seq2seq & attention

Sequence to Sequence Learning with Neural Networks (NIPS 2014)

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE (ICLR 2015)

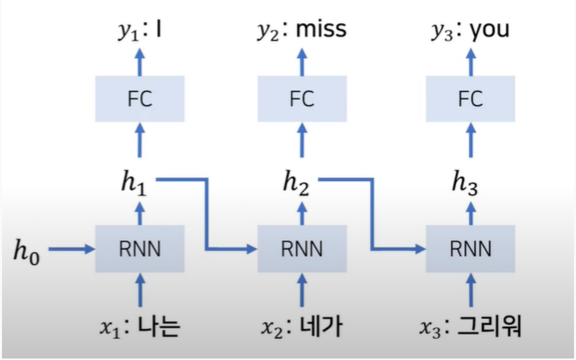
Hankyo Jeong

September 6, 2024

Seoul National University

Traditional RNN model

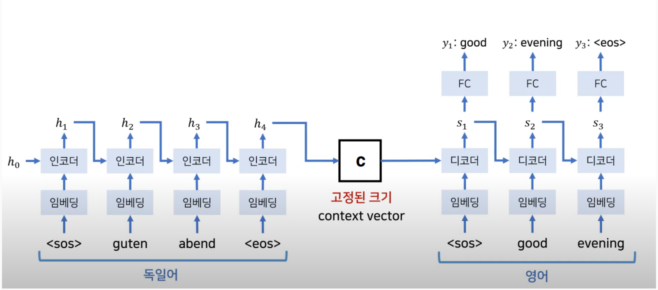
- $sequence\ of\ inputs : (x_1, ..., x_T)$, T is length of input sequence
- $sequence\ of\ outputs : (y_1, ..., y_T)$
- $h_t = sigm(W^{hx}x_t + W^{hh}h_{t-1})$
- $y_t = W^{yh}h_t$



Limitation & Alternative

- Since traditional RNN model assume that length of input sequence and output sequence are equal, it is not clear how to apply an RNN to problems whose input and the output sequences have different length
- The strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN
  : **Encoder - Decoder (seq2seq) architecture**
- It would be difficult to train the RNNs due to the resulting long term dependencies. $\rightarrow$ LSTM

seq2seq



- *sequence of inputs* : $\mathbf{x} = (x_1, ..., x_{T_x})$
- *sequence of outputs* : $\mathbf{y} = (y_1, ..., y_{T_y})$
- $h_t = f(x_t, h_{t-1}),\ c = q(h_1, ..., h_{T_x})$
  - Sutskever et al. (2014): f is LSTM, $q(h_1, ..., h_{T_x}) = h_T$
- $p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}), c)$
- $p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$

Limitation of seq2seq based LSTM

- A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector.
- This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus.

▶ JOINTLY LEARN TO ALIGNMENT AND TRANSLATION

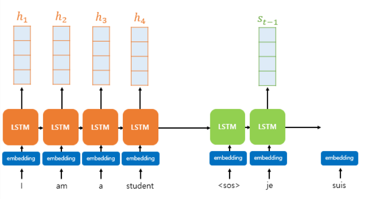
Soft alignment (attention)

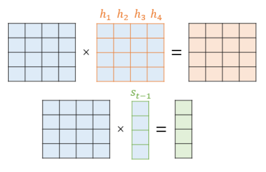
- **Soft alignment(attention)**: the model dynamically assigns importance to each word in the input sentence, deciding which words to focus on for each prediction.

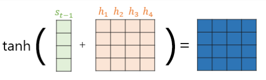
1. Obtain Attention score: $e_{ij} = a(s_{i-1}, h_j) = v_a^T \tanh\left(W_a s_{i-1} + U_a h_j\right)$
   - alignment model which scores how well the inputs around position j and the output at position i match.
   - h_j: hidden state of encoder, s_i: hidden state of decoder
2. Obtain Attention distribution: $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$
3. Obtain context vector(attention value): $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$
4. Obtain s_i from context vector: $f(s_{i-1}, y_{i-1}, c_i)$

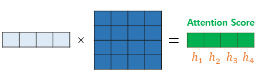
Attention

1. Attention score: $e_{ij} = a(s_{i-1}, h_j) = v_a^T \tanh\left(W_a s_{i-1} + U_a h_j\right)$



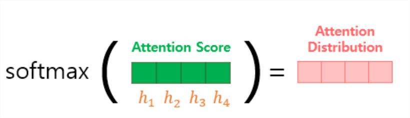




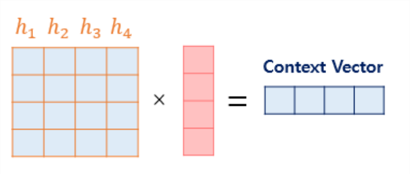


Attention

2. Obtain Attention distribution: $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$

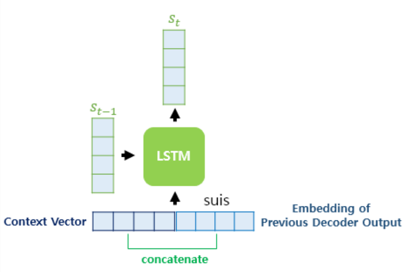


3. Obtain context vector(attention value): $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$



Attention

4. Obtain s_i from context vector: $f(s_{i-1}, y_{i-1}, c_i)$



Bidirectional RNN

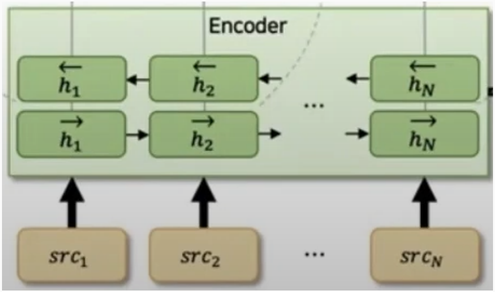
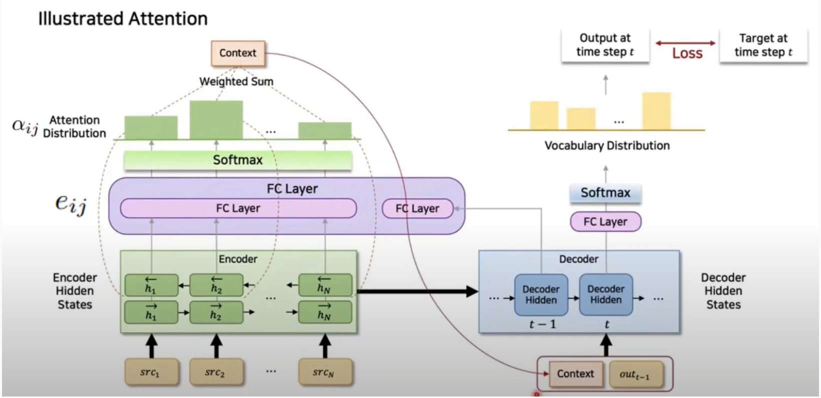


Figure 1: bidirectional RNN

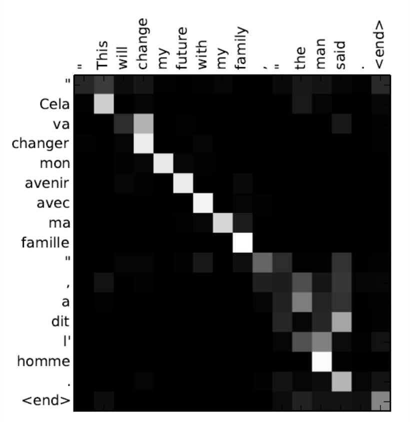
ENCODER: Bidirectional RNN for annotation sequences

- RNN: reads input sequence in order from x_1 to x_{T_x} (forward)
- BiRNN: forward RNN($\overrightarrow{f}$) + backward RNN($\overleftarrow{f}$)
  - forward RNN:
  calculates a sequence of forward hidden states $((\overrightarrow{h_1}, ..., \overrightarrow{h_{T_x}})$.
  - bakcward RNN:
  calculates a sequence of backward hidden states $((\overleftarrow{h_1}, ..., \overleftarrow{h_{T_x}})$.
- Obtain annotation for each word x_j by concatenating the forward hidden state and backward hidden state. $h_j = [\overrightarrow{h_j^T}, \overleftarrow{h_j^T}]^T$
  - In this way the annotation h_j contains the summaries of both the preceding words and the following words.

Attention mechanism

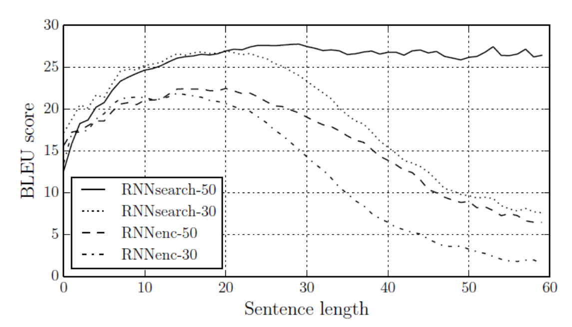


RESULTS



Each cell: attention score α_{ij}, balck: 0, white: 1

RESULTS



proposed model is robust than basic RNN model