A Short Tutorial of the Attention Mechanism in Natural Language Processing

Spencer Whitehead
whites5@rpi.edu

This is a short tutorial of the attention mechanism found in a wide variety of NLP tasks. Though there are many variants of the attention mechanism, we will focus on one of the original formulations which is from Bahdanau et al., 2015. At the end, you will answer some questions that will walk you through some of the details.

The attention mechanism is used in conjunction with an encoder-decoder model to dynamically weight the encoder hidden states as the decoder is generating tokens (words). We first define our encoder-decoder model architecture and then define the attention mechanism. We omit bias terms for brevity, but, if one were to implement this, bias terms could be included.

We are given an input vocabulary of size V', an input sequence $x = (x_1, \ldots, x_{T_x})$, and an output vocabulary of size V. We want to generate an output sequence $y = (y_1, \ldots, y_{T_y})$. Each word x_i in the input sequence has a vector $\mathbf{x}_i$ that is a V'-dimensional 1-hot vector, with the location of the 1 corresponding to word x_i in the input vocabulary. The vectors $\mathbf{y}_t$ are defined in the same fashion, but are V-dimensional and the location of the 1 correpsonds to word y_t in the output vocabulary.

Both our encoder and decoder are vanilla, uni-directional recurrent neural networks:

$$\mathbf{h}_i = f(\mathbf{x}_i, \mathbf{h}_{i-1}) = \tanh(W_x(\hat{E}\mathbf{x}_i) + W_h\mathbf{h}_{i-1}), i = 1, \ldots, T_x \tag{1}$$

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}_t) = \tanh(W_y(E\mathbf{y}_{t-1}) + W_s\mathbf{s}_{t-1} + W_c\mathbf{c}_t), t = 1, \ldots, T_y \tag{2}$$

$$P(y_t|y_1, \ldots, y_{t-1}, x) = \text{softmax}(\phi(\mathbf{s}_t, E\mathbf{y}_{t-1}, \mathbf{c}_t)) \tag{3}$$

Here all W are learned parameters, $\mathbf{h}_i$ and $\mathbf{s}_t$ are the encoder and decoder hidden states, $\mathbf{c}_t$ is the context vector, and ϕ is a function that produces a V-dimensional vector. $\hat{E}$ and E are embedding matrices. Note, if our encoder is a bi-directional recurrent neural network, then we simply have two encoder layers where one layer computes a forward hidden state, $\overrightarrow{\mathbf{h}_i}$, and the other computes a backward hidden state, $\overleftarrow{\mathbf{h}_i}$. For the backward layer, the equations are the same, but you simply process the input sequence in the reverse order. You then concatenate the representations from each direction to form a single hidden state for each input token, $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}, \overleftarrow{\mathbf{h}_i}]$. However, in this example, we're only using a uni-directional encoder.

The attention mechanism is given by

$$a(\mathbf{s}_{t-1}, \mathbf{h}_j) = \mathbf{v}^\top \tanh(W_{a,s}\mathbf{s}_{t-1} + W_{a,h}\mathbf{h}_j) \tag{4}$$

$$\xi_{t,j} = \frac{e^{a(\mathbf{s}_{t-1}, \mathbf{h}_j)}}{\sum_{k=1}^{T_x} e^{a(\mathbf{s}_{t-1}, \mathbf{h}_k)}} \tag{5}$$

$$\mathbf{c}_t = \sum_{j=1}^{T_x} \xi_{t,j}\mathbf{h}_j \tag{6}$$

where $\mathbf{v}$ and both W are leaned parameters and $a(\cdot)$ is an alignment model, which is a single-layer multilayered perceptron (feed forward network) in this case.

The general process of the model is, first, the encoder generates a hidden state vector for each word in the input sequence using Equation 1, which yields $(\mathbf{h}_1, \ldots, \mathbf{h}_{T_x})$. Next, we begin decoding. For the very first step of the decoder, the previous word embedding is that of a special `START` token and the previous hidden state is an initial state, $\mathbf{s}_0$, which is typically either a vector of 0's or the last hidden state of the encoder, $\mathbf{h}_{T_x}$. At each step of the decoder, the decoder applies the attention mechanism to obtain context vector $\mathbf{c}_t$. For the attention mechanism, the alignment model is applied to each encoder hidden state indiviudally to obtain a score for each encoder hidden state (Equation 4). Then, a softmax is applied to the scores for each encoder hidden state to normalize them (Equation 5). The context vector, $\mathbf{c}_t$, is then computed as the weighted average of the encoder hidden states (Equation 6). With the previous decoder hidden state $\mathbf{s}_{t-1}$, context vector $\mathbf{c}_t$, and the word embedding of the previously generated word $E\mathbf{y}_{t-1}$, the current decoder hidden state, $\mathbf{s}_t$, is computed using Equation 2. Finally, to generate the next word, an output layer (Equation 3) is used to produce a V-dimensional vector representing a probability distribution over the output vocabulary and a word, y_t, is selected by sampling from this distribution. The output layer can use a combination of $\mathbf{s}_t$, $E\mathbf{y}_{t-1}$, and/or $\mathbf{c}_t$ as input to produce this distribution. This process continues until the next generated word is a special `STOP` token or a maximum length is reached (or any other stopping criteria). After this stopping criteria is met, we take the sequence of words prior to this token as the generated sequence. For training, the generated sequence is compared against the ground truth sequence to compute the loss. The entire model is end-to-end differentiable so there are no special steps for training the encoder-decoder with an attention mechansim.

Using this description and the equations above, answer the following questions:

1. Given an encoder hidden state size of m, decoder hidden state size of n, and embedding size of d, what are the dimensions for the learned parameters, including embedding matrices?
   - $\hat{E} \in \mathbb{R}^{d \times V'}$
   - $E \in \mathbb{R}^{d \times V}$
   - $W_x \in \mathbb{R}^{m \times d}$
   - $W_h \in \mathbb{R}^{m \times m}$
   - $W_y \in \mathbb{R}^{n \times d}$
   - $W_s \in \mathbb{R}^{n \times n}$
   - $W_c \in \mathbb{R}^{n \times m}$
   - $W_{a,s} \in \mathbb{R}^{n \times n}$
   - $W_{a,h} \in \mathbb{R}^{n \times m}$
   - $\mathbf{v} \in \mathbb{R}^{n}$

2. Does each part of the additive attention mechanism's alignment model (or scoring function) need to be recomputed at each decoder step? Justify your answer. (*This should be short.*)
   - No, the $W_{a,h}\mathbf{h}_j$ portion can be computed after the encoder hidden states are calculated. This is because that portion of the equation does not depend on t.

3. Assume that all the weight matrices, W, are equal to the identity matrix and that $\mathbf{v}$ is a vector of all 1's. Given $\mathbf{s}_{t-1} = (0.5, 0.5)^\top$, $\mathbf{h}_1 = (0.2, 0.2)^\top$, $\mathbf{h}_2 = (0.8, 0.8)^\top$, and $E\mathbf{y}_{t-1} = (0.6, 0.4)^\top$, find $\mathbf{s}_t$. This corresponds to one decoder step and without the output layer. Note, $T_x = 2$.
   - $\mathbf{s}_t = (0.93, 0.90)^\top$

P.S.: Here are links to two important papers about the attention mechanism:

- Bahdanau et al., 2015: `https://arxiv.org/pdf/1409.0473.pdf` **check out the appendix for detailed equations.
- Luong et al., 2015: `http://aclweb.org/anthology/D15-1166`