1. Sequence-to-Sequence

The seq2seq model was born in the field of language modeling. Broadly speaking, it aims to transform an input sequence (source) to a new one (target) and both sequences can be of arbitrary lengths, e.g., machine translation between multiple languages.
The seq2seq model (see Figure 1) normally has an encoder-decoder architecture, composed of:

- An **encoder** processes the input sequence and compresses the information into a context vector (also known as sentence embedding or "thought" vector) of a fixed length. This representation is expected to be a good summary of the meaning of the whole source sequence.
- A **decoder** is initialized with the context vector to emit the transformed output. The early work only used the last state of the encoder network as the decoder initial state.

Both the encoder and decoder are recurrent neural networks, i.e. using LSTM or GRU units.

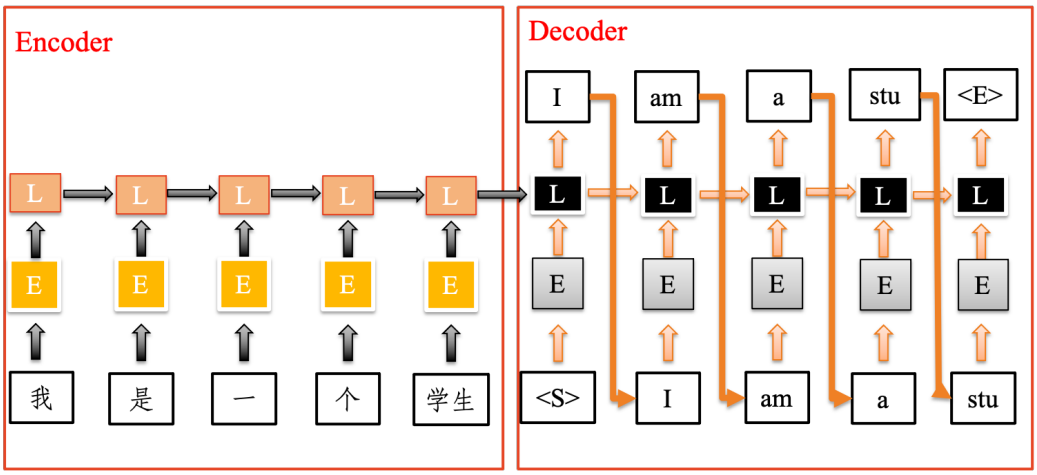

Figure 1. A Seq2Seq model for machine translation.

A critical and apparent disadvantage of this fixed-length context vector design is incapability of remembering long sentences. Often it has forgotten the first part once it completes processing the whole input. The attention mechanism was born to resolve this problem, as shown in Figure 2.

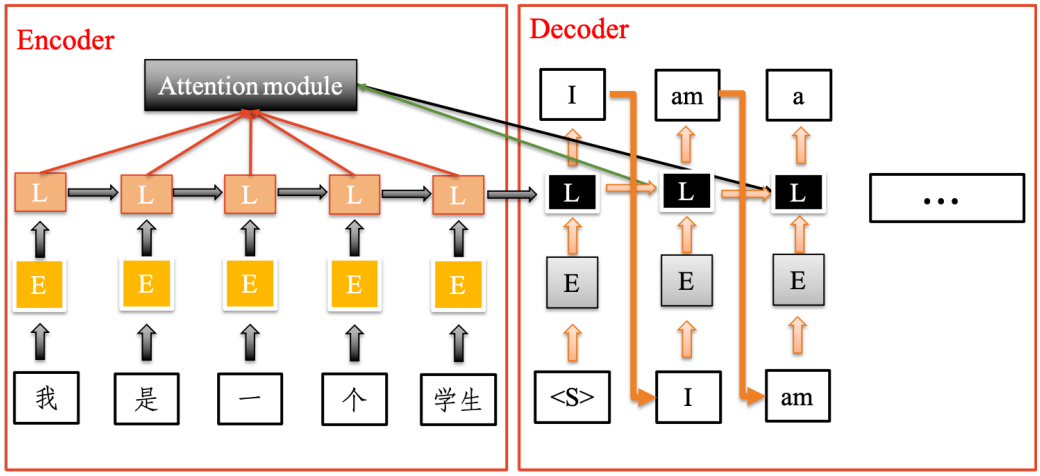

Figure 2. A Seq2Seq with an attention model for machine translation.

2. **Attention**

Now let's define the attention mechanism introduced in NMT in a scientific way. Say, we have a source sequence x of length n and try to output a target sequence y of length m

$$x = [x_1, x_2, \cdots, x_n] \tag{1}$$

$$y = [y_1, y_2, \cdots, y_m] \tag{2}$$

The encoder is a bidirectional RNN with a hidden state h_i. The decoder network has hidden state $s_t = f(s_{t-1}, y_{t-1}, c_t)$ for the output word at position t, $t = 1, \cdots, m$, where the context vector c_t is a sum of hidden states of the input sequence, weighted by alignment scores:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i \tag{3}$$

$$\alpha_{t,i} = \frac{\exp(score(s_{t-1}, h_i))}{\sum_{i'=1}^{n} \exp(score(s_{t-1}, h_i'))} \tag{4}$$

The alignment model assigns a score $\alpha_{t,i}$ to the pair of input at position i and output at position t, (y_t, x_i), based on how well they match. The set of $\{\alpha_{t,i}\}$ are weights defining how much of each source hidden state should be considered for each output. The alignment score α is parametrized by a feed-forward network with a single hidden layer and this network is jointly trained with other parts of the model. The score function is therefore in the following form, given that tanh is used as the non-linear activation function:

$$score(s_t, h_i)) = v_a^T \tanh\left(W_a[s_t; h_i]\right), \tag{5}$$

where both v_a and W_a are weight matrices to be learned in the alignment model. The matrix of alignment scores (see Figure 3) is a nice byproduct to explicitly show the correlation between source and target words.

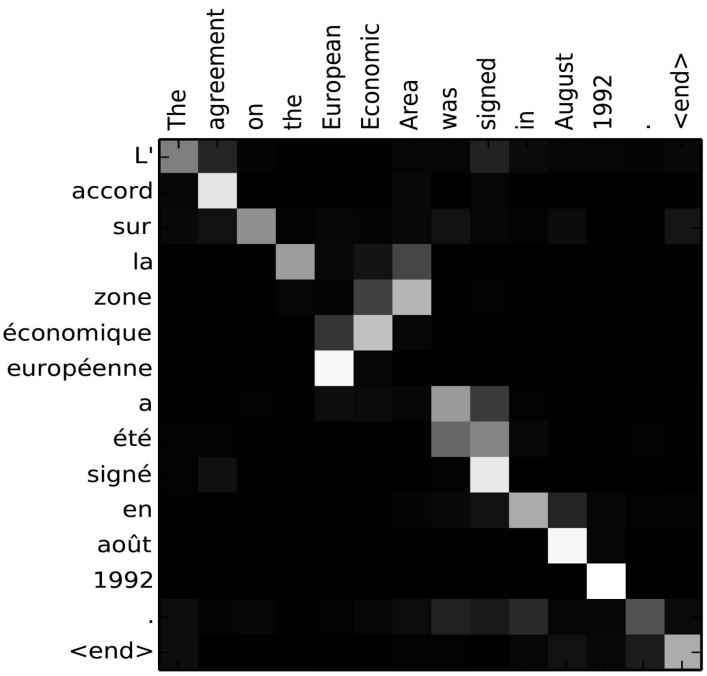

FIGURE 3. Alignment matrix of "L'accord sur l'Espace économique européen a été signé en août 1992" (French) and its English translation "The agreement on the European Economic Area was signed in August 1992".

With the help of the attention, the dependencies between source and target sequences are not restricted by the in-between distance anymore! Given the big improvement by attention in machine translation, it soon got extended into the computer vision field and people started exploring various other forms of attention mechanisms.

3. Self-Attention

Self-attention, also known as intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. It has been shown to be very useful in machine reading, abstractive summarization, or image description generation.

The "transformer" model is entirely built on the self-attention mechanisms without using sequence-aligned recurrent architecture.

3.1. **Key, Value and Query.** The major component in the transformer is the unit of multi-head self-attention mechanism. The transformer views the encoded representation of the input as a set of key-value pairs, (K, V), both of dimension n (input sequence length); in the context of NMT, both the keys and values are the encoder hidden states. In the decoder, the previous output is compressed into a query (Q of dimension m) and the next output is produced by mapping this query and the set of keys and values.
The transformer adopts the scaled dot-product attention: the output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{n}})V \tag{6}$$

The key/value/query concepts come from retrieval systems. For example, when you type a query to search for some video on Youtube, the search engine will map your query against a set of keys (video title, description, etc.) associated with candidate videos in the database, then present you the best matched videos (values).
Eq. (6) can also be understood as soft addressing, as shown in Figure 4. If each element is stored in a (K, V) in the form of a sequence, then the attention addressing is done by calculating the Q and K similarity. Q and K reflect the calculated similarity value is taken out of the V importance, i.e. the weight, and then summing the weighted values to get attention.

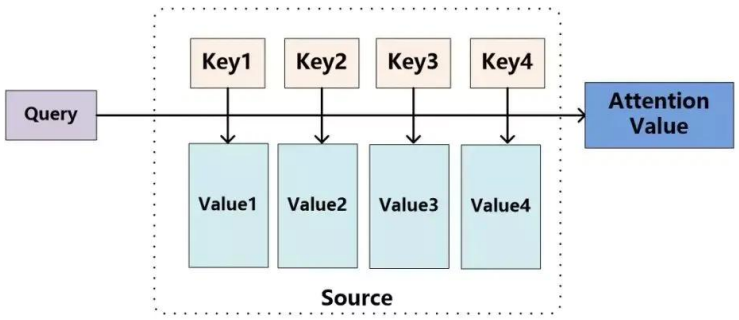

FIGURE 4. Soft addressing.

Remember that Q_i and K_i were different projections of the tokens into a n dimensional space, i.e, $Q_i = W^q a_i$ and $K_i = W^k a_i$. *Therefore, we can think about the dot product of those projections as a measure of similarity between tokens projections.* For every token projected through Q_i the dot product with the projections through K_i measures the similarity between those tokens. The

self-attention is exactly to relate different tokens!

Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel. The independent attention outputs are simply concatenated and linearly transformed into the expected dimensions. According to the paper, "multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this."