Mathematical Description (Training procedure)

The model uses a bidirectional LSTM for source language sentence encoding. Let $(x_1, \dots, x_m)$ be the embeddings of the source sentence, with each $x_i \in \mathbb{R}^{e \times 1}$ for $i \in [m]$. The last hidden/cell state of the forward directional encoder and the first hidden/cell state of the backward directional encoder are concatenated and projected via linear transforms $W_h, W_c \in \mathbb{R}^{h \times 2h}$:

$$h_0^{dec} = W_h[\overleftarrow{h_1^{enc}}; \overrightarrow{h_m^{enc}}], \quad c_0^{dec} = W_c[\overleftarrow{c_1^{enc}}; \overrightarrow{c_m^{enc}}]$$

The decoder's hidden state dynamics follow:

$$h_t^{dec}, c_t^{dec} = Decoder(\overline{y_t}, h_{t-1}^{dec}, c_{t-1}^{dec})$$

The hidden state h_t^{dec} is used to compute multiplicative attention over $h_1^{enc}, \dots, h_m^{enc}$. Attention scores with respect to the encoders over the time sequence (let's say m) are computed as follows, for all $i \in [m]$:

$$e_{t,i} = \langle h_t^{dec}, W_{attProj} h_i^{enc} \rangle, \quad W_{attProj} \in \mathbb{R}^{h \times 2h}$$

The attention probabilities (weights) are:

$$\alpha_t = softmax(e_t), \quad e_t \in \mathbb{R}^{m \times 1}$$

The attention output is computed as:

$$a_t := \sum_{i=1}^{m} \alpha_{t,i} h_i^{enc} \in \mathbb{R}^{2h \times 1}$$

Next, we concatenate the attention output and the hidden state of the decoder to get $u_t = [a_t; h_t^{dec}] \in \mathbb{R}^{3h \times 1}$. Using the combined output projection matrix $W_u \in \mathbb{R}^{h \times 3h}$, we compute:

$$v_t = W_u u_t, \quad o_t := dropout\big(\tanh(v_t)\big)$$

If V_{tgt} is the vocabulary size of the target language, we use the target vocab projection matrix $W_{vocab} \in \mathbb{R}^{V_{tgt} \times h}$ to compute the output probabilities:

$$P_t = softmax(W_{vocab} o_t) \in \mathbb{R}^{V_{tgt} \times 1}$$

On the t-th step, we look up the embedding of the t-th subword $y_t \in \mathbb{R}^{e \times 1}$ and concatenate it with $o_{t-1} \in \mathbb{R}^{h \times 1}$ to get the input for the decoder at the t-th step:

$$\overline{y_t} := [y_t; o_{t-1}] \in \mathbb{R}^{(h+e) \times 1}$$

Finally, to train the network, we compute the softmax cross-entropy loss between P_t and g_t, where g_t is the one-hot vector of the target subword at timestep t. The associated loss at the t-th decoding step is:

$$J_t(\theta) := CrossEntropy(P_t, g_t)$$

Here, θ represents all the trainable parameters of the architecture.