RNN is one of the popular language processing models, but the RNN model has limitations that make it inefficient and not widely used.

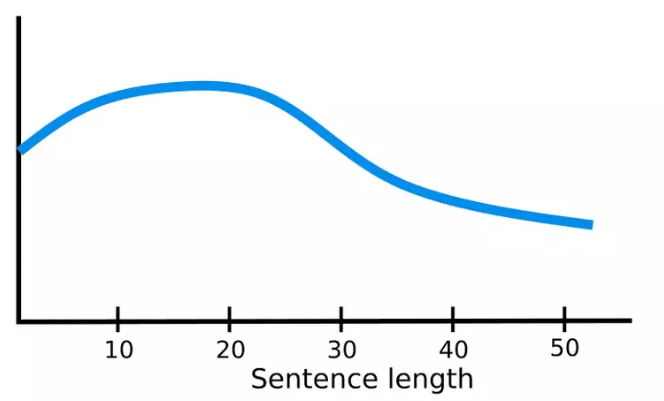
The RNN model suffers from the following limitations:

* Vanishing gradient: In short, the gradient phenomenon will be so small that it almost disappears in the last hidden states when the input is a long string like a paragraph....
* Exploding gradient: This is a phenomenon where the gradient is too large due to gradient accumulation in the last layers, which is especially common for long sentences.
* Memory Compression: Due to the compression of the input string into a fixed-size vector, experiments have shown that these models have very poor memory ability for long sentences, while wasting memory for short sentences. than. This disadvantage still exists in LSTM or GRU.
* Incompatible with structured data: For example, let's say "She is eating a green apple". Apparently "apple" has more to do with "eating" than any other word. However, the mechanism of RNN's sequential learning from left to right (inductive bias) lacks the mechanisms for the model to learn what is actually relevant. (structural bias)

To solve the above disadvantages, the LSTM and GRU models were born. But every solution will give rise to new problems and LSTM encounters the following drawbacks:

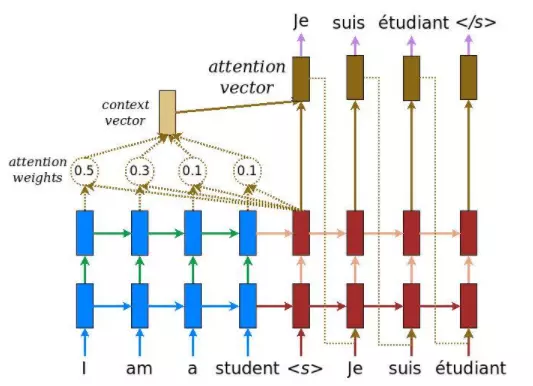
* Difficult to train, long training time due to very long gradient path (100 word sequence with gradient as 100 layer network).
* Transfer learning does not work with LSTM, which means that for a new problem, we need to retrain the model with a separate dataset for the given task (expensive).

Before going into the attention mechanism, we need to find out why this mechanism was born, more specifically in the machine translation (NMT) problem. We often use the seq2seq model with two components, the encoder and decoder blocks, with the task of creating a target string in one language from an initial string in another language. These two blocks are both made up of RNN layers. The Encoder block will process the input information and the output is a single representation vector, aka this process is information compression. This representation vector will carry all the information so that the Decoder block can generate the target sentence. In fact, the seq2seq model by architecture from RNN works very well for strings with long and short items, as the string length increases, the quality of the model will decrease significantly.



Going back to the seq2seq model with RNN, then, the encoder will have to "compress" the entire input string into a single vector - this is difficult, when the string is long enough and the encoder is forced to put all the information into one vector this single representation, it will definitely "forget" some information (bottleneck)! In addition, the decoder sees only a single input vector, although at each time-step different parts of the input sequence may be more useful than others. But for the current model, the decoder would have to extract this relevant information from a single representation vector - which is also extremely difficult.

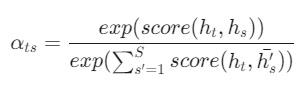
Attention was born in 2015 Bahdanau2015 with the aim of solving the above problem. With this mechanism, at each different time-step, the model will focus on different parts of the input.

Thus, the attention mechanism was born to solve the problems of the seq2seq model (transformer and attention mechanism was born to replace seq2seq without the need for feedback neural networks), with the idea of using a context vector. The scene can interact with the entire encoder's hidden state vector instead of just using the final hidden state vector to generate the representation vector for the decoder. More specifically, the seq2seq model, when applying the attention mechanism, will have the following structure (blue blocks are encoders, red blocks are decoders):

Details of the steps, at each time-step *t* on the decoder side:

Step 1: Get the decoder's hidden state vector ht and all the hidden state vectors of the encoder hs.

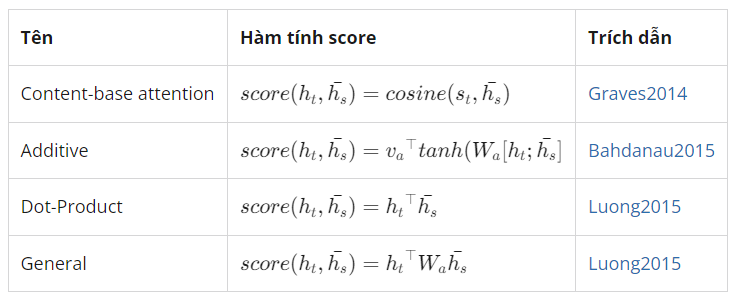
Step 2: Calculate the attention score. For each hidden state vector of the encoder, we need to calculate the score showing the relationship with the hidden state vector ht of the decoder.

Step 3: Calculate attention weight. Apply softmax function with input point attention

Step 4: Calculate the context vector ct as the sum of the attention weights multiplied by the decoder's hidden state vector at the corresponding time-step.

Types of Attention Mechanisms

The introduction of the attention mechanism eliminated the distance dependence of the input and output sequences. With attention, the field of machine translation has significantly improved. There are more and more different types of attention mechanism. Here are a few attention mechanisms with the score function:

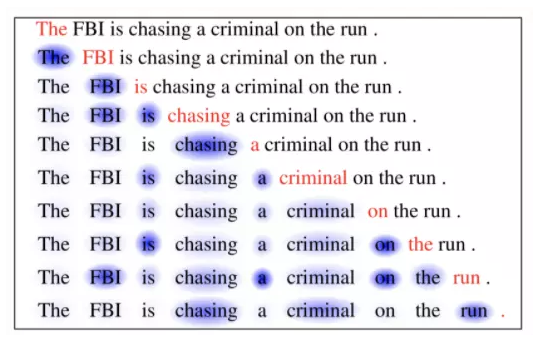


More broadly, attention mechanisms are divided into the following three categories:

* Self-attention
* Global/Soft attention
* Local/Hard attention

Self-Attention

Self-attention, also known as intra-attention, is an attention mechanism used for only one sentence. It is conceivable that we would manually create a matrix with rows and columns all being the same sentence to understand which parts of the sentence would relate to each other. This mechanism has proven effective in applications such as summarizing text, creating captions for images, machine reading, etc. Along with self-attention comes the introduction of the tranformer architecture, allowing for complete replacement. neural network architecture feeds back RNNs using fully connected models and still gives very good results. This is an important milestone for applying the attention mechanism to NLP problems. For example, in the image below, the present words (in red) and the words highlighted in green show its influence on the present word.



Soft vs Hard Attention

These two mechanisms are applied to the application to create captions for images. The first image will be processed by CNN to extract features, then the LSTM network combined with the attachment mechanism will perform the task of decoding those features to create captions. Two mechanisms of soft attention and hard attention are distinguished as follows:

* Soft attention: use the attention point as the weight to calculate the context vector and it is a differentiable function, so it is possible to use gradient decense combined with backpropagation for training. However, since the computation is over the entire input, for an image-related application the computational cost will be very expensive when the image is large.
* Hard attention: instead of averaging the weights of all hidden state vectors, it uses the attention point to select the most appropriate hidden state vector location. Hard attention is often trained using reinforcement learning. The advantage of the mechanism is its lower computational cost, but it requires more complex techniques to train.