Attention-based LSTM with Aspect Embedding (ATAE-LSTM) Explained

Zhimao Lin

December 15, 2021

1 Attention-based LSTM with Aspect Embedding (ATAE-LSTM)

ATAE-LSTM [4] is an aspect-based sentiment analysis method based on Long Short-term Memory (LSTM) [1]. It encodes the aspect information into the word embedding and the outputs of the LSTM layer. Then, the LSTM outputs with aspect information added are sent to an attention layer, and the sentiment prediction is generated by another typical feedforward neural network using the outputs of the attention layer.

1.1 Long Short-term Memory (LSTM)

LSTM is an extension of Recurrent Neural Network (RNN) [3], which takes the previous output as an additional input for the current run. However, RNN cannot carry critical information forward because it is difficult for weights to care about both previous context and future context [2]. Also, as the sentences become longer, vanishing gradients issue occurs during the backpropagation. On the other hand, LSTM (Figure 1) [2] uses multiple gates to solve the problems that RNN has. It takes the current word x_t and the previous output h_{t-1} and context c_{t-1} as inputs and outputs current context c_t and output h_t.

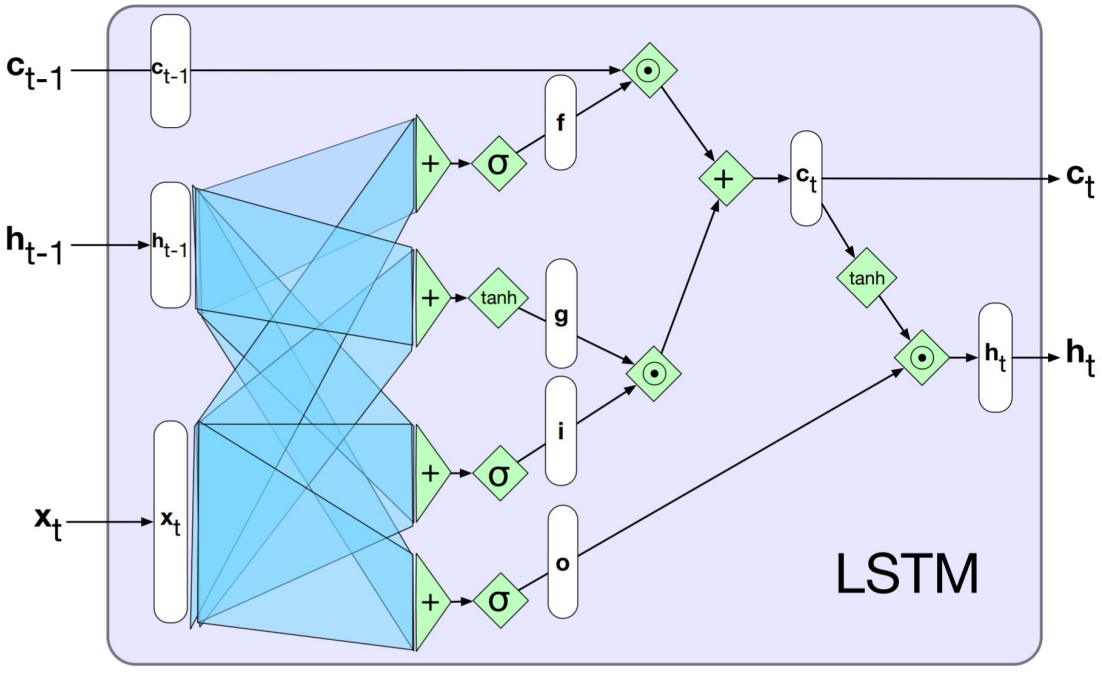

Figure 1: A single LSTM unit. Here, t represents the current step, and $t-1$ represents the previous step.

LSTM has forget gate (f), add gate (i), and output gate (o). Each gate acts like a filter which filters the target by computing the element-wise product between the gate and the target.

$$k = f \odot c_{t-1}$$

$$j = g \odot i$$

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

The gate filters are calculated by applying the sigmoid function to the different weighted sums of h_{t-1} and x_t.

$$f = \sigma(U_f h_{t-1} + W_f x_t)$$

$$i = \sigma(U_i h_{t-1} + W_i x_t)$$

$$o = \sigma(U_o h_{t-1} + W_o x_t)$$

Forget gate The forget gate removes unnecessary information from the previous context so that more important information can be passed along the neural network.

Add gate The add gate picks critical information and adds to the current context. The current context is calculated by applying the hyperbolic tangent function to the weighted sum of h_{t-1} and x_t, which is combined with the previous filtered context.

$$g = \tanh(U_g h_{t-1} + W_g x_t))$$

$$c_t = j + k$$

Output gate The output gate filters the hyperbolic tangent of the current context c_t and only outputs important information for the next LSTM unit.

With these gates, LSTM can decide which information should be passed forward and which information should be discarded.

1.2 Attention

The attention mechanism allows the layer to directly access the information in the previous context without passing that information one by one along with the network. For example, in the LSTM, it does not have direct access to the word x_{t-100} at the time t. Instead, it has to wait for the network to pass the information of x_{t-100} to the c_{t-1} and h_{t-1}. An attention layer can solve this problem by connecting each word to the layer as shown in Figure 2 [2].

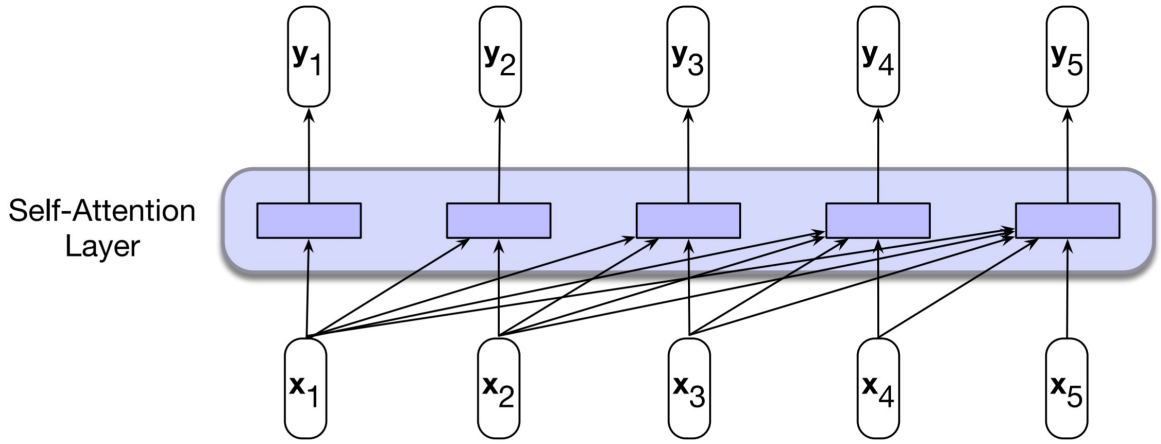

Figure 2: A typical self-attention layer

In this figure, when calculating the output y_5, the layer has direct access to all the words $\{x_1, x_2, x_3, x_4, x_5\}$. With this direct access, the layer can freely decide which words are more important to its task, and it can pay more attention to those words.

1.3 ATAE-LSTM

The ATAE-LSTM [4] takes advantages of both LSTM and attention mechanism. It adds an attention layer after the LSTM layer. Also, it encodes the aspect information to the word embedding and the outputs of LSTM by appending the aspect embedding. The model is illustrated in Figure 3 [4].

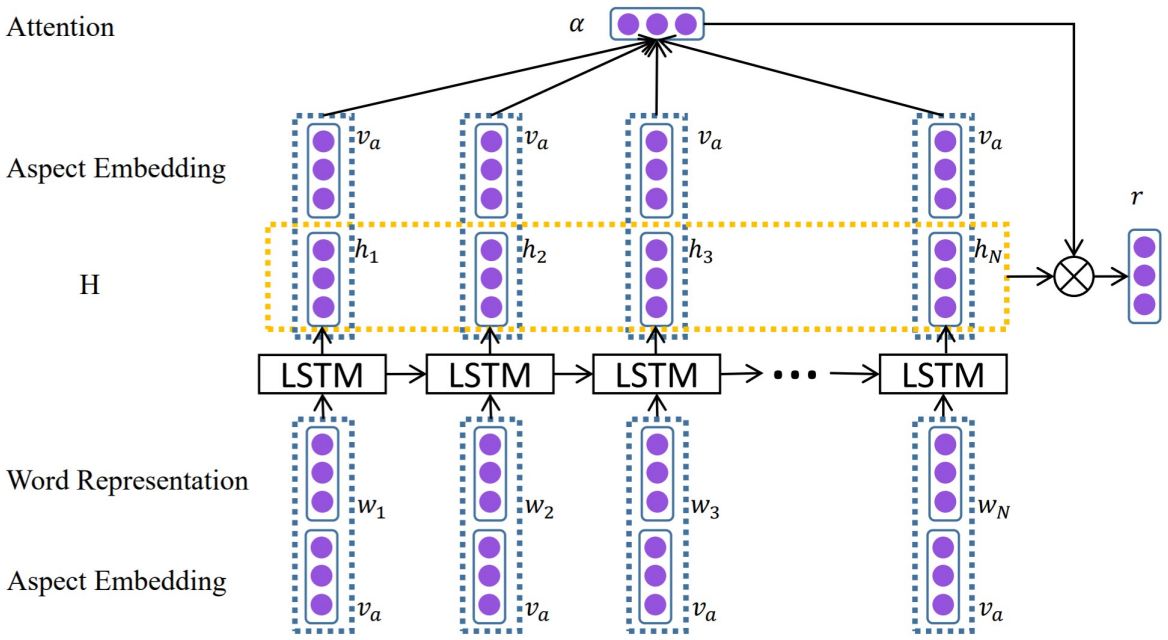

Figure 3: Attention-based LSTM with Aspect Embedding (ATAE-LSTM)

Here, $w_i \in R^d$ is the word embedding of each word in the sentence with length of N. $v_a \in R^{d_a}$ is the aspect embedding. By appending the aspect embedding to the word embedding, we can get a $(d+d_a) \times 1$ vector: $(-w_i - | - v_a -)^T$. Then, the aspect information can be propagated to the outputs of the LSTM. Here, we use $h_i \in R^d$ to represent the output of LSTM on each word w_i. Let H be a matrix in $R^{d \times N}$, so

$$H = \begin{bmatrix} | & | & | & | \\ \mathrm{h}_1 & \mathrm{h}_2 & \dots & \mathrm{h}_N \\ | & | & | & | \end{bmatrix}_{d \times N}$$

Additionally, we append the weighted aspect embedding to the output of LSTM. Let A be a matrix in $R^{d_a \times N}$, so

$$A = \begin{bmatrix} | & | & | & | \\ \mathrm{v}_a & \mathrm{v}_a & \dots & \mathrm{v}_a \\ | & | & | & | \end{bmatrix}_{d_a \times N}$$

Let $W_h \in R^{d \times d}$ and $W_v \in R^{d_a \times d_a}$ be the weight matrices of H and A, and let K be a matrix in

$R^{(d+d_a)\times N}$, so

$$K = \begin{bmatrix} \mathrm{W}_h H \\ \mathrm{W}_v A \end{bmatrix}_{(d+d_a)\times N} = \begin{bmatrix} | & | & | & | \\ \mathrm{w}_{h_1} h_1 & \mathrm{w}_{h_2} h_2 & \dots & \mathrm{w}_{h_N} h_N \\ | & | & | & | \\ - & - & - & - \\ | & | & | & | \\ \mathrm{w}_{v_1} v_a & \mathrm{w}_{v_2} v_a & \dots & \mathrm{w}_{v_N} v_a \\ | & | & | & | \end{bmatrix}_{(d+d_a)\times N}$$

Then, the following calculations are performed.

$$M = \tanh(K)$$

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

$$r = H\alpha^T$$

$$h^* = \tanh(W_p r + W_x h_N)$$

$$y = softmax(W_s h^*)$$

The matrix M which contains the information of each word gets passed to the attention layer. In the end, a linear layer and a softmax function are used to generate the prediction y, which has a dimension of 3 since there are three sentiment classes.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, 2nd edition.* Prentice Hall, USA, 2nd edition, 2008.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986.

[4] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas, November 2016. Association for Computational Linguistics.