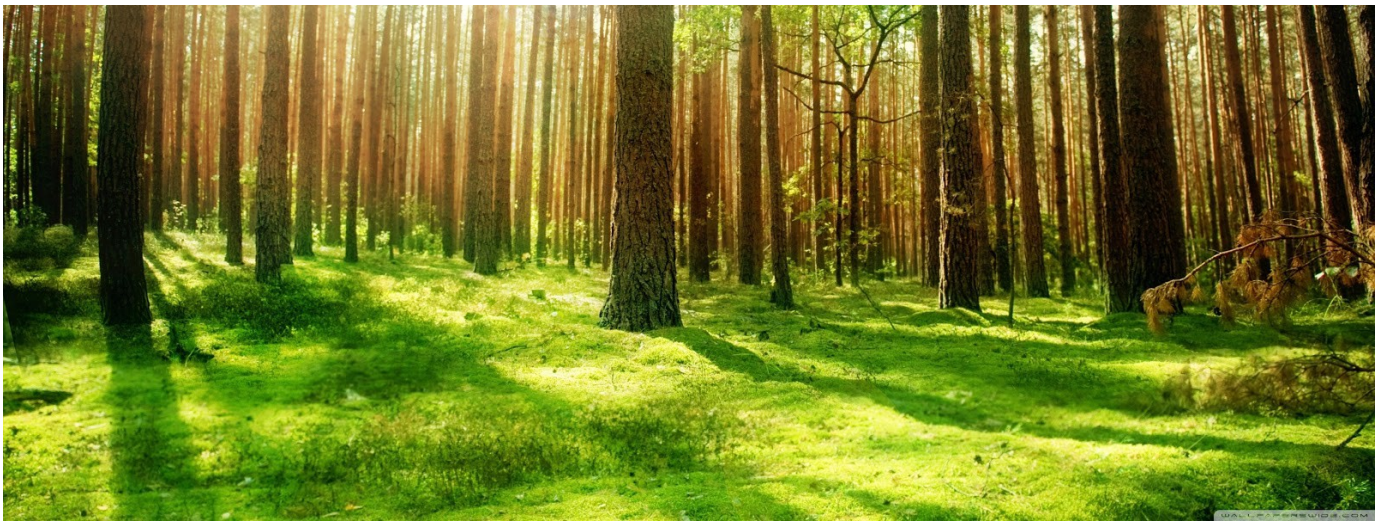
Published in Towards Data Science · Following

Abhishek Sharma · Follow
Mar 10, 2019 · 8 min read

Attention-based Neural Machine Translation



Attention mechanisms are being increasingly used to improve the performance of Neural Machine Translation (NMT) by selectively focusing on sub-parts of the sentence during translation. In this post, we will cover 2 simple types of attention mechanism: ***A global approach*** (which attends to all source words) and ***A local approach*** (which only looks at a subset of source words). Do keep it in mind that source refers to encoder and target refer to the decoder.

This blog will cover ***this*** paper which proved that adding attention could result in significant performance gain over non-attention based networks. The ensemble model proposed in the above-mentioned paper yielded a new state of the art model for WMT'15 English to German translation.

Apart from improving the performance on machine translation exercises, attention-based networks allow models to learn alignments between different modalities (different data types) for e.g. between speech frames and text or between visual features of a picture and its text description.

Neural Machine Translation (NMT)

NMT is a large neural network that is trained in an end to end fashion for translating one language into another. The figure below is an illustration of NMT with an RNN based encoder-decoder architecture.

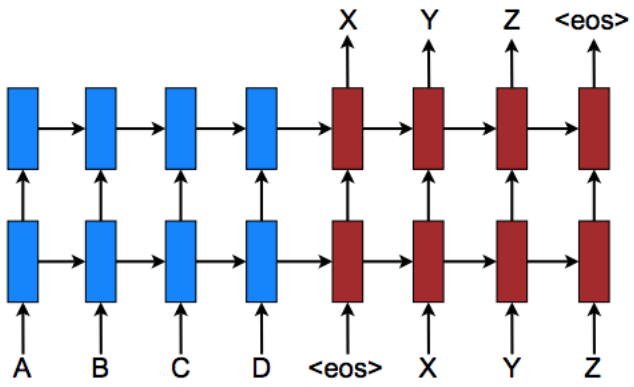


Figure 1 : Neural machine translation as a stacking recurrent architecture for translating a source sequence A B C D

NMT consist of two components:

1. An encoder which computes a representation ***S*** for each source sentence
2. A decoder which generates translation one word at a time and hence decomposes the conditional probability as:

$$\log p(y|x) = \sum_{j=1}^{m} \log p\left(y_j|y_{<j}, \boldsymbol{s}\right)$$

A probability of translation y given the source sentence x

One could parametrize the probability of decoding each word y(j) as

$$p\left(y_j|y_{<j}, \boldsymbol{s}\right) = \text{softmax}\left(g\left(\boldsymbol{h}_j\right)\right)$$

where ***h(j)*** could be modeled as

$$\boldsymbol{h}_j = f(\boldsymbol{h}_{j-1}, \boldsymbol{s}),$$

RNN hidden unit definition (h)

where
g: a transformative function that outputs a vocabulary size vector
h: RNN hidden unit
f: computes the current hidden state given the previously hidden state.

The ***training objective*** for the translation process could be framed as

$$J_t = \sum_{(x,y)\in\mathbb{D}} -\log p(y|x)$$

Loss Function

What makes NMT so popular?

1. NMT has achieved the state of the art performances in large scale translation tasks like English to French/German.
2. NMT requires minimal domain knowledge and is conceptually very simple.
3. NMT has a small memory footprint as it does not store gigantic phase tables and language models.
4. NMT has the ability to generalize well to very long word sentences.

Difference between attention and non-attention based networks

In most of the non-attention based RNN architecture source representation, ***S*** is used only once to initialize the decoder hidden state. [In Figure 1 the decoder has access only to the last layer of the encoder]

On the other hand, attention-based networks refer to a **set of source hidden states** S throughout the translation process. [In Figure 2the decoder has access to all the hidden states of the encoder]

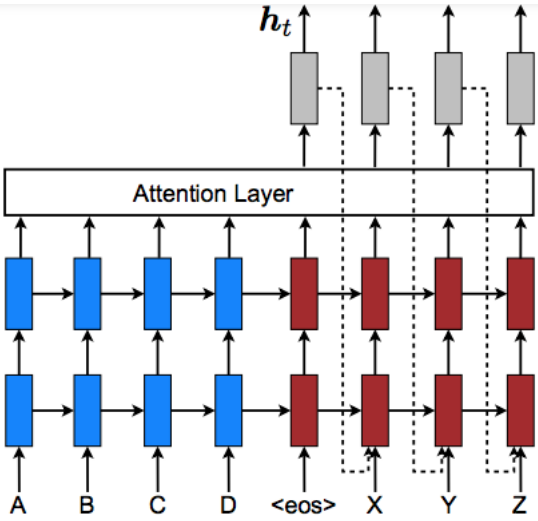


Figure 2: NMT with attention and input-feeding approach

The above figure outlines an RNN based encoder-decoder architecture with attention. As we have explained earlier attention could be broadly differentiated into 2 types:

1. **Global Attention**: Attention is placed on all source positions.
2. **Local Attention**: Attention is placed only on a few source positions.

Both attention based models differ from the normal encoder-decoder architecture only in the decoding phase. These attention based methods differ in the way that they compute context vector (***c(t)***).

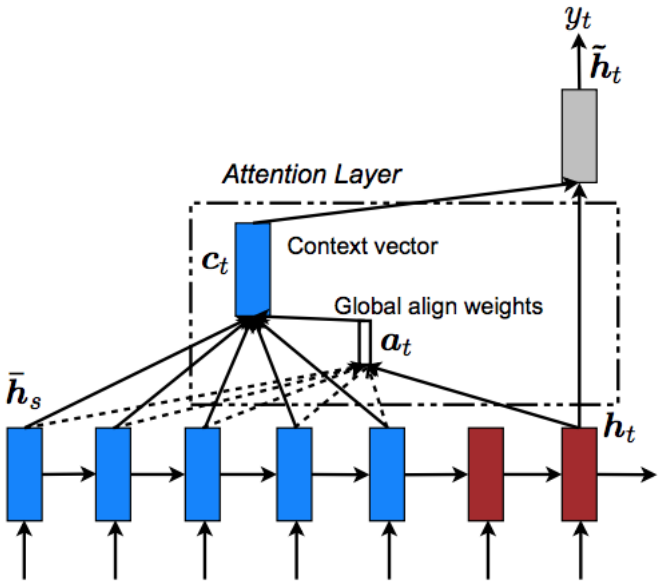


Figure 3: Hidden state of NMT architecture with global attention

Glossary for figure 3 is as follows

h(t): Hidden target state
c(t): Source side context vector
y(t): Current target word
h_bar(t): Attentional hidden state
a(t): Alignment vector

1. Both approaches first take as input the hidden state ***h(t)*** at the top layer of a stacking LSTM. (The brown cell / The target state of the decoder)
2. Derive ***c(t)*** to capture relevant source side information to help predict ***y(t)*** (Top blue cell). ***c(t)*** is basically the context that you have built for every word depending upon its alignment weights and hidden state of encoders.
3. Compute ***h_bar(t)*** from a simple concatenation of h(t) and c(t) (Top grey cell).

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

In contrast to the non-attention based architectures where only the final output of the encoder is provided to the decoder, ***h_bar(t)*** has access to all the states of hidden states of the encoder which provides an informative view of the source sentence.

4. The attentional vector is transformed using the softmax layer to produce the predictive distribution. We are using the softmax layer as we have to found the most probable word from all the available words in our vocabulary.

$$p(y_t|y_{<t}, x) = \text{softmax}(\boldsymbol{W_s}\tilde{\boldsymbol{h}}_t)$$

The above para explains a barebone architecture of the attention based networks. In the following para, we will understand how is the context vector ***c(t)*** calculated differently in local and global attention and what are the repercussions of it.

1. **Global Attention**

Global attention takes into consideration all encoder hidden states to derive the context vector (c(t)). In order to calculate ***c(t),*** we compute ***a(t)*** which is a variable length alignment vector. The alignment vector is derived by computing a similarity measure between ***h(t)*** and ***h_bar(s)*** where ***h(t)*** is the source hidden state while ***h_bar(s)*** is the target hidden state. Similar states in encoder and decoder are actually referring to the same meaning.

Alignment Vector (a(t))

The alignment vector (***a(t,s)***)is defined as

$$\begin{aligned}\boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}\end{aligned}$$

The score is a content-based function for which any of the following alternatives could have been used:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

The score function

Through score function, we are trying to calculate the similarity between the hidden states of the target and the source. Intuitively similar states in hidden and source refer to the same meaning but in different languages.

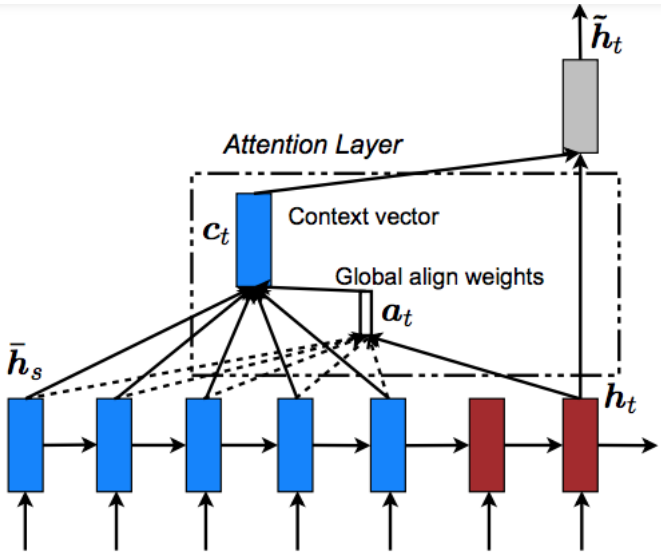


Figure 4: Global attentional model: At each time step t, the model infers a variable-length alignment weight vector a(t) based on the current target state h(t) and all source states h_bar(s). A global context vector,c(t) is then computed as the weighted average, according to a(t), over all the source states.

The connecting lines in Figure 4 represent the interdependent variables.

For example

1. *a(t)* is dependent on h(t) and h_bar(s)
2. *c(t)* is dependent on a(t) and h_bar(s)
3. *h_bar(t)* is dependent on c(t) and h(t)

2. Local Attention

As Global attention focus on all source side words for all target words, it is computationally very expensive and is impractical when translating for long sentences. To overcome this deficiency local attention chooses to focus only on a small subset of the hidden states of the encoder per target word.

Local attention has the following steps which are different from what is present in global attention:

1. The model first generates an aligned position *p(t)* for each target word at time t. In contrast to the global attention model where we assume monotonic alignment, we learn aligned positions in local attention. In other words apart from learning translations you also learn if the order of translation is different from the source sentence (word 1 of the source could be word 4 in the translated sentence, hence we need to calculate this otherwise our similarity score will be all wrong as our attention will be focussed on a word in the source sentence which is not related to word 1 of source sentence).
2. The context vector (c(t)) is derived as a weighted average over the set of source hidden states within the window [p(t) — D, p(t)+D]; D is empirically selected. As compared to the global alignment vector local alignment vector *a(t) is* now fixed dimensional.

Until now we have assumed that both the translated and source sentence are monotonically aligned. On the basis of this, we have a further differentiation of the local attention which is as follows :

1. **Monotonic Alignment(*local-m*)**

Set p(t)=t, which means that we are assuming that source and target sequences are roughly monotonically aligned. Alignment vector is the same as the global alignment

$$= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

Local alignment is the same as that of global alignment

2. Predictive alignment (***local-p***)

Instead of assuming monotonic alignments, our model predicts an aligned position as follows:

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W_p h}_t)),$$

Alignment Position for **local-p** model

W(p) and **v(p)** are models parameters which will be learned to predict positions.
S is the source sentence length
p(t): [0,S]

To favor alignment position ***p(t),*** we place a Gaussian distribution centered around p(t). This gives more weight to the position ***p(t)***. We modify our alignment weights as

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

Alignment vector for **local-p** model

to capture the same.

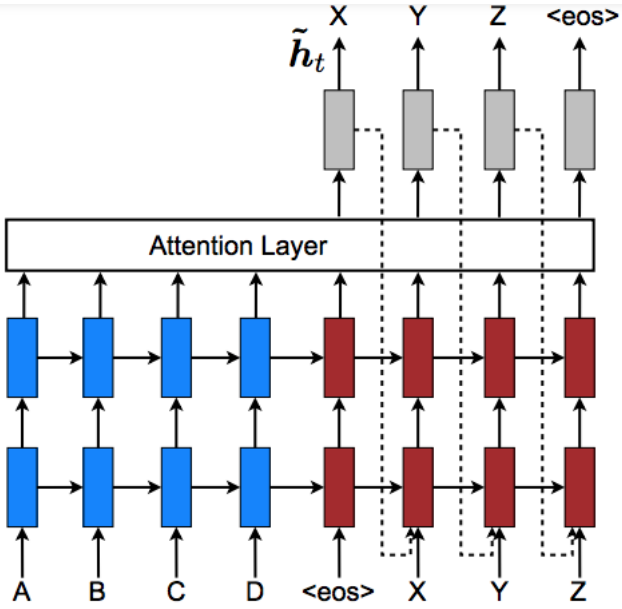
To summarise, global attention is computationally more expensive and is useless for long sentences while local attention focusses on ***D*** hidden states on both sides of ***p(t)*** to overcome this. Local attention has 2 flavors ***local-m (***the source and target alignment are assumed to be the same***)*** and ***local-p*** (where we calculate the ***p(t)).***

Input-feeding Approach

In the proposed attention mechanisms the attention decisions are made independently (previously predicted alignments does not influence next alignment) which is suboptimal. In order to make sure that future alignment decisions take into consideration past alignment information ***h_bar(t)*** is concatenated with inputs at the next time steps as illustrated.

This is done so as to:

1. make the model fully aware of the previous alignment choices.
2. create a very deep network spanning both horizontally and vertically.



Attentional Vectors h_bar(t) is fed to the next time steps to inform the model about past alignment decisions

In the above post, we have covered the basics of the attention network along with NMT.

References

1. Effective approaches to Attention-based Neural Machine Translation

Please feel free to reach me for any ideas or additions to the above post.