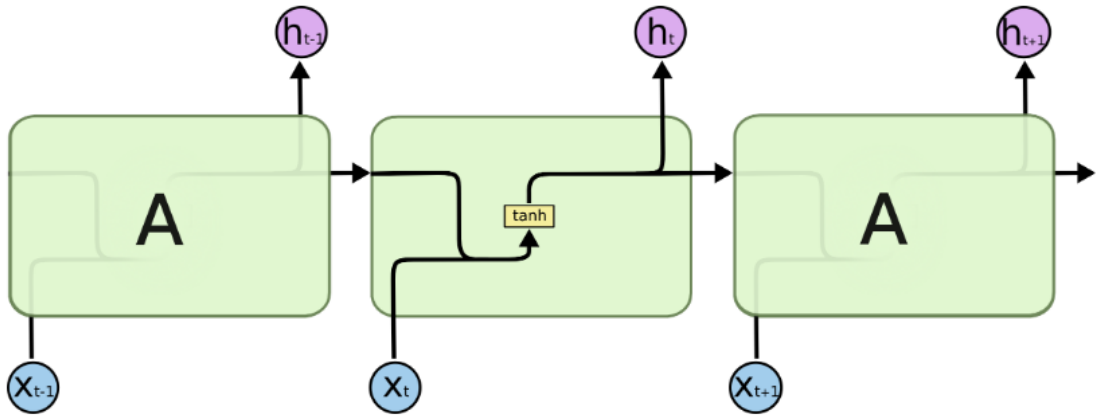
Closer Look at LSTM

Reference

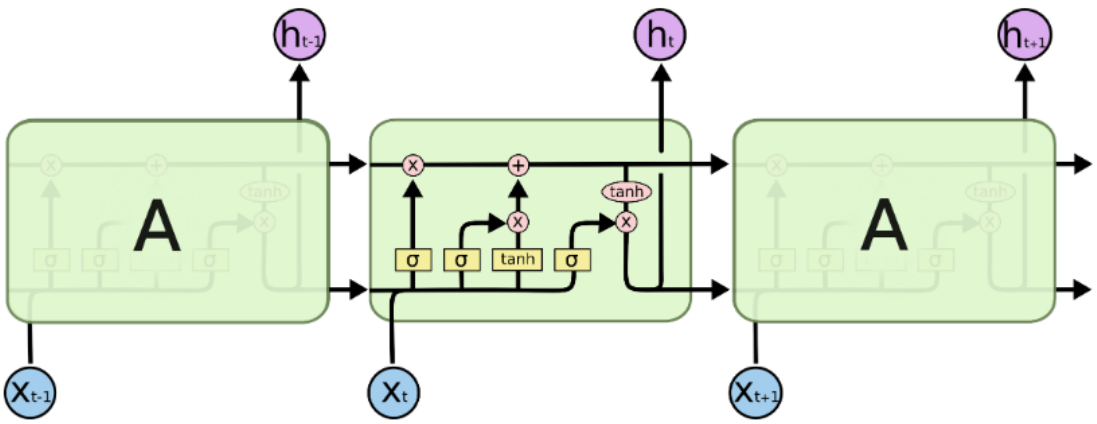
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/

사람들은 NLP task로 Language Modeling에 관심이 있었으며 가장 먼저 시도된 방법은 n-gram language modeling이다. 이전 sequence를 기반으로 다음에 어떤 단어가 나올지, 조건부 확률을 모델링한다. 하지만 이런 n-gram language modeling은 sparsity, storage 문제 등이 있었다.
n-gram language model의 대안으로, 신경만 기반 Recurrent Neural Network (RNN)가 제안되었다. sequential data에 최적화된 모델로, 처음 제안 되었을 당시 뛰어난 결과를 보였다. 기존의 n-gram language model이 가지고 있는 sparsity, storage 문제를 보안하며 주목을 받았다.
하지만 RNN조차 단점이 있었는데, 계산이 느리고 오래 전의 state에 대한 정보가 손실된다는 것이다. 계산이 느린 것은 둘째 치고, 오래 전의 state에 대한 정보가 손실된다는 점은 natural language같은 sequential data를 처리할 때, 큰 단점으로 다가왔다. 예를 들어, language modeling에서 바로 이전 단어가 아닌, 더 앞선 단어의 정보가 필요할 수도 있는데, RNN은 이를 잘 수행하지 못한다. 많은 연구에 의해 RNN이 vanishing gradient 문제를 가짐이 밝혀졌고 이러한 RNN에 대안으로 나온 것이 LSTM이다.

즉, LSTM은 RNN의 vanishing gradient 문제를 해결해주는 모델이다. 먼저, 기존의 standard RNN 구조를 살펴보자.

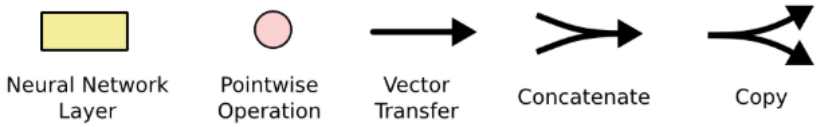


t번째 시점에서 input으로 x_t와 h_{t-1}가 들어온다. 그리고 이를 tanh activation function에 통과시켜 h_t를 만들고 이 hidden state가 다음 layer의 input으로 들어간다. LSTM의 구조는 아래와 같다.



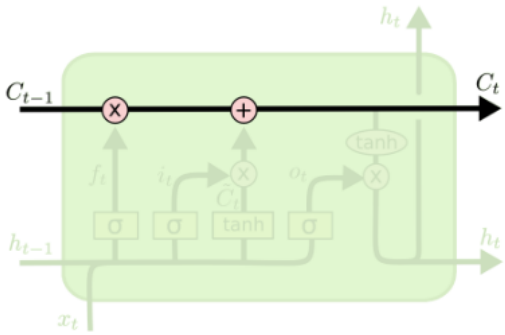
The repeating module in an LSTM contains four interacting layers.

즉, LSTM은 하나의 activation function으로만 layer가 구성되는 것이 아니라 네 개의, 서로 상호 작용하는 방법이 존재한다. 위 그림의 기호에 대한 정의는 아래와 같다.



Core Idea Behind LSTMs

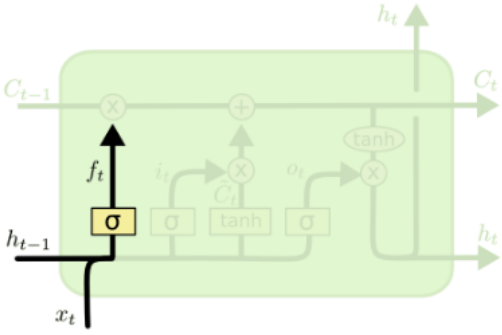
LSTM과 standard RNN의 가장 큰 차이점은 아래의 cell state가 있다는 것이다.



cell state는 conveyor belt 역할을 한다. 이전의 state에 대한 정보를 기억하거나 삭제하는 동시에 새로 들어오는 정보를 '선택적으로' 수용한다. 바로 이 역할을 'gates'가 한다. 이제 LSTM에서 어떤 gates가 있는지 알아보자.

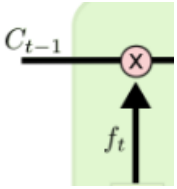
Step-by-Step LSTM Walk Through

1. Forget gate layer



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

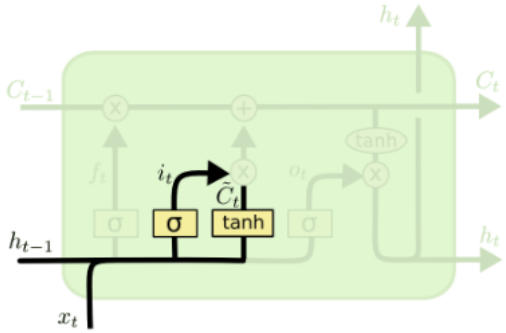
이 과정을 통해 이전 hidden state의 값과 현재 input이 들어오면 sigmoid function을 통해 0과 1사이의 값을 반환한다. 이 값을 아래와 같이 이전 cell state인 C_{t-1}에 곱한다.



즉, sigmoid 값이 1이라면 이전 cell state 정보가 온전히 전달되고 0이면 아예 삭제하는 것이다.

2. Input gate layer

다음으로는 어떤 새로운 정보를 cell state에 넘길지 결정한다.

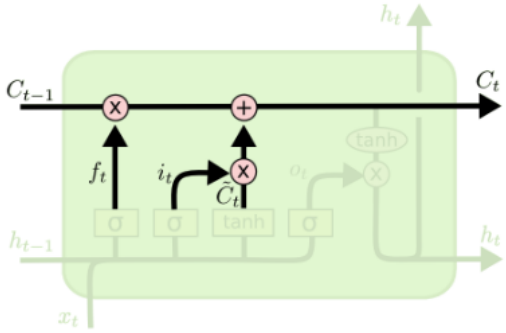


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

우선 위 그림에서 i_t를 sigmoid 값으로 만든다. 즉, 0과 1 사이의 값으로 $\tilde{C}_t$에 곱해져서 얼마만큼의 $\tilde{C}_t$를 전달할지 결정한다. 여기서 $\tilde{C}_t$는 tanh layer를 통과한 값으로, 다음 cell state로 통과할 후보이다. i_t 값이 1이라면 $\tilde{C}_t$ 모두를 cell state에 포함하고 0이라면 그 반대다.

3. Output gate layer

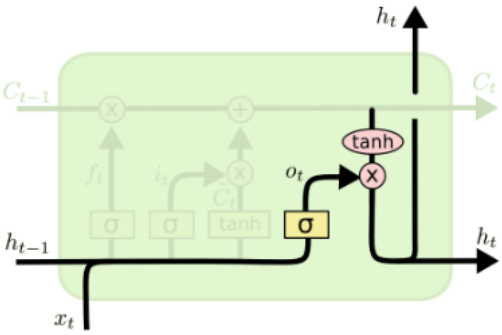
이제 이전의 cell state인 C_{t-1}를 C_t로 업데이트 한다. C_t를 C_{t-1}과 $\tilde{C}_t$의 combination으로 정의한다. 이 둘을 얼마나 포함할 지는, 'weight' 개념으로 sigmoid 값을 곱한다.



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

4. Sigmoid gate layer

마지막으로 output을 결정한다. 먼저 위에서 새로 계산한 C_t 중 얼만큼 output으로 내놓을지 계산해야 하는데, 그 비율을 o_t로 결정한다. 그리고 C_t를 tanh에 통과시켜 -1에서 1 사이의 값으로 만들고 sigmoid gate에 곱한다.



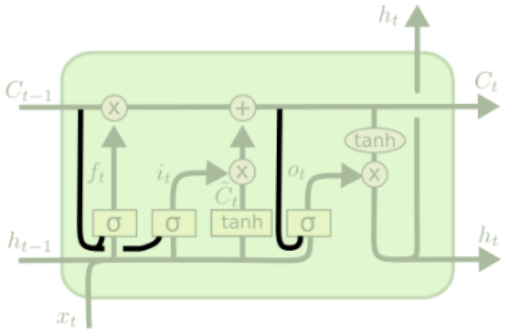
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Variants on Long Short Term Memory

여태까지는 normal LSTM에 대한 설명이었다. LSTM이 처음으로 나오고, 여러 변형들이 나왔다. 그 중 일부를 살펴보자.

peephole conections

normal LSTM에서 gate layers는 이전 cell state, C_{t-1}에 대한 정보를 반영하지 않았다. peephole connections는 아래와 같이 각 gaet layers가 C_{t-1}를 포함한다.

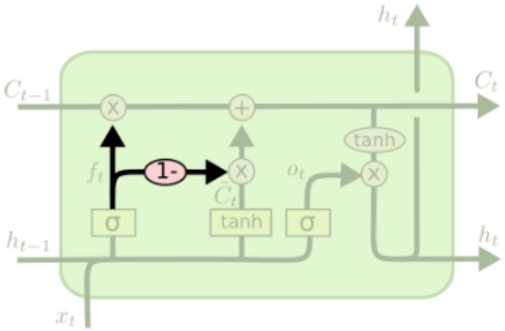


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

위 그림에서 진하게 표시된 C_{t-1}가 새롭게 추가된 내용이다.

Coupled forget and input gates

normal LSTM에서 output gate layer는 C_{t-1} 과 $\tilde{C}_t$의 linear comination으로 C_t를 만들었다. 각각의 계수는 f_t, i_t인데 이 둘이 합해서 꼭 1이라는 보장은 없다. Coupled forget and input gates에서는 linear combination의 계수의 합이 1이 되도록 한다. 즉 아래와 같이 C_{t-1} 과 $\tilde{C}_t$을 합친다.

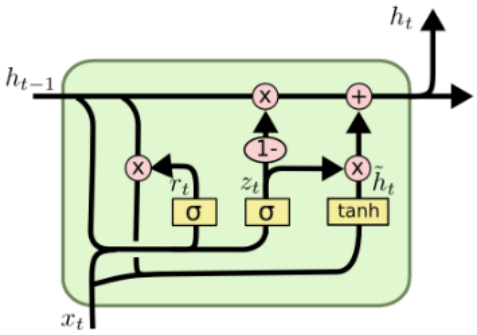


$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

위 그림을 잘보면, input gate가 없고 forget gate에서 '1-' 가 output gate layer로 연결되어 있다.

Gated Recurrent Unit (GRU)

LSTM의 cell state을 생략하고 이를 hidden state와 합쳤다. 현재의 hidden state인 h_t를 이전의 hidden state인 h_{t-1}과 $\tilde{h}_t$를 z_t를 weight로 해서 합한다. 여기서 $\tilde{h}_t$는 현재 state의 input인 x_t와 이전 hidden state인 h_{t-1}중 일부, $r_t \times h_{t-1}$ (여기서 r_t는 0에서 1의 값을 가지는 sigmoid output이다)을 tanh activation func에 통과시킨 값이다.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$