Attention, Transformers

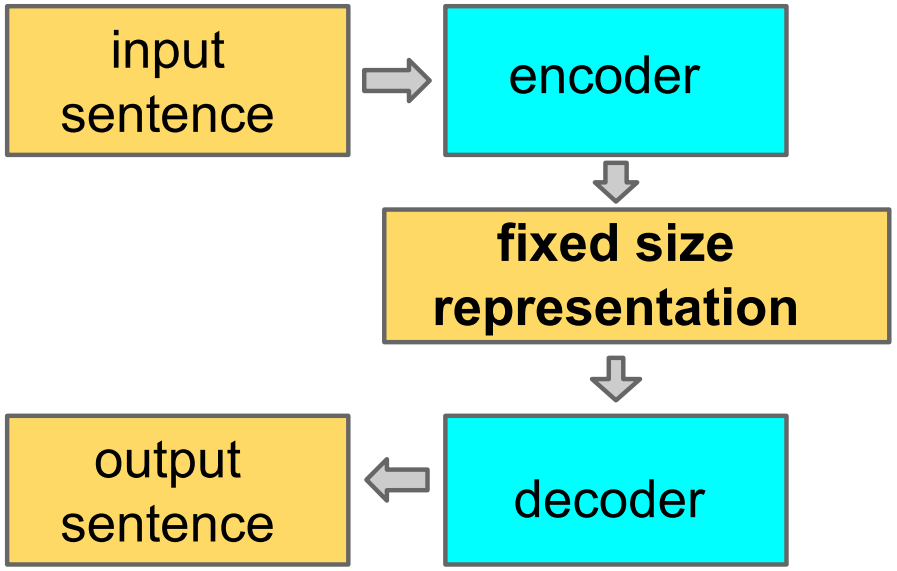
Jian Tang

HEC Montreal

Mila-Quebec AI Institute

Email: jian.tang@hec.ca

Sequence2Sequence (Encoder-Decoder)



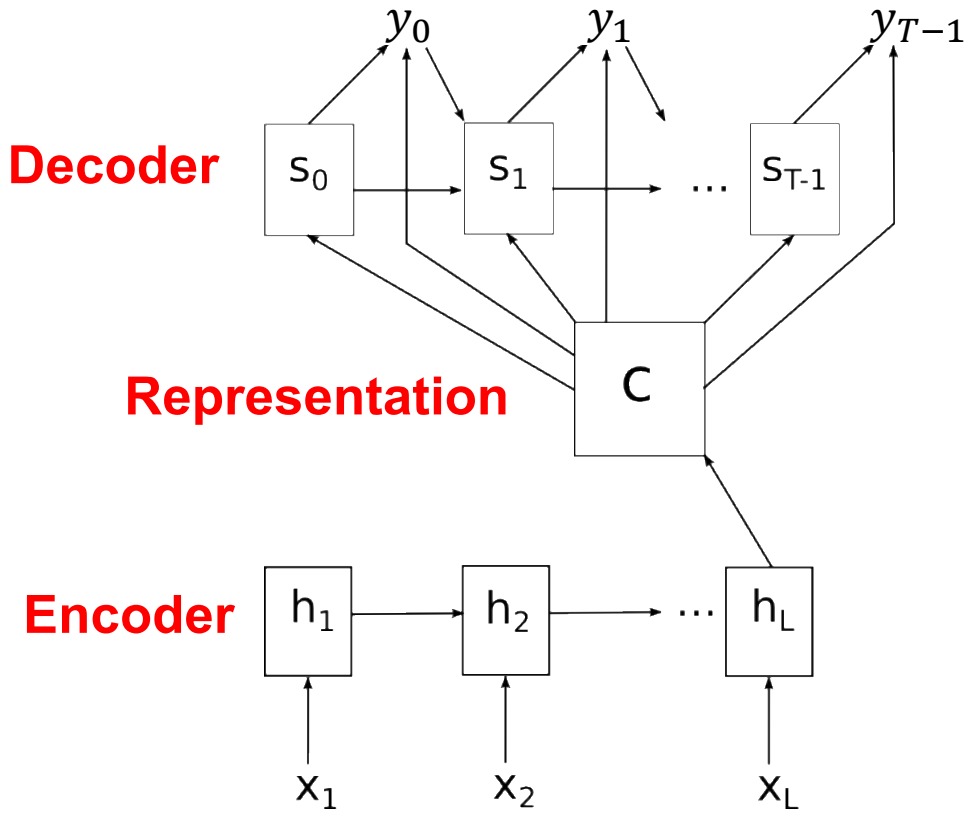
(Ñeco&Forcada, 1997)

(Kalchbrenner et al., 2013)

(Cho et al., 2014)

(Sutskever et al., 2014)

RNN Encoder-Decoder (Cho et al. 2014):



Sequence to Sequence Model

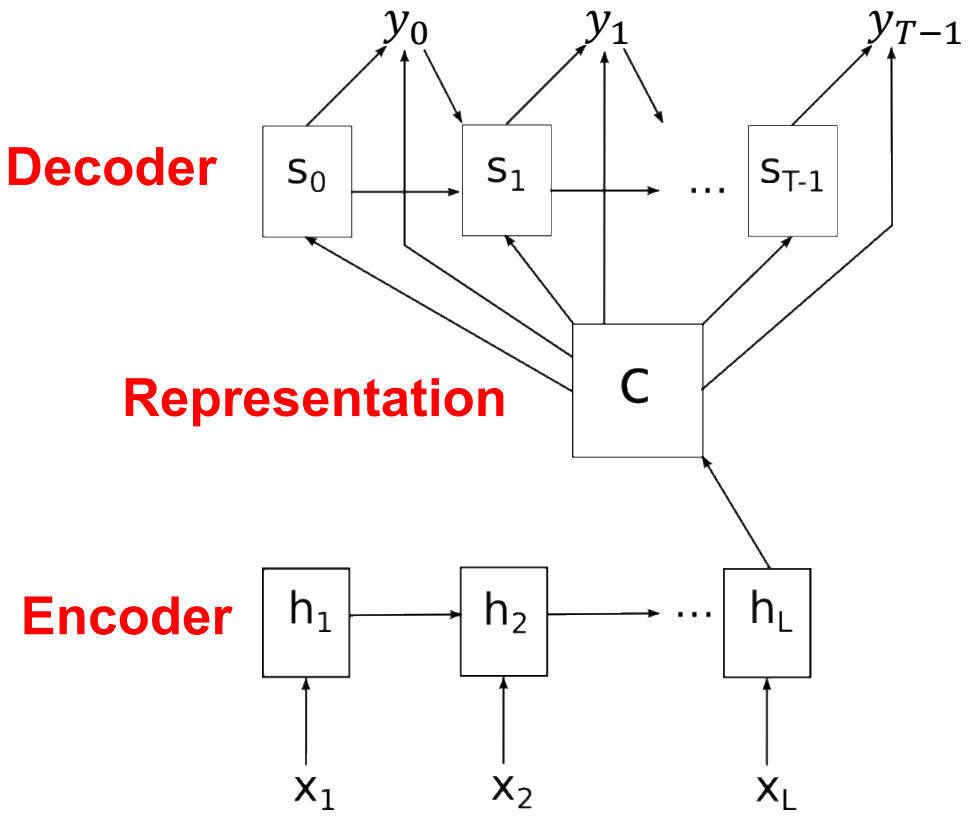
- Given an input sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$, we want to map it to an output sequence $\mathbf{y} = (y_1, y_2, \ldots, y_{T'})$. We are essentially trying to model the conditional probability

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1, y_2, \ldots, y_{T'}|x_1, x_2, \ldots, x_T)$$

Encoder: one Recurrent Neural Network

- C is the last hidden state of input sequence
  - summary of the input sequence

RNN Encoder-Decoder (Cho et al. 2014):



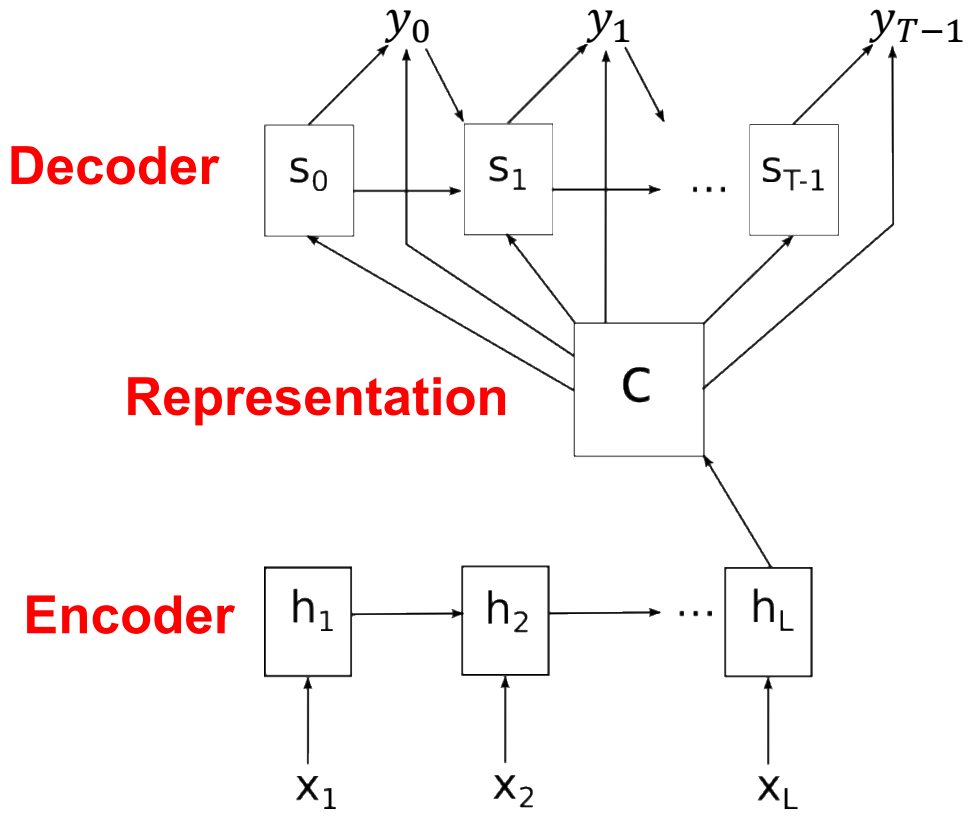
Decoder: another Recurrent Neural Network

- Decode the output sequence **y** based on
- the input sequence **x**

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1, y_2, \ldots, y_{T'}|x_1, x_2, ..$$

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1, y_2, \ldots, y_{T'}|\boldsymbol{c})$$

RNN Encoder-Decoder (Cho et al. 2014):

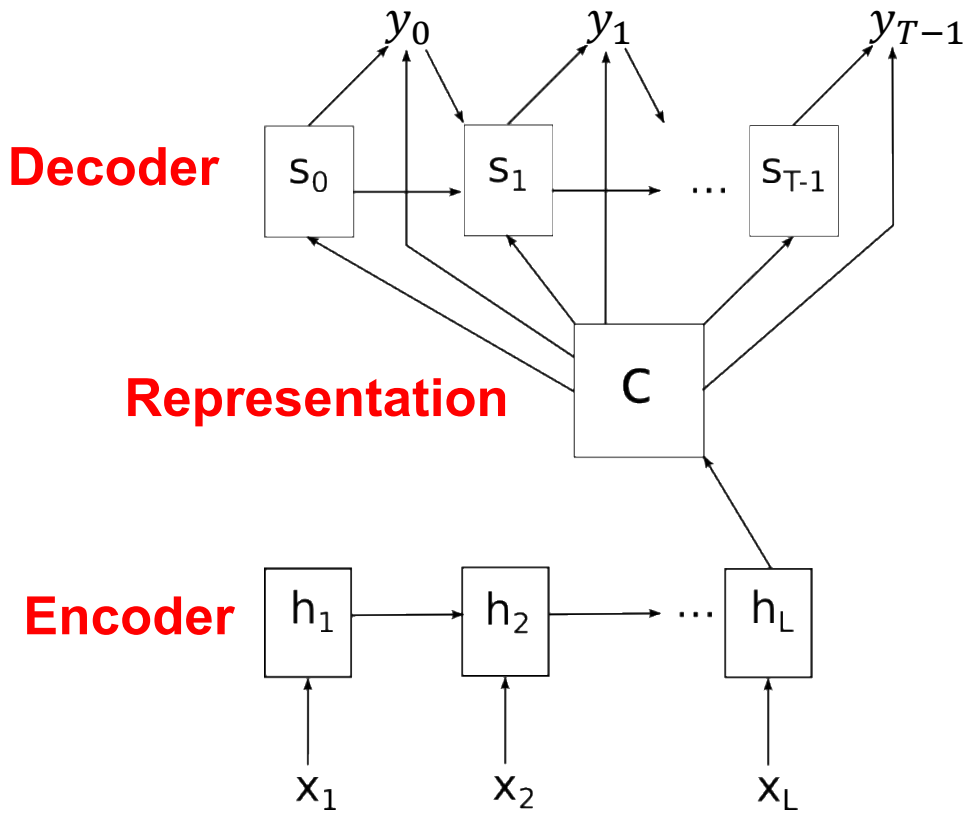


Decoder: another Recurrent Neural Network

- Decode the output sequence **y** based on **RNN Encoder-Decoder (Cho et al. 2014)**:
- the input sequence **x**

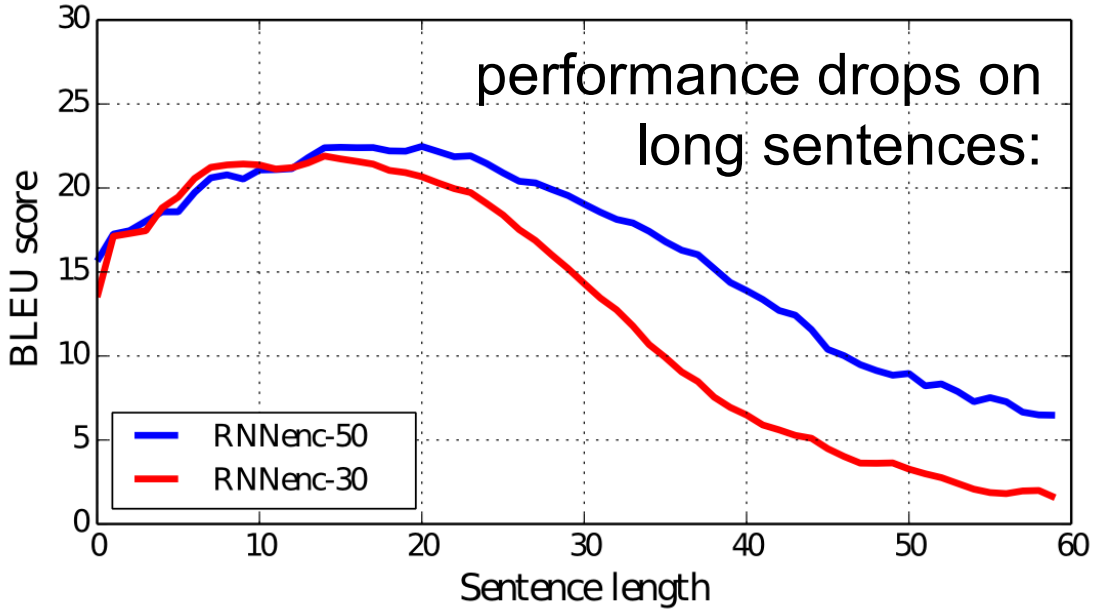
$$\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c})$$

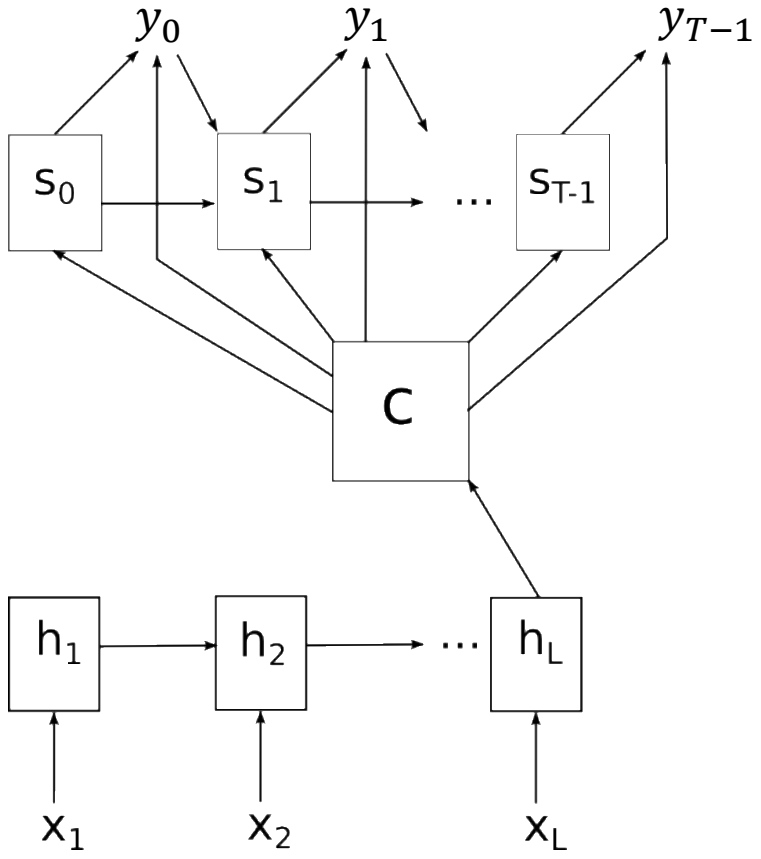
$$p(y_t | y_{t-1}, y_{t-2}, \dots, y_1, c) = g(\boldsymbol{s}_t, y_{t-1}, c)$$



RNN Encoder-Decoder Issues

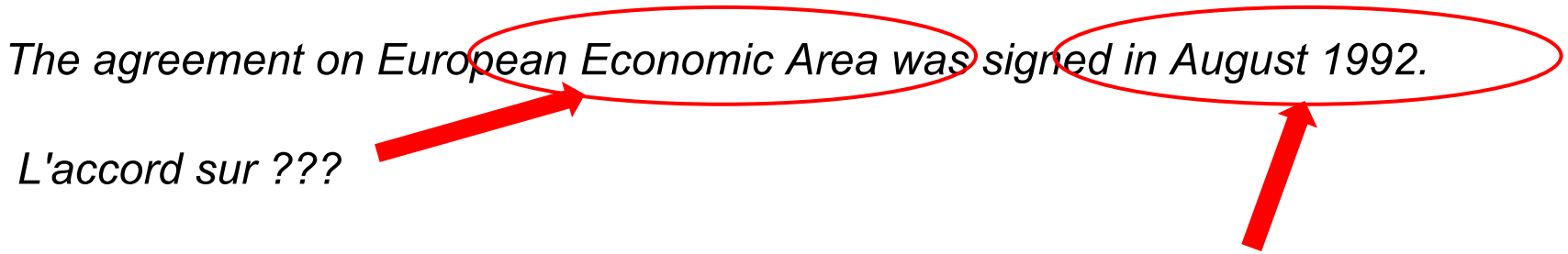
- has to remember the whole sentence
- fixed size representation can be the bottleneck
- humans do it differently





Attention-based Encoder-Decoder

Tell Decoder what is now translated:

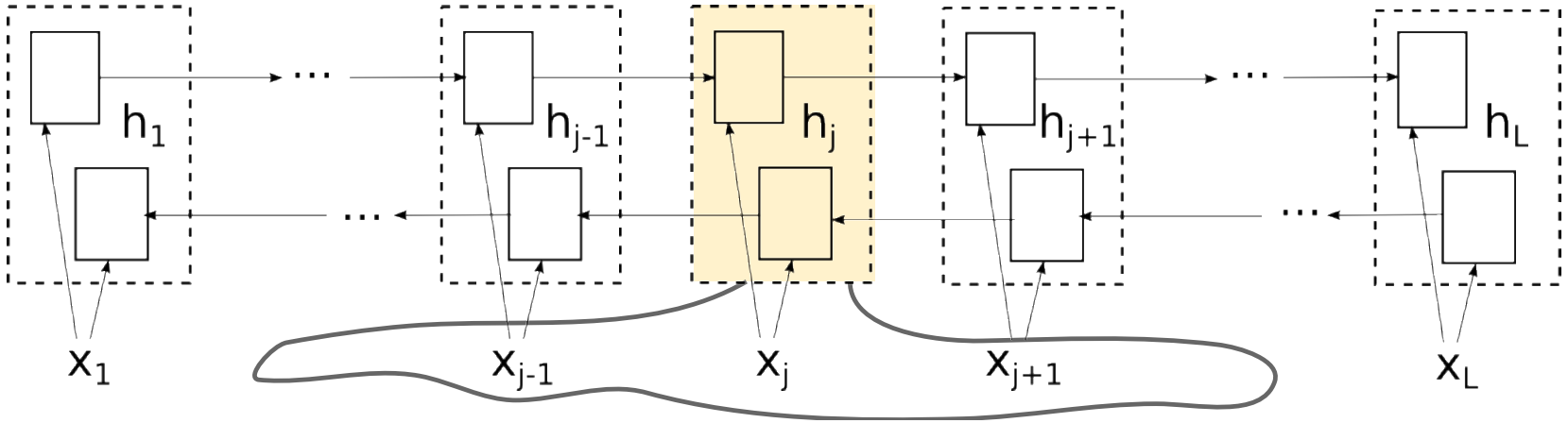


Have such hints computed by the net itself!

New Encoder

Bidirectional RNN: h_j contains x_j together with its context (..., x_{j-1}, x_{j+1}, …).

(h_1, …, h_L) is the new *variable-length* representation instead of *fixed-length* c.



New Decoder

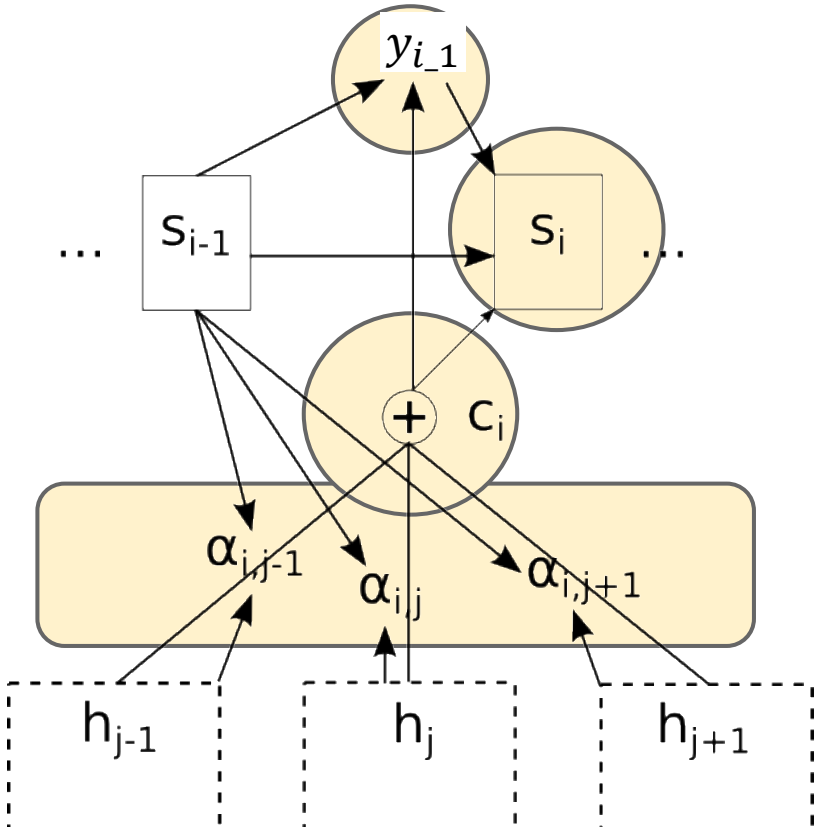
Step i:

compute alignment

compute context

generate new output

compute new decoder state



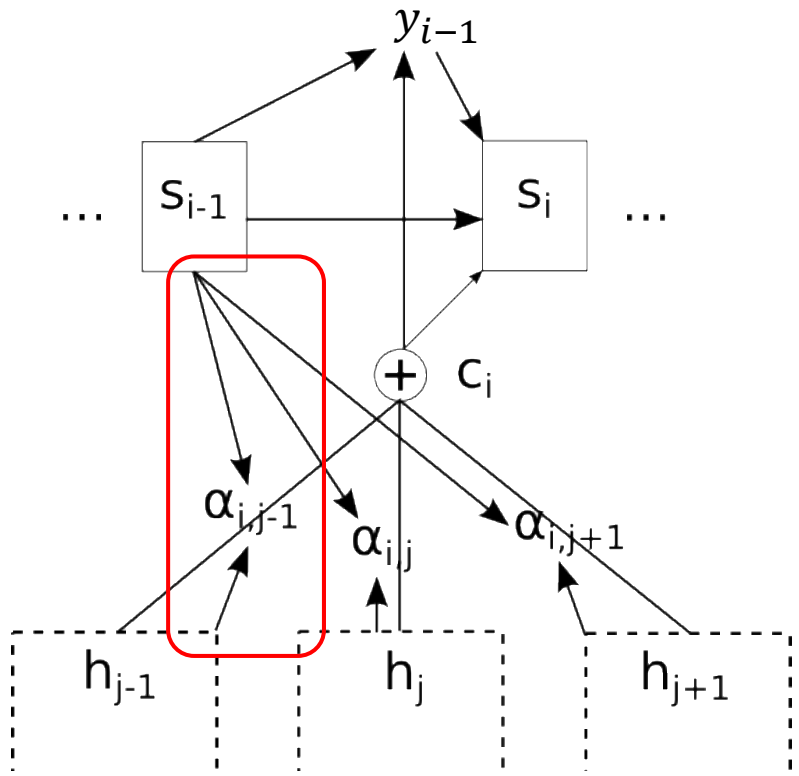
Alignment Model

$$e_{ij} = v^T \tanh(W s_{i-1} + V h_j) \quad (1)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{L} \exp(e_{ik})} \quad (2)$$

- nonlinearity (tanh) is crucial!
- simplest model possible
- Vh_j is precomputed => quadratic complexity with low constant

Calculate context: $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$



Output model

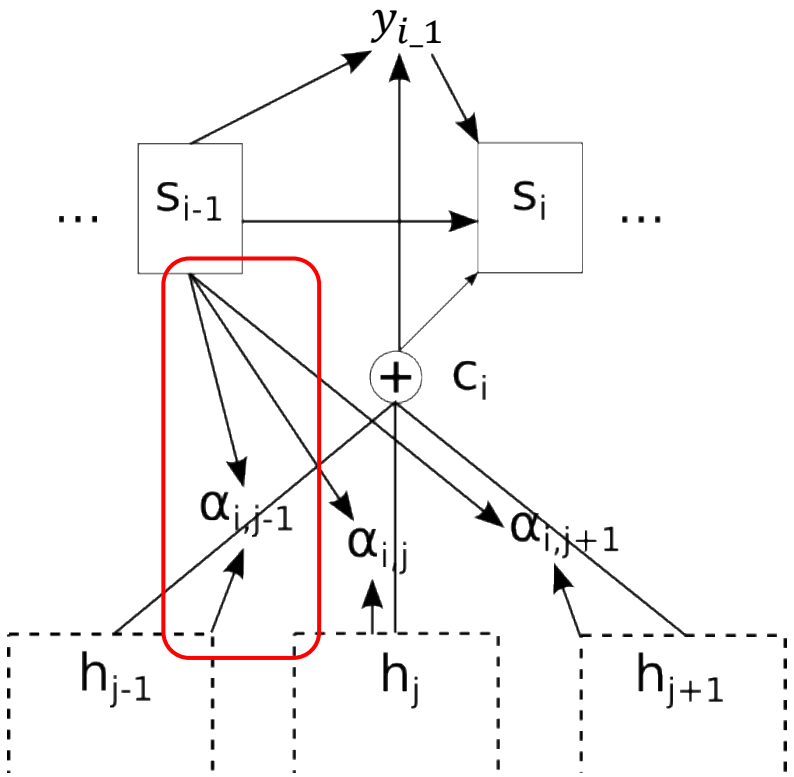
$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1},s_i,c_i),$$

Previous output

Current context

current hidden state

Architecture: Fully connected + Maxout



Update hidden state

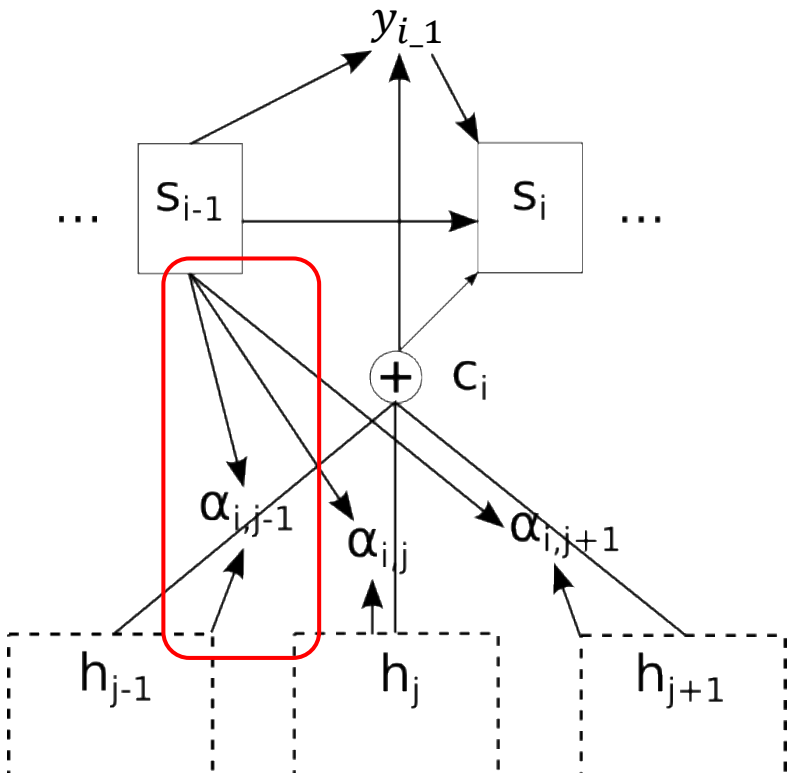
$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

Previous hidden state

Previous output

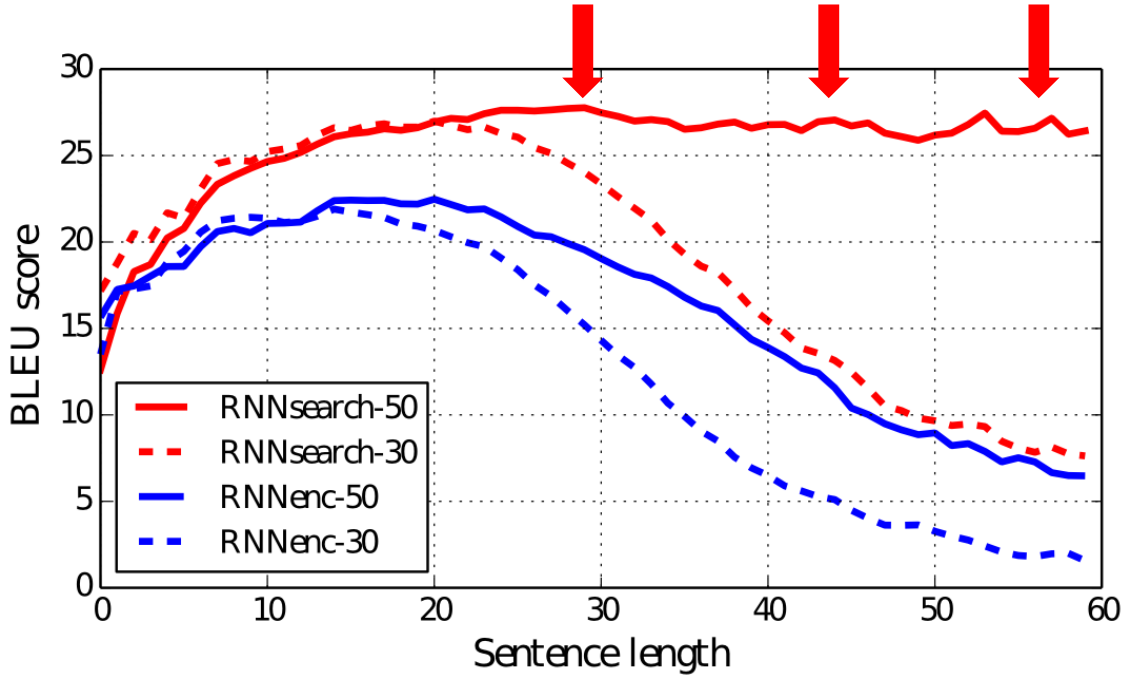
Current context

Architecture: GRU



Quantitative Results

no performance drop on long sentences

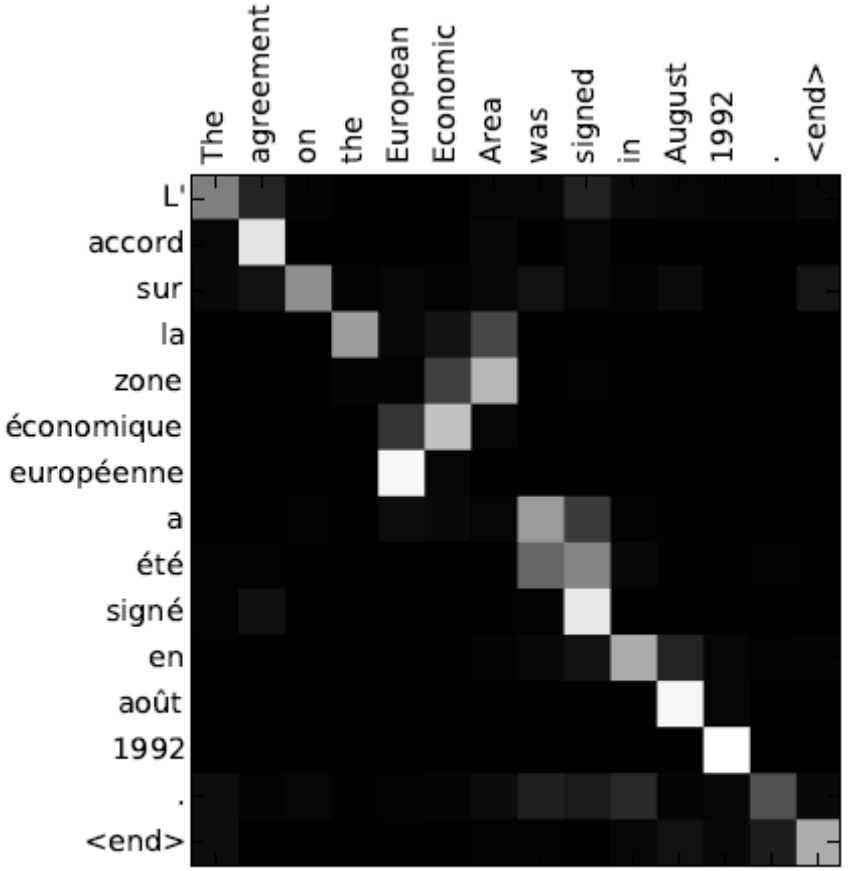


much better than RNN Encoder-Decoder

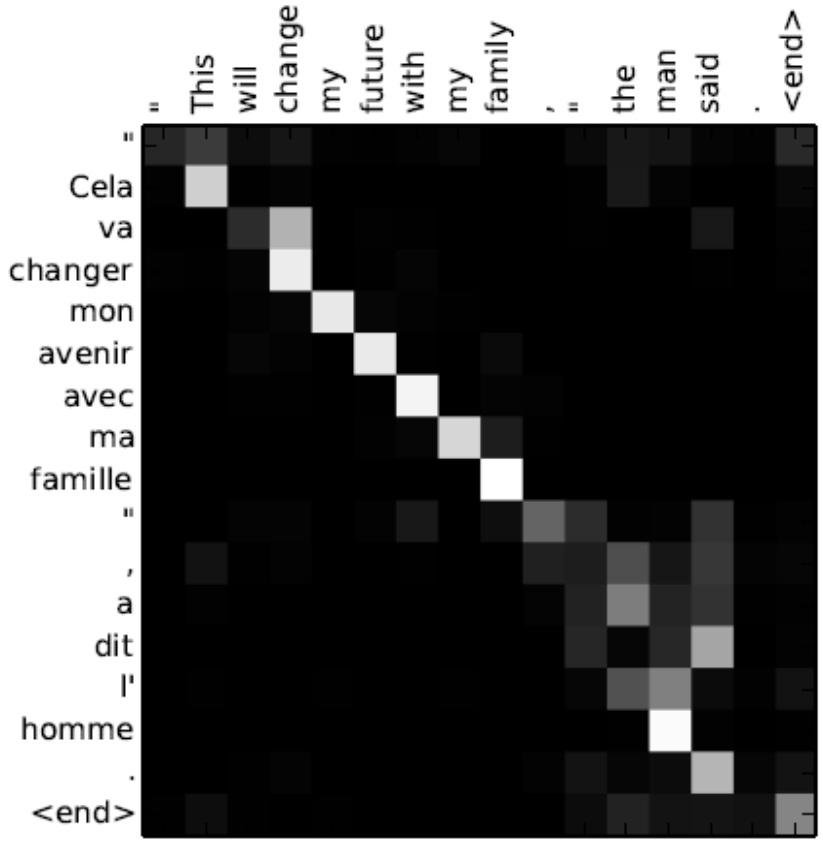
Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

without unknown words comparable with the SMT system

Alignment between the Words



(a)

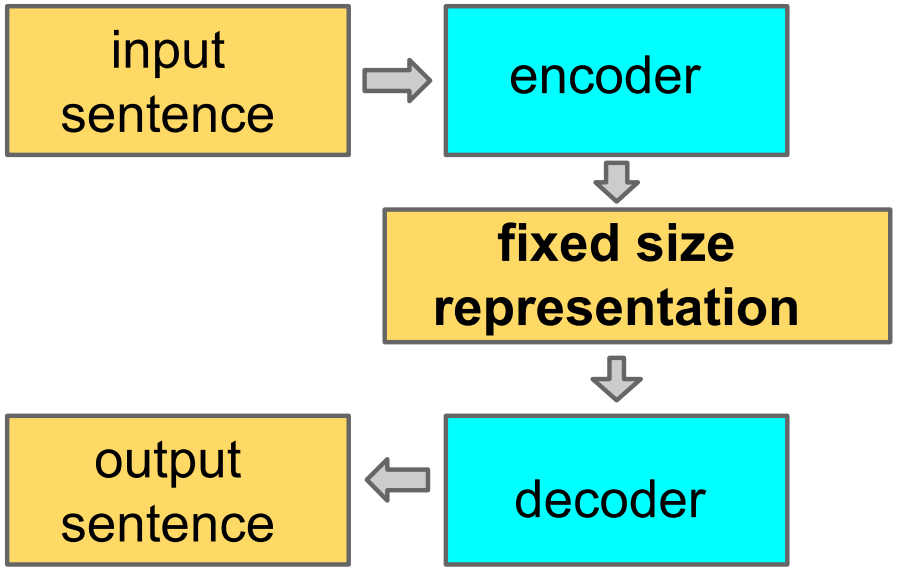


Attention Seq2Seq

- https://github.com/bentrevett/pytorch-seq2seq/blob/master/3%20-%20Neural%20Machine%20Translation%20by%20Jointly%20Learning%20to%20Align%20and%20Translate.ipynb

Attention is All You Need

Sequence2Sequence (Encoder-Decoder)



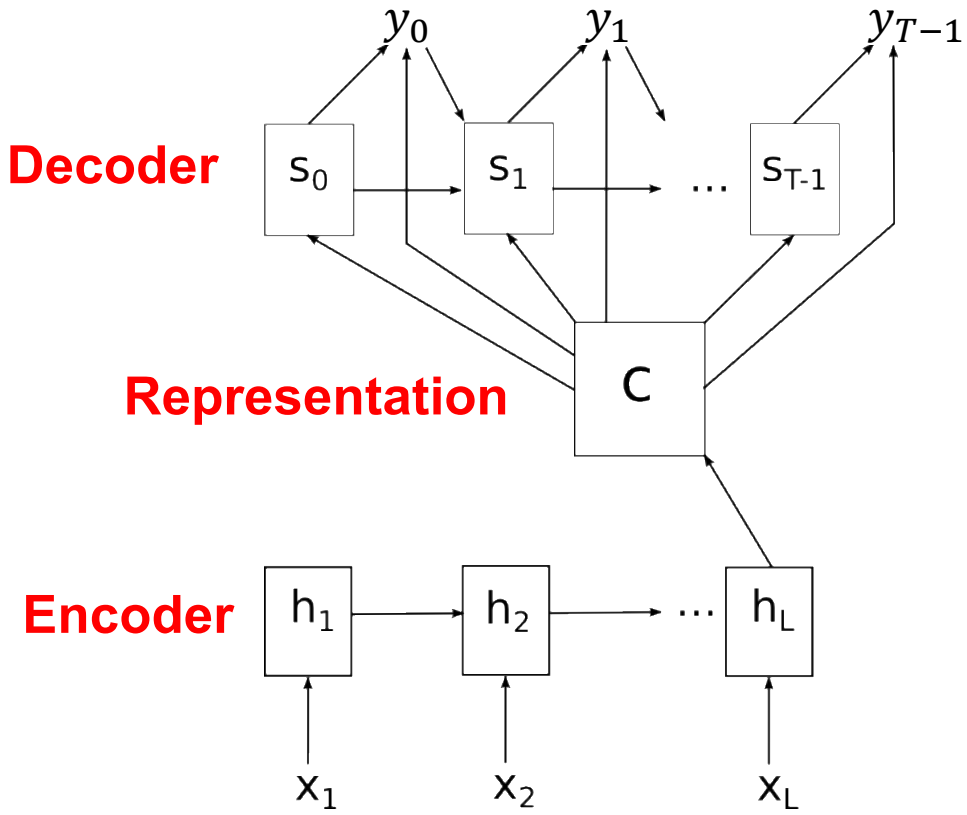
(Ñeco&Forcada, 1997)

(Kalchbrenner et al., 2013)

(Cho et al., 2014)

(Sutskever et al., 2014)

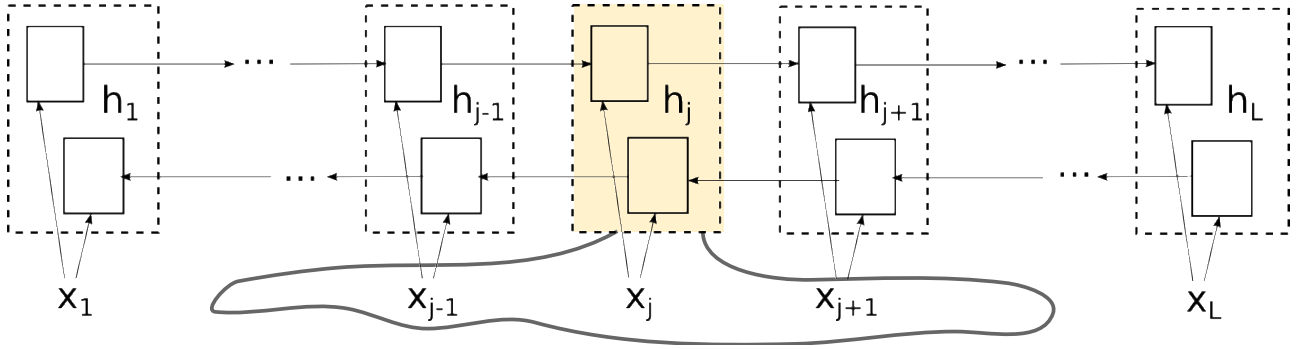
RNN Encoder-Decoder (Cho et al. 2014):



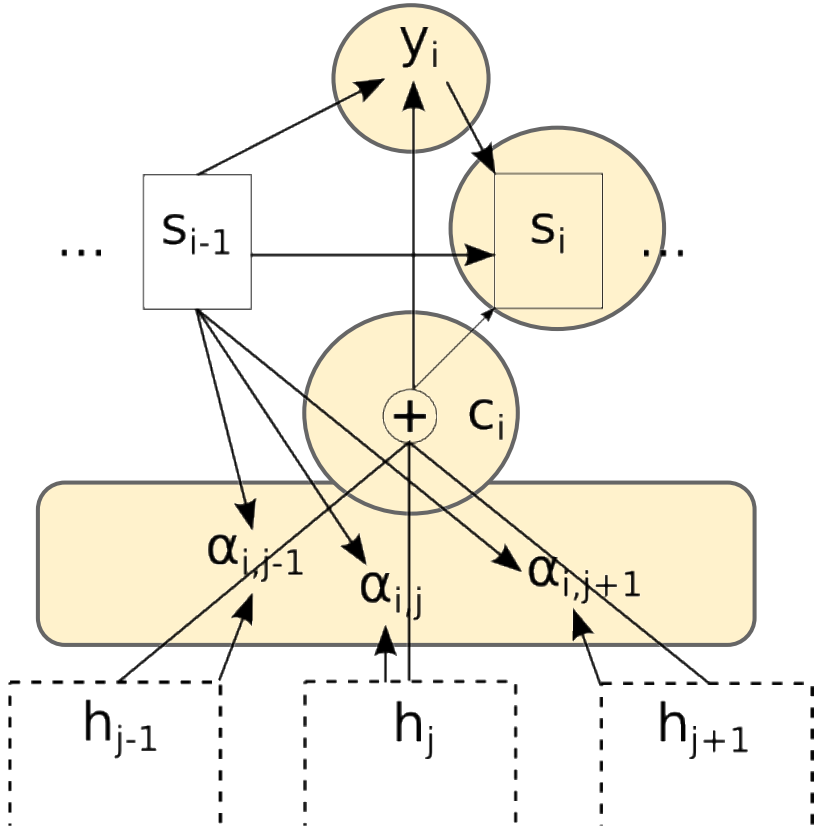
Attention-Based Encoder-Decoder

Bidirectional RNN: h_j contains x_j together with its context (..., x_{j-1}, x_{j+1}, …).

(h_1, …, h_L) is the new *variable-length* representation instead of *fixed-length* c.



Encoder



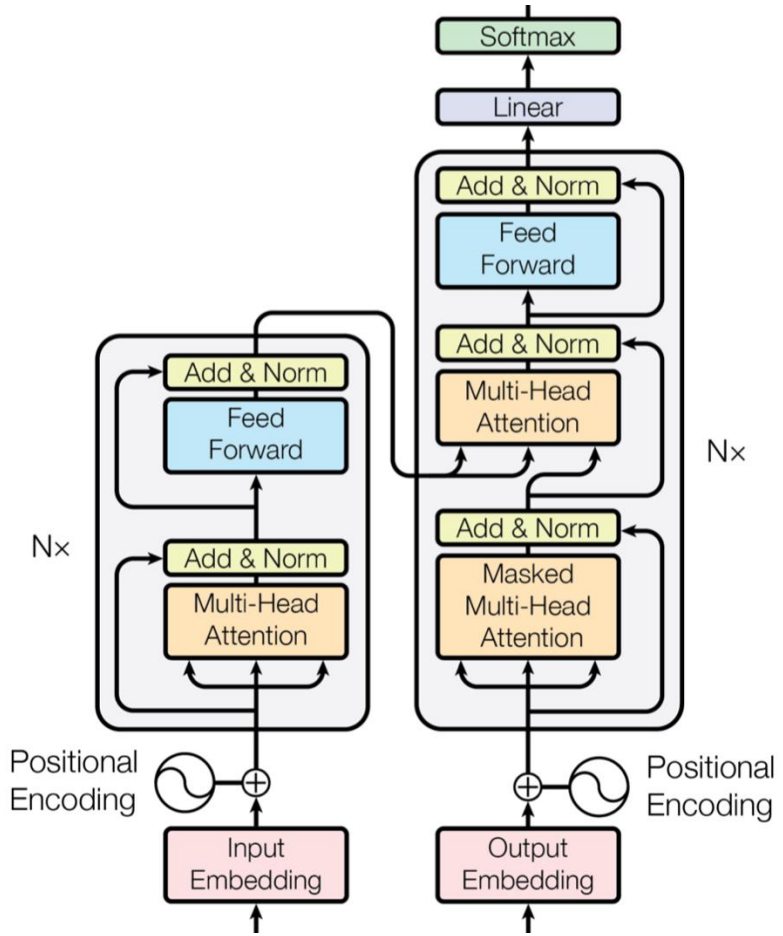
Attention-based Decoder

Limitations of Previous Seq2Seq Models

- The encoders or decoders in the sequence to sequence models are generally implemented with recurrent neural networks
  - Sequential computation
  - Non-parallel

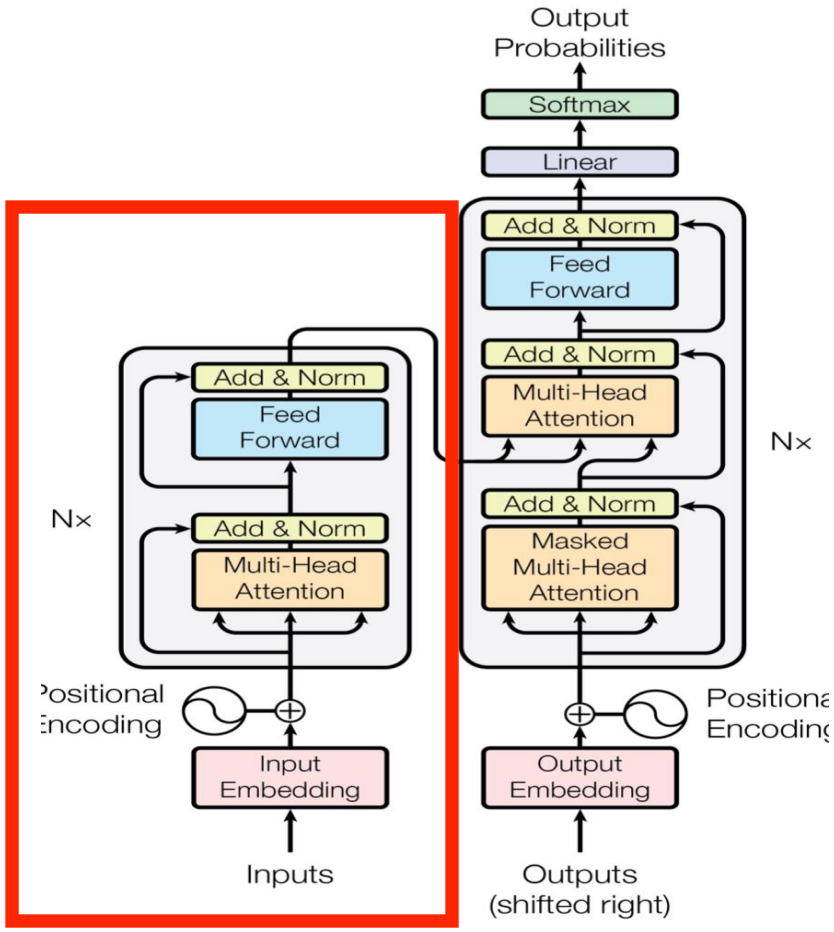
Attention is all you need (Vaswani et al. 2017)

- The Transformer (Vaswani et al. 2017)
  - Only attention is used in both encoder and decoder
  - Parallelizable



Encoder

- A stack of N=6 identical layers
- Each layer are composed of two sublayers
  - Multi-head self-attention
  - Position-wise fully connected feed-forward network
- Residual connection followed by normalization are used in both sublayers
  - $LayerNorm(x + Sublayer(x))$



Multi-head Attention

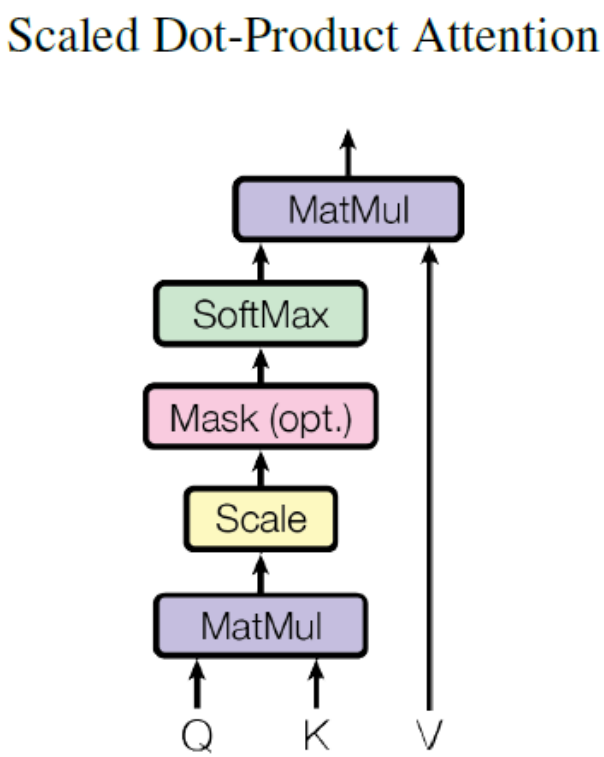
- Attention
  - Mapping a query and a set of key-value pairs to an output
  - Query, Keys, and Values are all vectors
  - The output is a weighted sum of the values, with the weights calculated according to a softmax function depending on the similarities between queries and keys

Multi-head Attention

- Scaled Dot-Product Attention
  - Avoiding pushing the softmax function into regions where it has extremely small gradients.

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

d_k: dimension of keys and queries

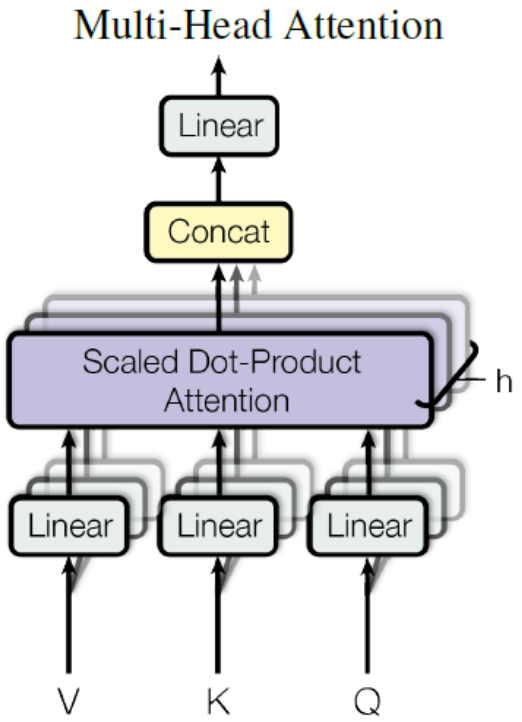


Multi-head Attention

- Multi-head Attention
  - Linearly project the queries, keys, and values h times with different, learned linear projects respectively
  - Concatenate the outputs and project again

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.



Position-wise Feed-Forward Network

- Applied to each position separately and identically
  - Two linear transformations with RELU as the activation in between
  - Different parameters are used across different layers

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Positional Encoding

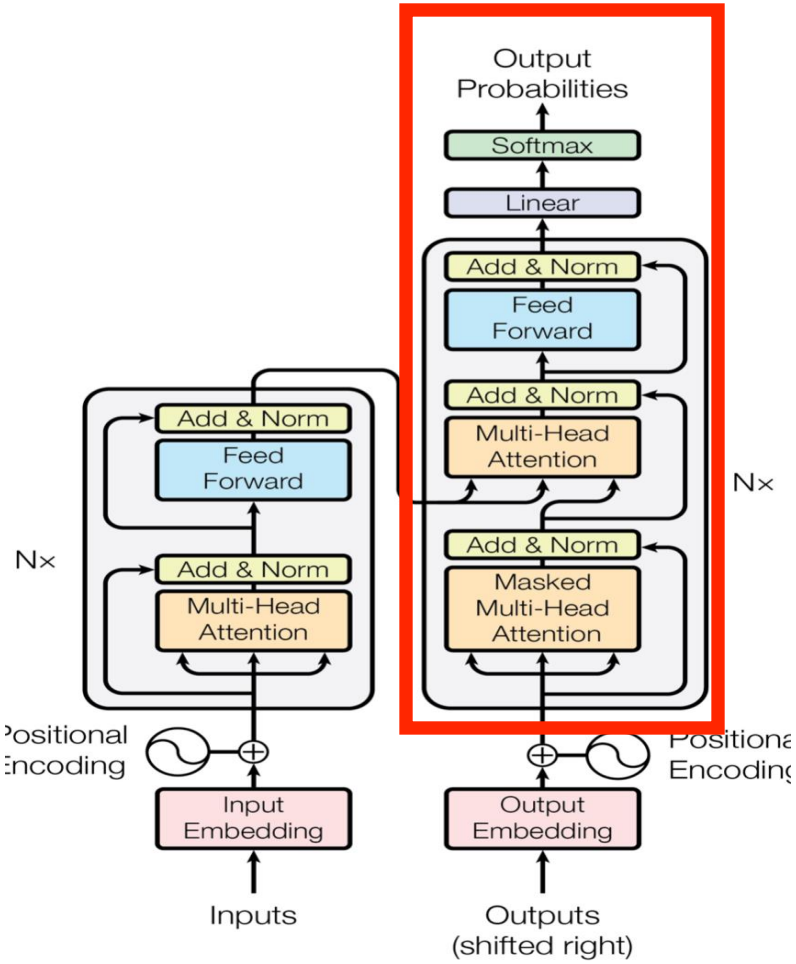
- Without recurrence and convolution, the order information is lost
- Need to encode the relative or absolute position of the tokens in the sequence
- Position encodings are added to both the embeddings of the tokens in both encoder and decoder
- Sine and cosine functions of different frequencies are used:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- Pos is the position and i is the dimension

Decoder

- N=6 identical layers
- Each layer
  - Masked multi-head attention
  - Position-wise fully connected feed-forward network
  - Multi-head attention over the output of the encoder stack
- Residual connection followed by normalization are used in all the three sublayers



Discussion: advantages of Self-Attention

- Complexity
- Short-range v.s. long-range dependency
- Interpretability

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Thanks!