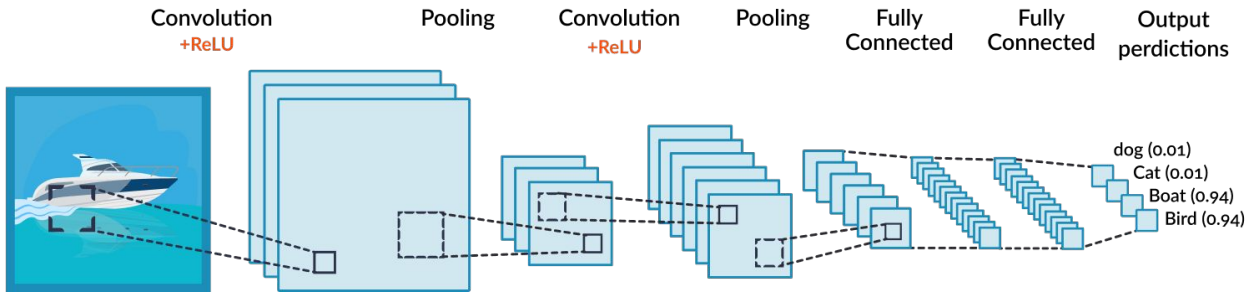
1D ConvNet 모델은 어떤 점이 달라질까?

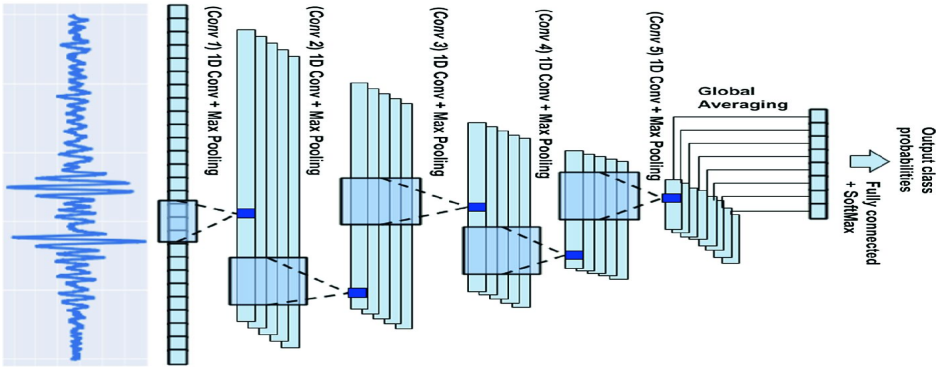
사용하는 layer의 기능과 구성은 동일하다.

특성 추출을 위한 convolution layer

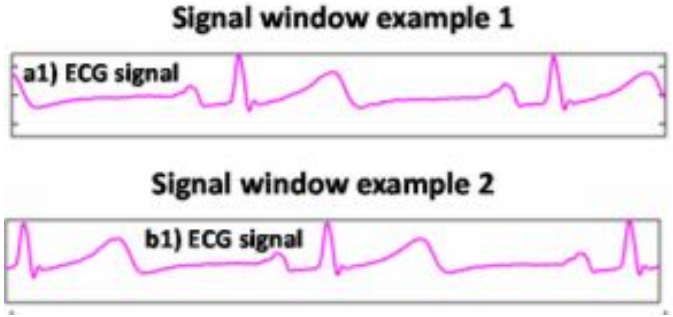
다운샘플링해 길이를 줄이는 1D Pooling

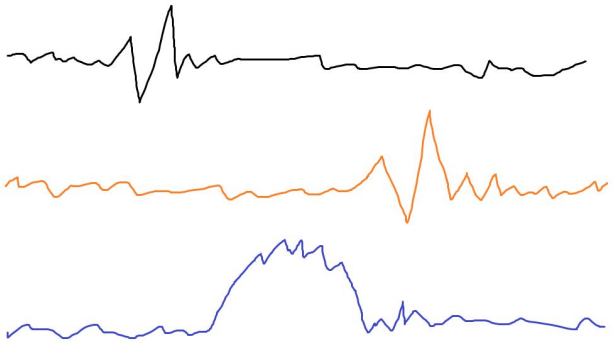
모든 데이터를 취합해주는 Fully Connected 또는 Global Pooling





1D ConvNet의 특성 추출





1D CNN은 shift가 일어난 데이터에 대해 동일하게 인식할 수 있다!

2D Convnet과도 동일한 특성.

(remind: 2D convnet의 이동 불변성)

이걸 '이동 불변성'(translation invariant) 이라고 부른다.

이동불변성(translation invariant)의 양면

순서에 영향을 거의 받지 않고, 시간적 위치를 고려하지 않는 특성 (특히나 패치 사이즈가 전체 시퀀스보다 훨씬 작을 때)

좋은 점: 굉장히 긴 시퀀스에서 RNN보다 적은 연산량으로 특성 추출 가능
나쁜 점: 최근 데이터가 예전 데이터보다 더 중요한 데이터셋-기온 예측, 주가 예측, 에너지 소비량 예측 등-에 대해선 학습하기 힘들다는 점

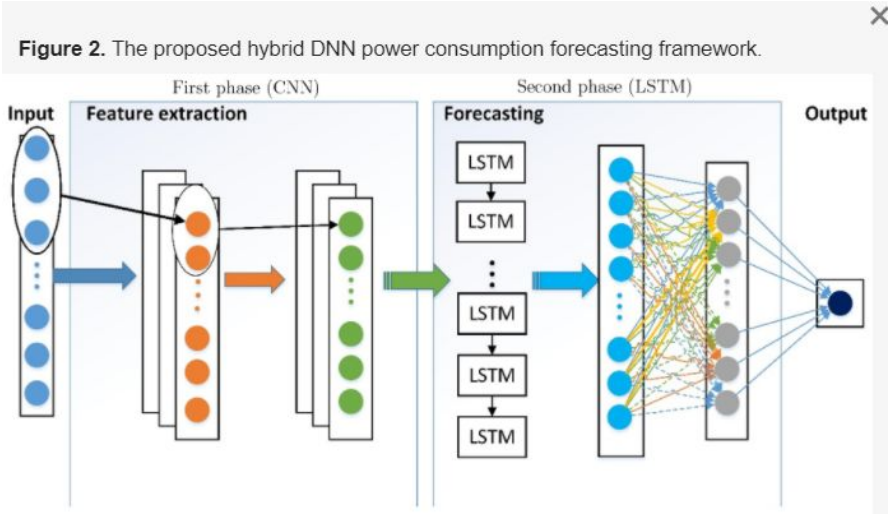
RNN과의 비교

처리속도는 훨씬 빠른데, 정확도에 한계가 있는 편이다.

RNN의 전처리과정으로 사용하면 둘의 장점 모두 혼합 가능하다는 저자의 제안이 있었음.

검색을 좀 해보니 'CNN-RNN hybrid', 'CNN-LSTM hybrid' 같은 제목의 논문으로 실제로 이런 구조를 사용하는 것을 확인할 수 있었다.

그런데 이런 키워드로 검색하면 동영상 데이터를 다루기 위해 이미지를 1차로 CNN으로 처리를 하고, RNN으로 순서를 처리하는 그런 형태가 대다수였다.



ConvNet 요약

어떤 데이터셋에 효과적인가?

- 순서가 그다지 상관이 없고, 패턴 단위로 인식하면 되는 sequential data

RNN과 ConvNet 중에서 고민될 땐?

- 속도가 빠르고 가벼운 건 ConvNet, 긴 시퀀스에 강함.
- 정확도 높고 최근 데이터에 더 비중을 줄 수 있는 것은 RNN, LSTM