Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification

이봉석

- **Author**
  - Peng Zhou
  - Wei Shi
  - Jun Tian
  - Bingchen Li
  - Hongwei Hao
  - Bo Xu

- **Title of Conference(Journal)**
  - ACL 2016

01. Introduction

• Relation classification

• Pairs of nominals에서 의미 있는 relation들을 찾는 task
• 이 task는 information extraction, question answering같은 NLP applications에 유용하다.

Ex) ⟨e1⟩ Flowers ⟨/e1⟩ are carried into the ⟨e2⟩ chapel ⟨/e2⟩.
⇒ Entity-Destination relation

01. Introduction

• Relation classification

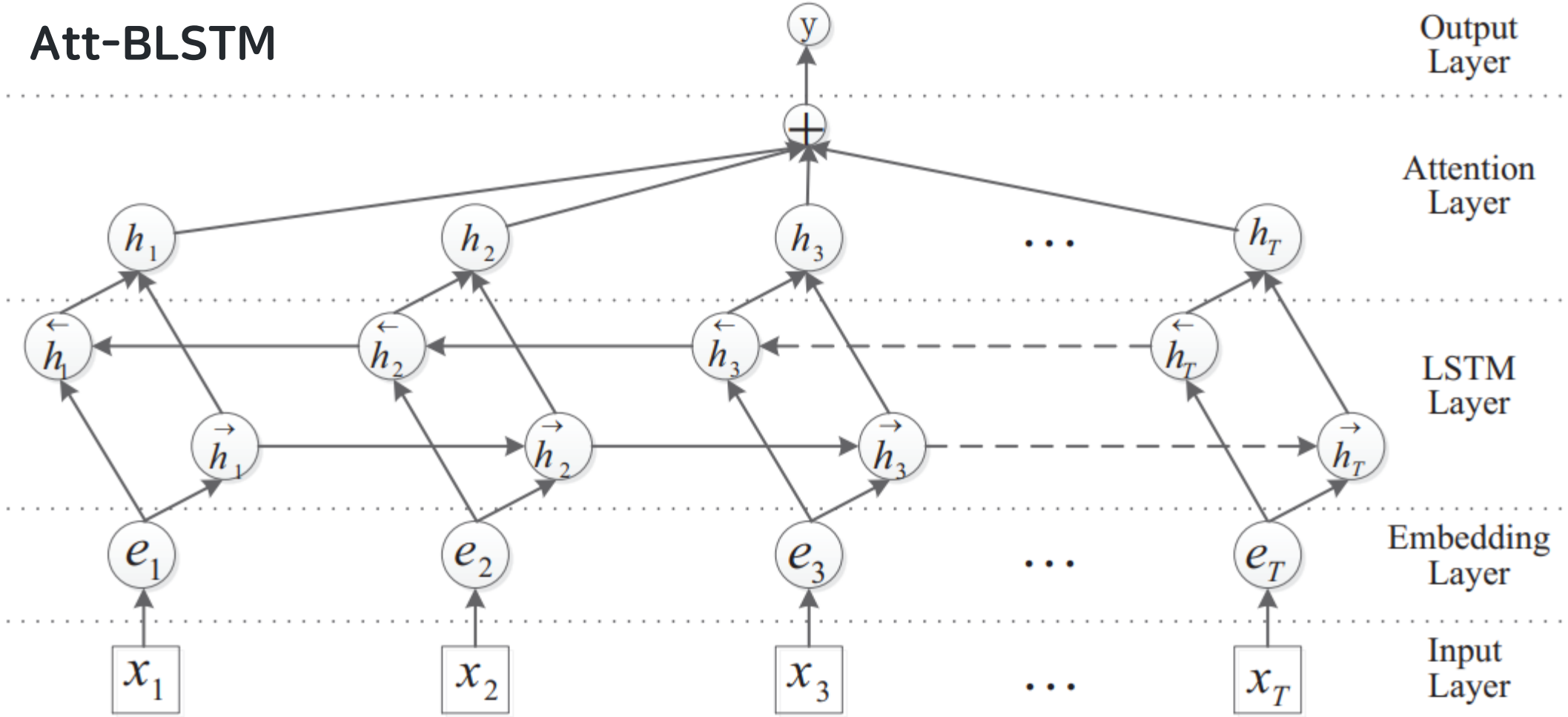
• 기존에는 lexical resources를 통해 pattern matching에 기반을 둔 방식이었다.

• 이러한 방법은 높은 수준의 features를 추출하는데 lexical resources와 NLP tool이 사용되어서 결과적으로 computational cost and additional propagation errors를 증가시킨다.

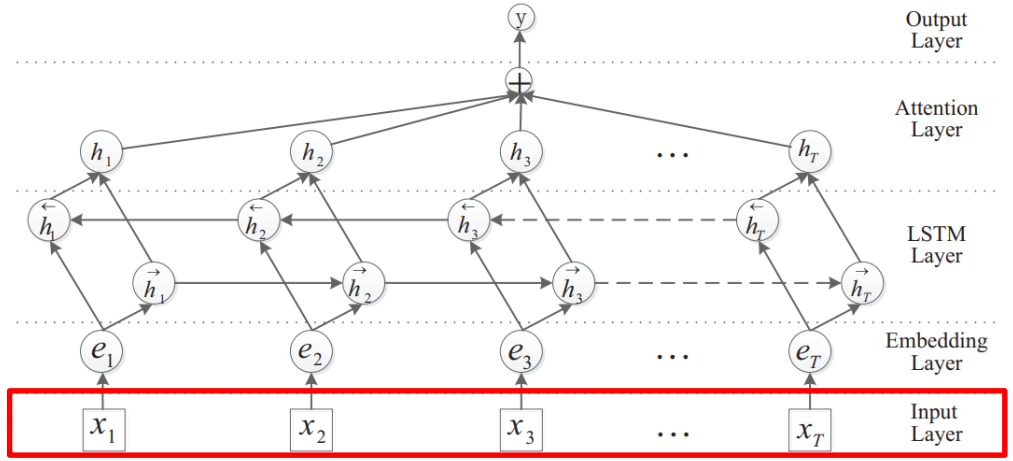
• 다른 단점은 수동으로 이런 features를 디자인 하는 것은 시간이 많이 걸리고 다른 training 데이터 세트의 적용 범위가 낮아 일반화를 제대로 수행하지 못한다.

02. Model

Att-BLSTM



02. Model



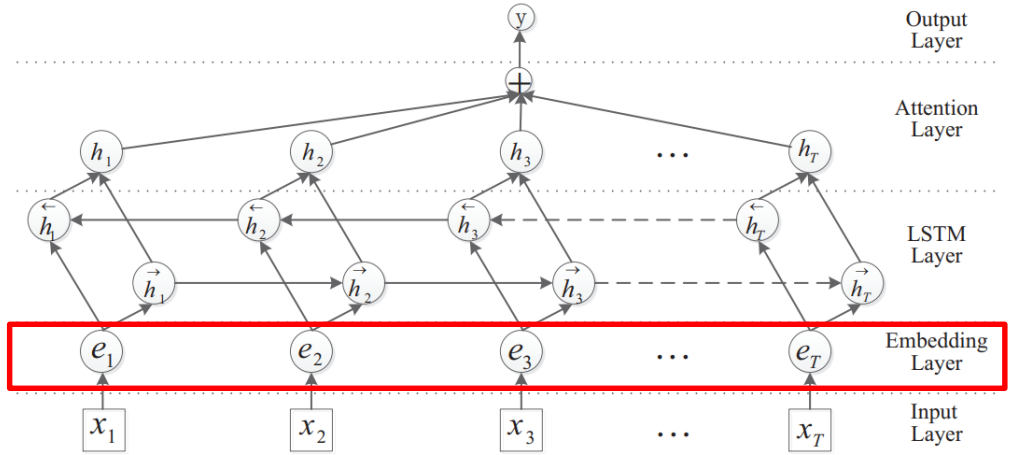
• Input Layer

• "The system as described above has its greatest application in an arrayed <e1>configuration</e1> of antenna <e2>elements</e2>."

• "The <e1>child</e1> was carefully wrapped and bound into the <e2>cradle</e2> by means of a cord."

• 이런식의 어떤 릴레이션을 가지고 있는 pair of nominals이 표시된 문장이 들어간다.

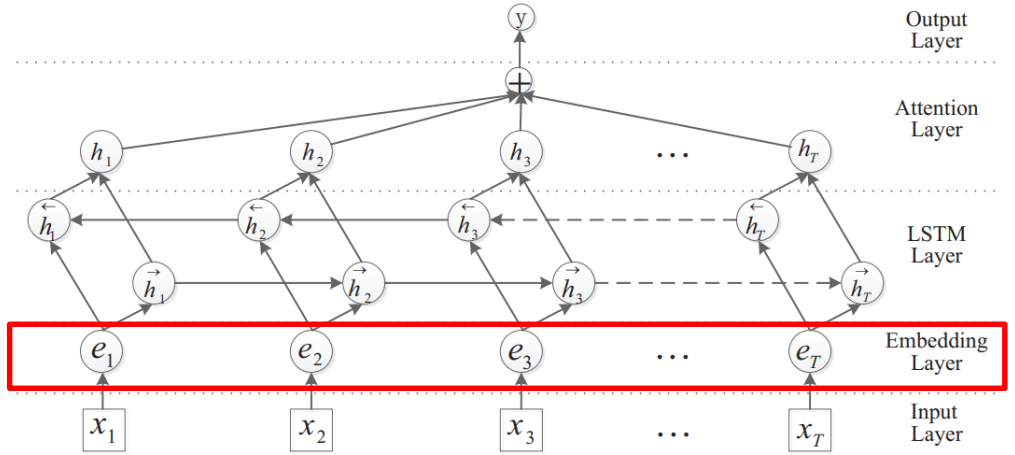
02. Model



• Embedding Layer

• Word embedding이란 컴퓨터가 어떤 단어에 대해 인지할 수 있게 하기 위해서 텍스트를 구성하는 하나의 단어를 수치화하는 방법이다. 즉, 단어를 특정 차원의 벡터로 매핑 시켜주는 것을 말한다.

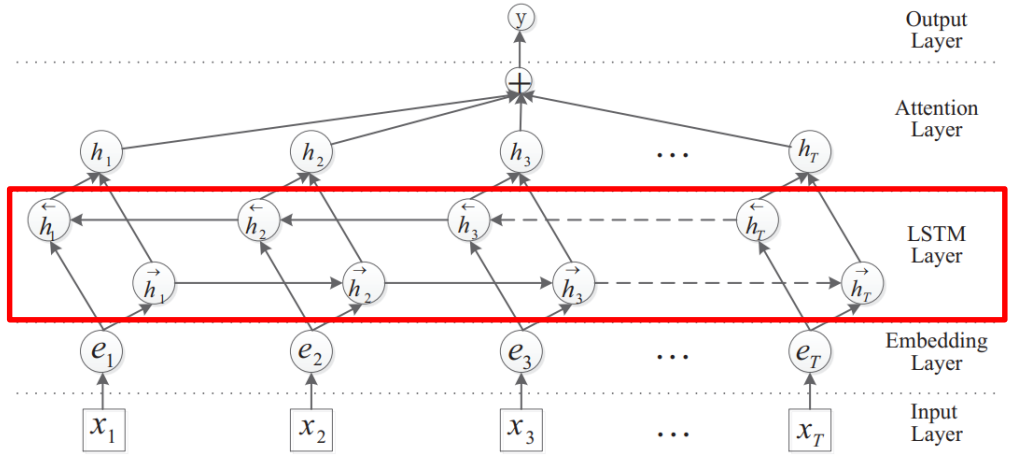
02. Model



• Embedding Layer

• "The <e1>child</e1> was carefully wrapped and bound into the <e2>cradle</e2> by means of a cord."
=>[e_{The}, $e_{<e1>}$, e_{child}, ⋯, e_{a}, e_{cord}]

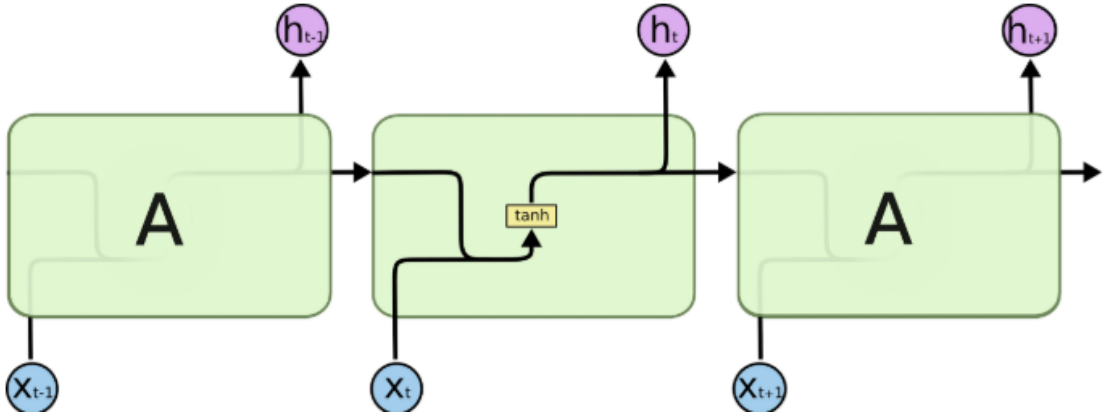
02. Model

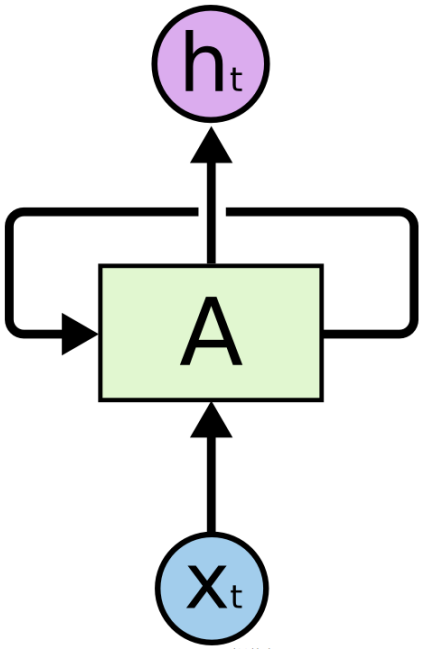


- LSTM Layer

02. Model

- RNN(Recurrent Neural Network)



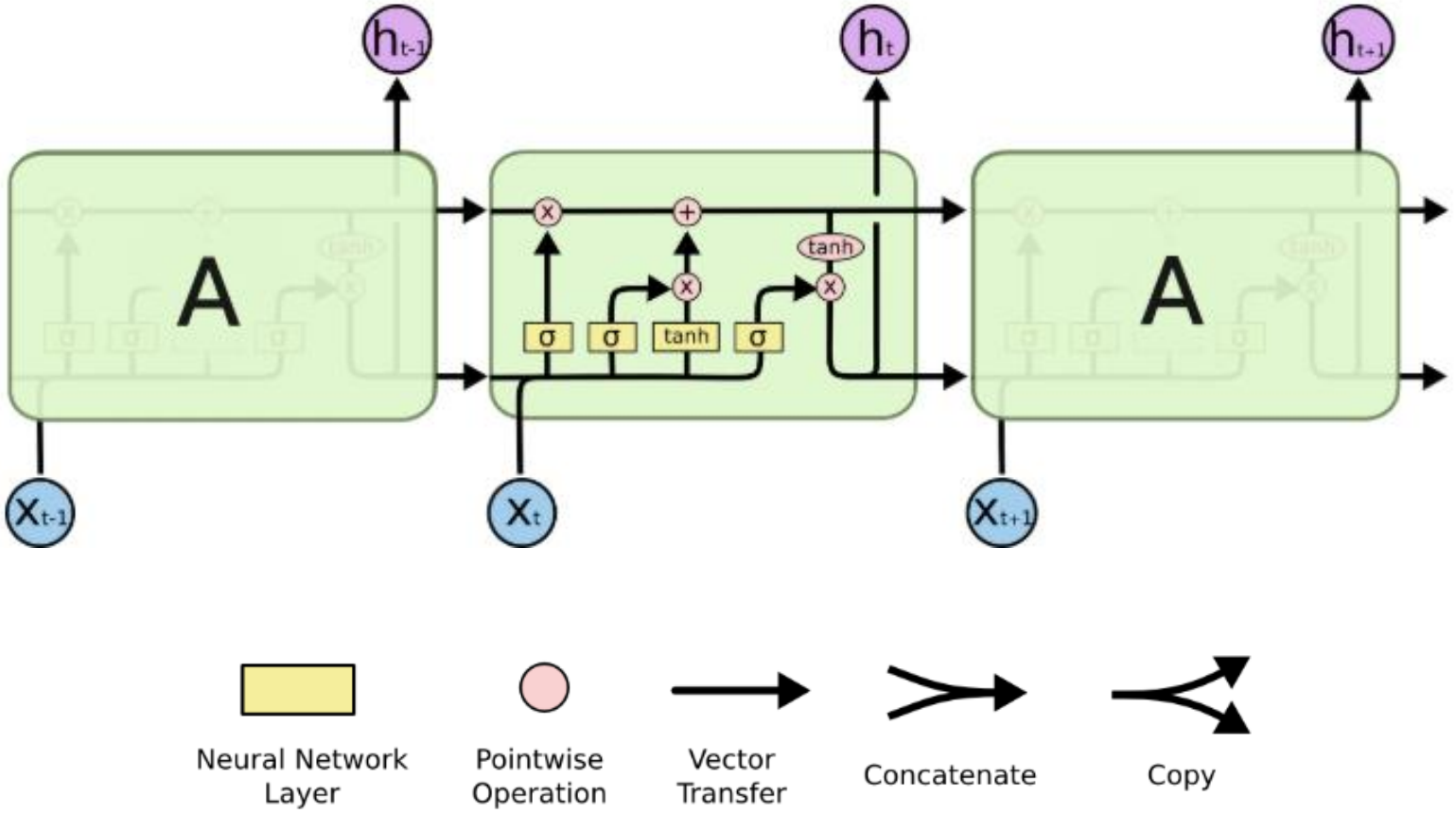


• 일반적인 신경망(feed-forward neural networks)와 다르게 한 노드의 결과가 다시 노드로 들어가도록 설계가 되어 있기 때문에 순서 또는 시간이라는 측면을 고려할 수 있는 특징을 가져다준다. 따라서 시계열 데이터를 학습하는데 유용한다!

• 하지만 이런 RNN은 시계열 데이터를 처리하는데 문제가 있다!
=> 시계열 데이터에서 시간적으로 멀리 떨어진, 장기(Long term)의존 관계를 잘 학습할 수 없다는 것입니다. (Long-Tern Dependency problem)

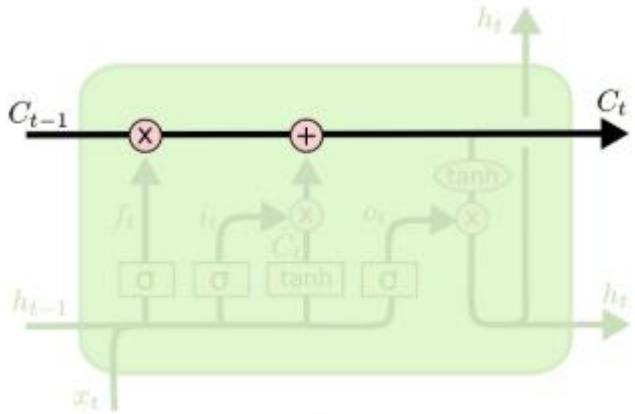
02. Model

- LSTM(Long Short-Term Memory)



02. Model

- LSTM(Long Short-Term Memory)



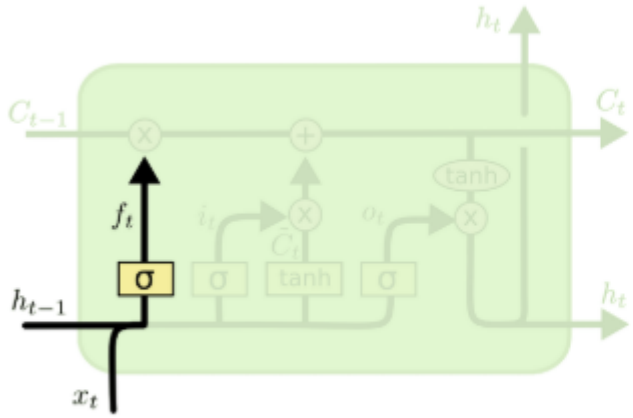
LSTM의 cell state

- LSTM의 핵심은 cell state인데 그림에서 수평으로 그어진 윗 선에 해당된다.

- LSTM은 cell state에 뭔가를 더하거나 없앨 수 있는 능력이 있는데, 이 능력은 gate라고 불리는 구조에 의해 제어된다.

02. Model

- LSTM(Long Short-Term Memory)



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

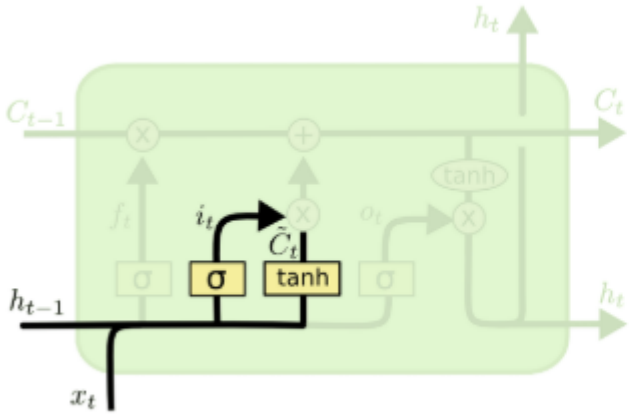
LSTM의 forget gate layer

- Forget gate layer

- LSTM의 첫 단계로 cell state로부터 어떤 정보를 버릴 것인지를 정하는 단계
- 이 단계에서는 h_{t-1}(단기기억)과 x_t를 받아서 0과 1사이의 값을 C_{t-1}(장기기억)에 보내준다. 그 값이 1이면 모든 정보를 보존하는 것이고 0이면 죄다 갖다 버리는 것이다.

02. Model

- LSTM(Long Short-Term Memory)



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

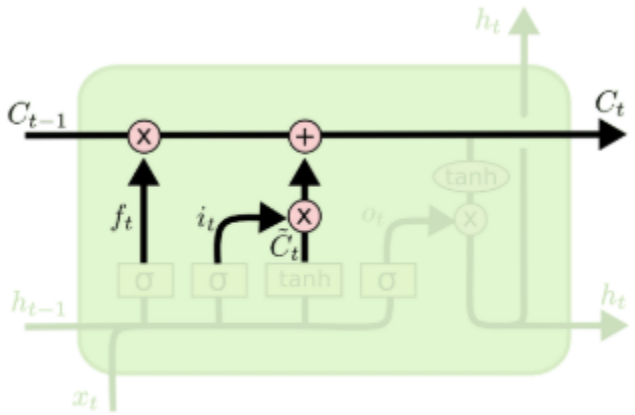
LSTM의 input gate layer

- Input gate layer

- 다음 단계는 앞으로 들어오는 새로운 정보 중 어떤 것을 cell state에 저장할 것인지를 정하다.
- h_{t-1}(단기기억)과 x_t를 받고 tanh layer를 통해 cell state에 추가할 새로운 정보 $\tilde{C}_t$를 만들고 이 정보를 어느 만큼 반영할지 정하기 위해 h_{t-1}(단기기억)과 x_t를 받아 시그모이드 layer를 통과한 값인 i_t를 만든다.

02. Model

- LSTM(Long Short-Term Memory)



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

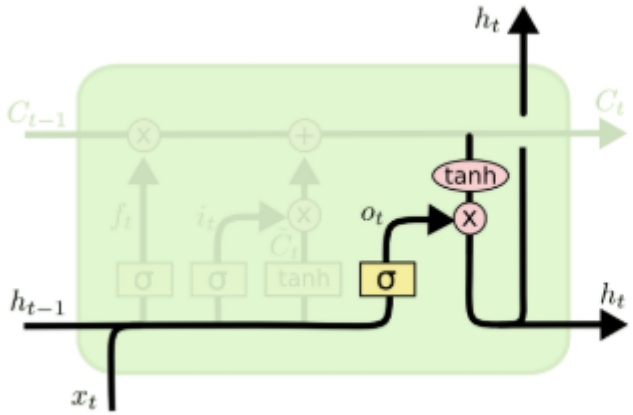
LSTM의 cell state 업데이트

- Cell state update

- 우선 이전 state(C_{t-1})에 f_t를 곱해서 가장 첫 단계에서 잊어버리기로 정했던 것들을 잊어버린다.
- 그 다음 cell state에서 잊어버리기로 정한 것들을 잊어버린 상태에서 새로운 정보 $\tilde{C}_t$와 반영비율 i_t를 곱한 값을 더하기 연산을 통해 cell state에 추가하여 cell state를 update한다.

02. Model

- LSTM(Long Short-Term Memory)



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

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

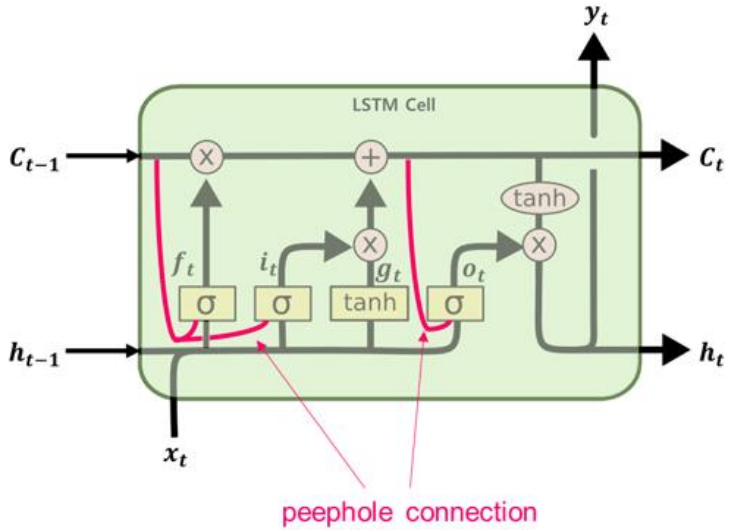
LSTM의 output gate layer

- Output gate layer.

- 이 output은 cell state를 바탕으로 필터된 값이 될 것이다. 가장 먼저, sigmoid layer에 input 데이터를 태워서 cell state의 어느 부분을 output으로 내보낼 지를 정한다.(o_t)
- 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.(h_t)

02. Model

- LSTM(Long Short-Term Memory) 변형

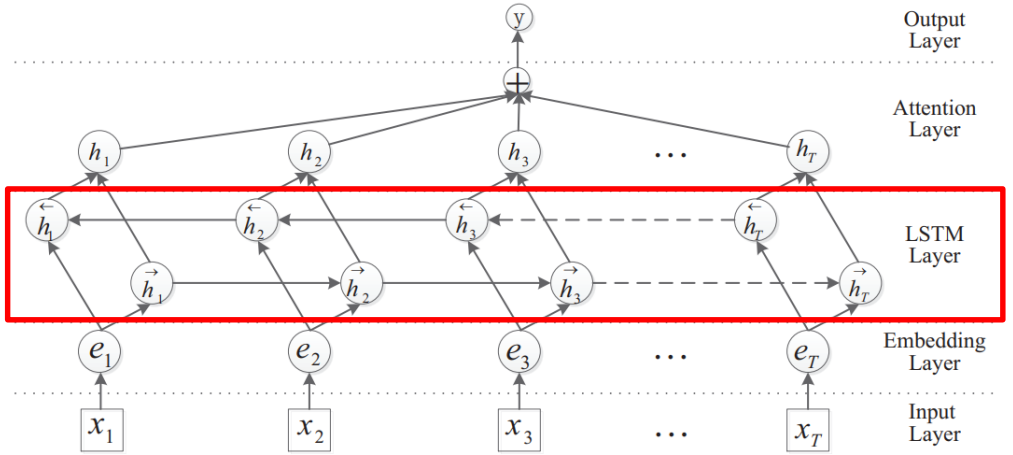


$$f_t = \sigma\left(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [C_t, h_{t-1}, x_t] + b_o\right)$$

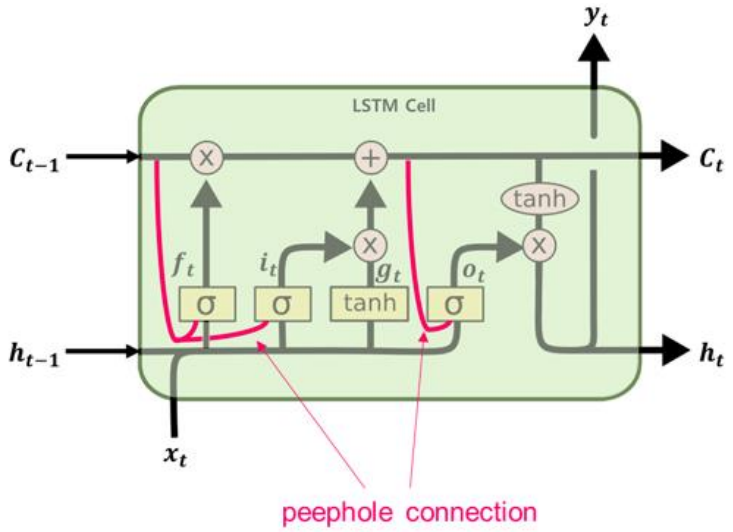
- peephole connection 버전

- Gate layer들이 cell state를 쳐다보게 만드는 모델이기 때문에 이런 이름이 붙여졌다.
- Gate controller에 이전 time step의 장기 상태 C_{t-1}가 입력으로 추가되면 보다 더 많은 맥락을 인식할 수 있다.
- 논문마다 어디에 엿보기 구멍(peephole)을 달고 어디에는 안 달고 할 수가 있다.
- Att-BLSTM의 경우 윗 그림에 $g_t(\tilde{C}_t)$가 만들어지는 부분에도 peephole이 추가로 있다.

02. Model

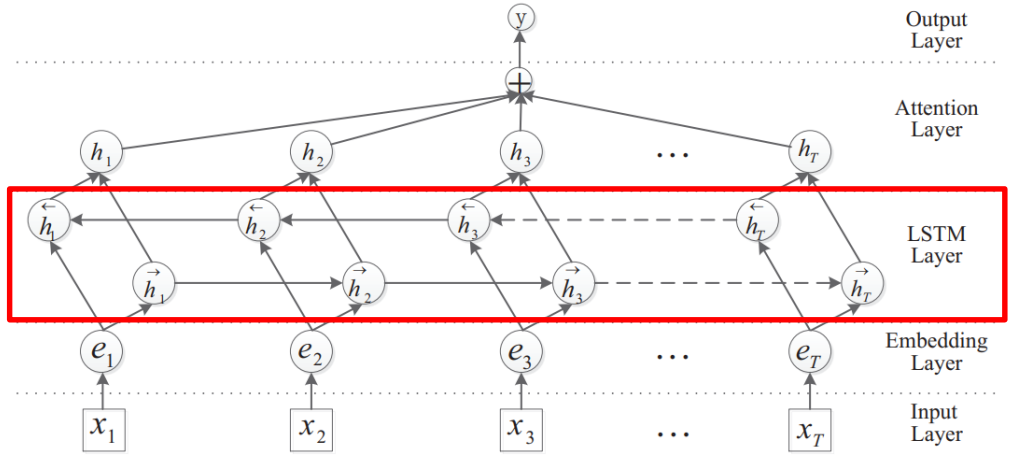


- LSTM Layer



$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
g_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + W_{cc}c_{t-1} + b_c) \\
c_t &= i_t g_t + f_t c_{t-1} \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

02. Model

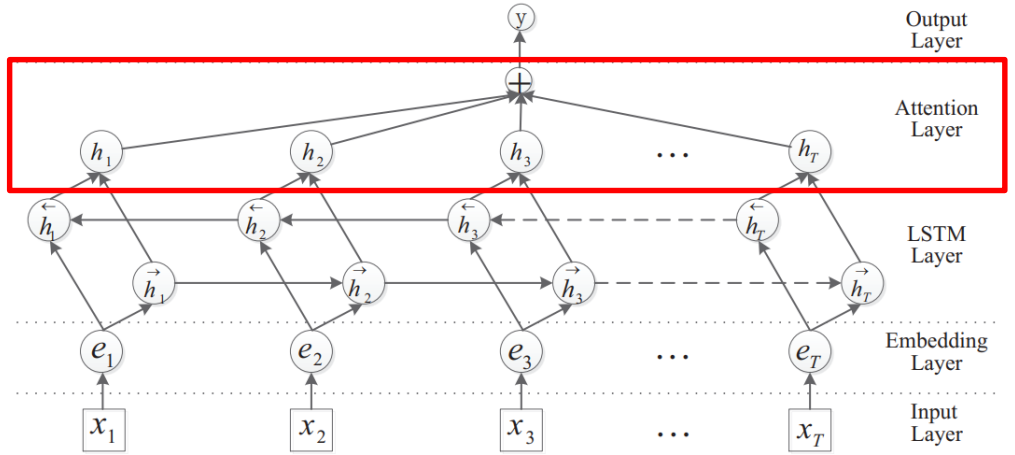


• BLSTM(Bidirectional LSTM) Layer

• 많은 sequence modelling task들은 과거의 정보뿐만 아니라 미래의 정보에 접근할 수 있으면 좋다.
• 하지만 LSTM은 시간의 순서로 데이터에 접근하기 때문에 미래의 정보를 무시한다.
• 그래서 시간의 순서가 반대로 흐르는 LSTM layer를 추가하여 LSTM 네트워크를 확장하여 과거와 미래 둘 다의 정보를 이용할 수 있는 BLSTM Layer를 사용한다.

• 따라서 i번째 단어에 대한 output은 $h_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}]$로 정의한다.

02. Model



• Attention Layer

• Attention의 기본 아이디어는 매 시점마다 나오는 output들을 전부 다 동일한 비율로 참고하는 것이 아니라 중요한 입력 부분을 좀 더 집중해서 보는 것입니다.
• 즉, 각 문장마다 LSTM Layer까지 거쳐 나온 output vectors들을 weighted sum하는 것이다.

• $H=[h_1, h_2, \cdots, h_T]$

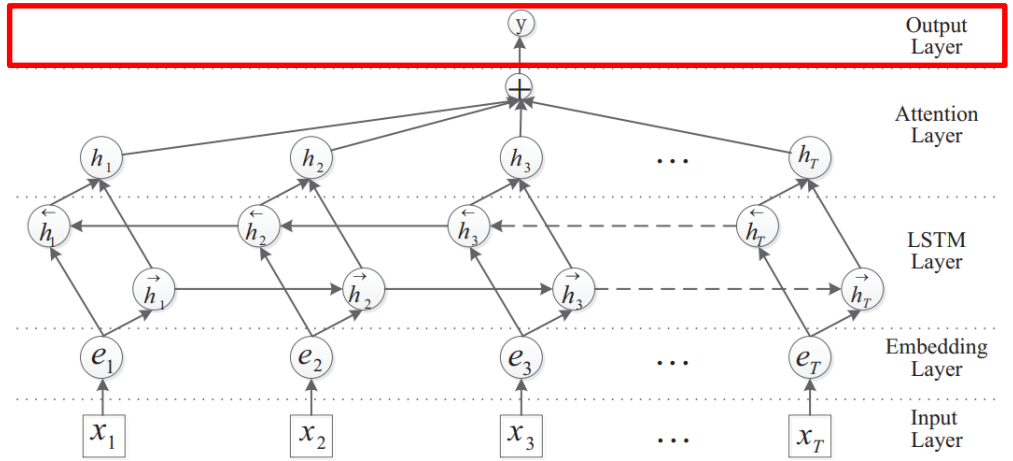
$$M = \tanh(H)$$

$$\alpha = softmax(w^T M)$$

$$r = H\alpha^T$$

$$h^* = \tanh(r)$$

02. Model



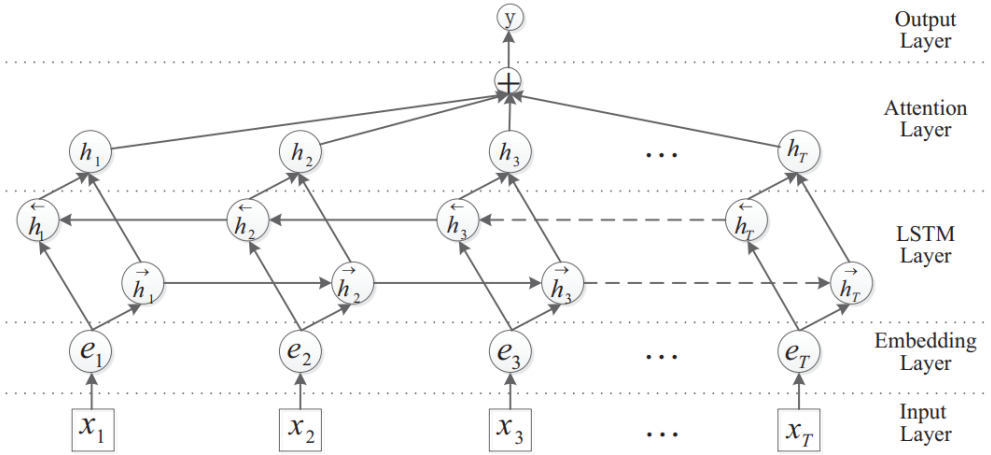
- **Classifying**

- Softmax

$$\hat{p}\left(y|S\right) = softmax\left(W^{(S)}h^{*} + b^{(S)}\right)$$

$$\hat{y} = \arg\max_{y} \hat{p}\left(y|S\right)$$

02. Model



- **Cost function(negative log-likehood)**

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} t_i \log(y_i) + \lambda \|\theta\|_F^2$$

02. Model

• Regularization

⇒ 오버피팅을 예방하고 일반화 성능을 높이는 것

• L2 regularization

⇒ 기존의 cost function에 가중치의 제곱을 포함하여 더하는 기법

⇒ 단순하게 cost값이 작아지는 방향으로만 진행되는 것이 아니라, w값들 역시 최소가 되는 방향으로 진행을 하게 된다.

• Dropout

⇒ 학습 과정에서 일정한 확률로 노드들을 사용하지 않게 만들어주는 기법

⇒ 특정 뉴런의 바이어스나 가중치가 큰 값을 가지게 되면 그것의 영향이 커지면서 다른 뉴런들의 학습 속도가 느려지거나 학습이 제대로 진행이 되니 못하는 경우가 있다. 따라서 dropout을 하면 결과적으로 어떤 뉴런의 가중치나 바이어스가 특정 뉴런의 영향을 받지 않기 때문에 오버피팅을 방지할 수 있다.

03. Experiments

- Dataset : SemEval-2010 Task 8
- Optimization : AdaDelta
- learning rate : 1.0
- minibatch size : 10
- L2 regularization strength : 10^{-5}
- Dropout : embedding layer 0.3, LSTM layer 0.3, Attention layer 0.5

03. Experiments

Model	Feature Set	F1
SVM (Rink and Harabagiu, 2010)	POS, prefixes, morphological, WordNet, dependency parse, Levin classed, ProBank, FramNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
CNN (Zeng et al., 2014)	WV (Turian et al., 2010) (dim=50) + PF + WordNet	69.7 82.7
RNN (Zhang and Wang, 2015)	WV (Turian et al., 2010) (dim=50) + PI WV (Mikolov et al., 2013) (dim=300) + PI	80.0 82.5
SDP-LSTM (Yan et al., 2015)	WV (pretrained by word2vec) (dim=200), syntactic parse + POS + WordNet + grammar relation embeddings	82.4 83.7
BLSTM (Zhang et al., 2015)	WV (Pennington et al., 2014) (dim=100) + PF + POS + NER + WNSYN + DEP	82.7 84.3
BLSTM Att-BLSTM BLSTM Att-BLSTM	WV (Turian et al., 2010) (dim=50) + PI WV (Turian et al., 2010) (dim=50) + PI WV (Pennington et al., 2014) (dim=100) + PI WV (Pennington et al., 2014) (dim=100) + PI	80.7 82.5 82.7 84.0

Table 1: Comparison with previous results. WV, PF, PI stand for word vectors, position features and position indicators respectively.

04. Conclusion

• NLP 도구나 lexical resources에 의존하지 않고 position indicator를 가진 raw text를 인풋으로 사용한다!

• 간단하지만 성능이 괜찮다!

감사합니다.