

Attention Models

Sargur N. Srihari
srihari@buffalo.edu

This is part of lecture slides on [Deep Learning](#):
<http://www.cedar.buffalo.edu/~srihari/CSE676>

Topics in NLP

1. N-gram Models
2. Neural Language Models
3. High-Dimensional Outputs
4. Combining Neural Language Models with n-grams
5. Neural Machine Translation
6. Attention Models
7. Transformer Models
8. Historical Perspective

Topics in Attention Models

- Attention Vector
- General and Self Attention
- Encoder-Decoder without Attention
- Types of Attention Models
- Encoder-Decoder with Attention
- Neural network for Self-Attention

What is Attention?

- Can I have your Attention please!

ITALIAN



GERMAN



FRENCH

L' attention

KANNADA

ಗಮನ

“gamana”

HINDI

ध्यान

“dhyan”

TAMIL

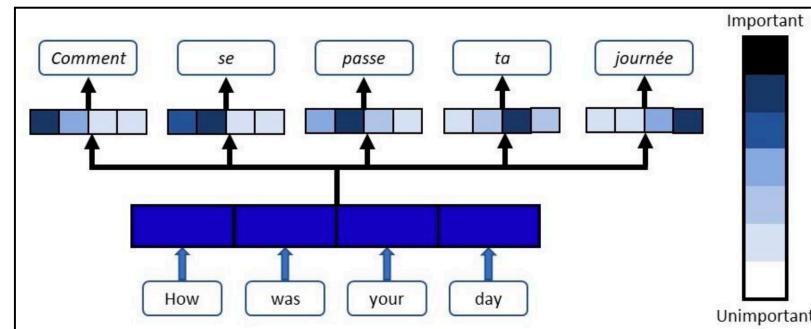
கவனம்

“kavanam”

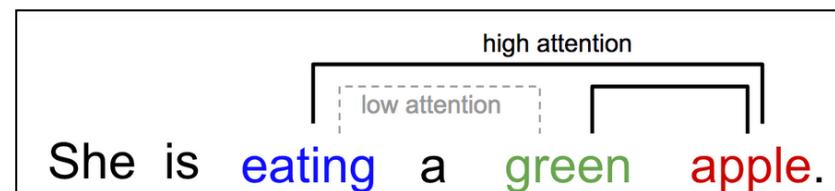
- It means to focus on something and take notice
- Paying greater attention to certain factors when processing the input

General and Self-Attention

- The attention **component** of a network manages and quantifies the **interdependence**:
 - General Attention: Between input and output elements

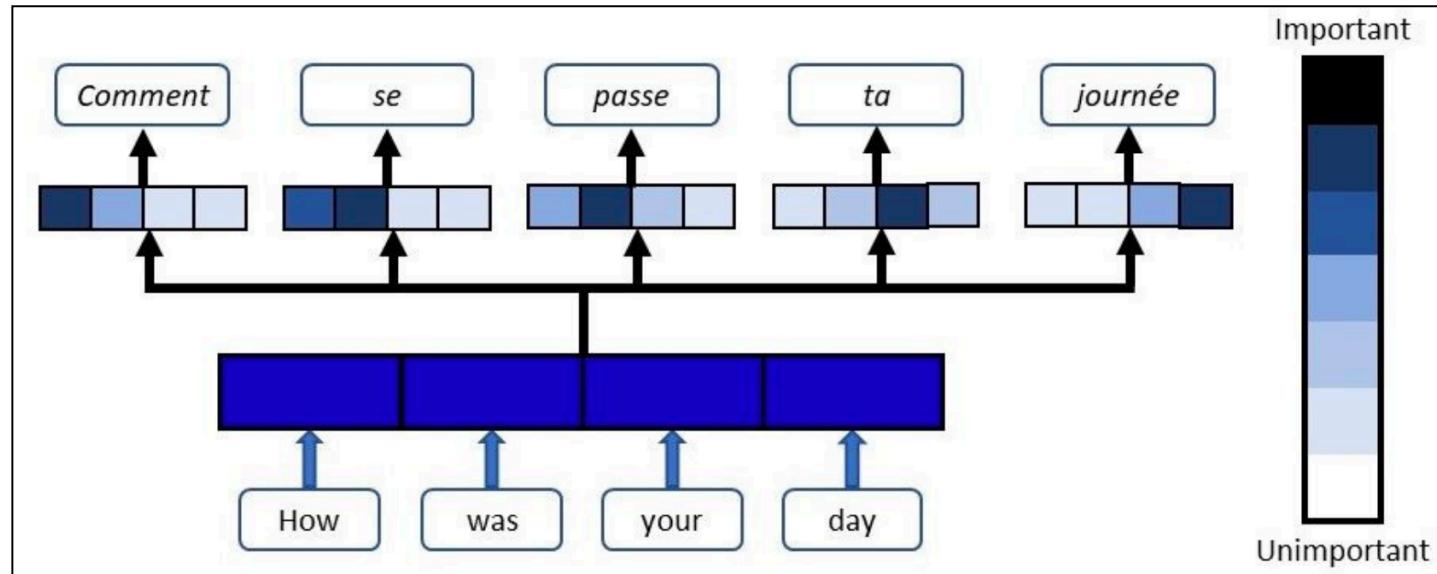


- Self Attention: Within input elements



Attention Vectors in NMT

- In Neural Machine Translation, for each output word
Assign higher weights to relevant words in input
- English to French Translation
How was your day
Comment se passe ta journée



Self Attention

- How words correlate words in a sentence

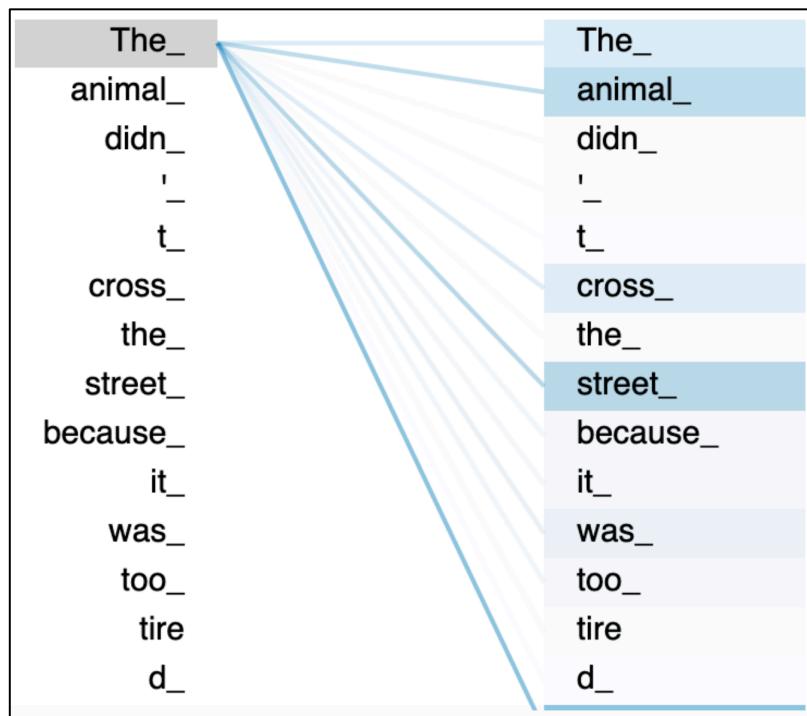


<https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>

- When we see “eating”, we expect a food word soon
 - “Green” describes food, but next word depends more on “eating”
- Attention is a vector of importance weights
 - To predict a word in a sentence, we estimate using the attention vector how strongly it is correlated with (or “*attends to*”) other elements
- This is a case of self-attention
 - Attention within different parts of same input

Self Attention Vector

- We want ML system to learn relationships between words in input sentence

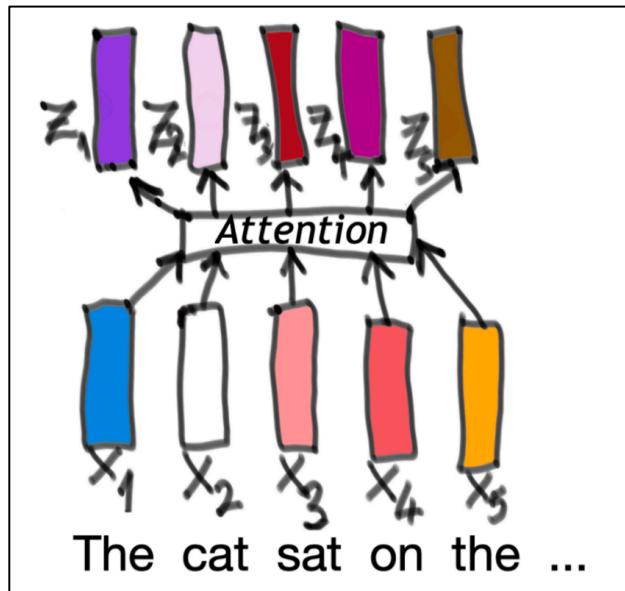


Color coding shows that there is some connection between *animal*, *cross*, *street* and *the* because they're all related to *animal*, the subject of the sentence.

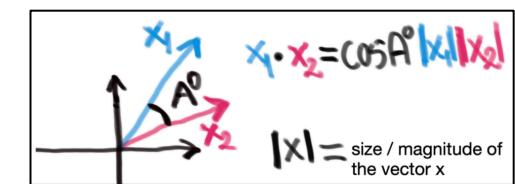
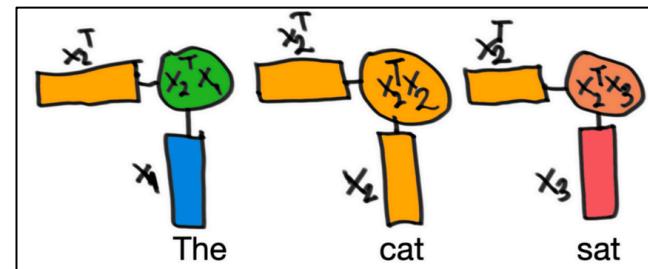
The Self-Attention Process

Self-Attention is a process

A sequence of vectors X is **encoded** into another sequence of vectors Z
 Its corresponding Z vector represents both the original word *and* its
relationship with the other words around it



Cosine similarity for x_2 considering only $x_1x_2x_3$
 Multiply each by x_2



Raw weights : w_{ij}^1

$$x_2^T x_1 = w_{21}$$

$$x_2^T x_2 = w_{22}$$

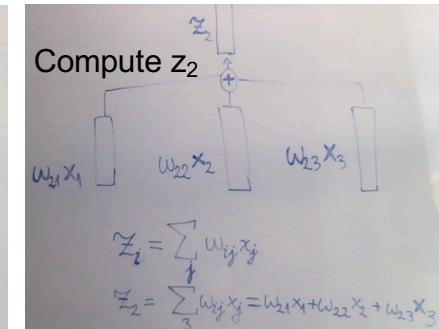
$$x_2^T x_3 = w_{23}$$

Softmax weights

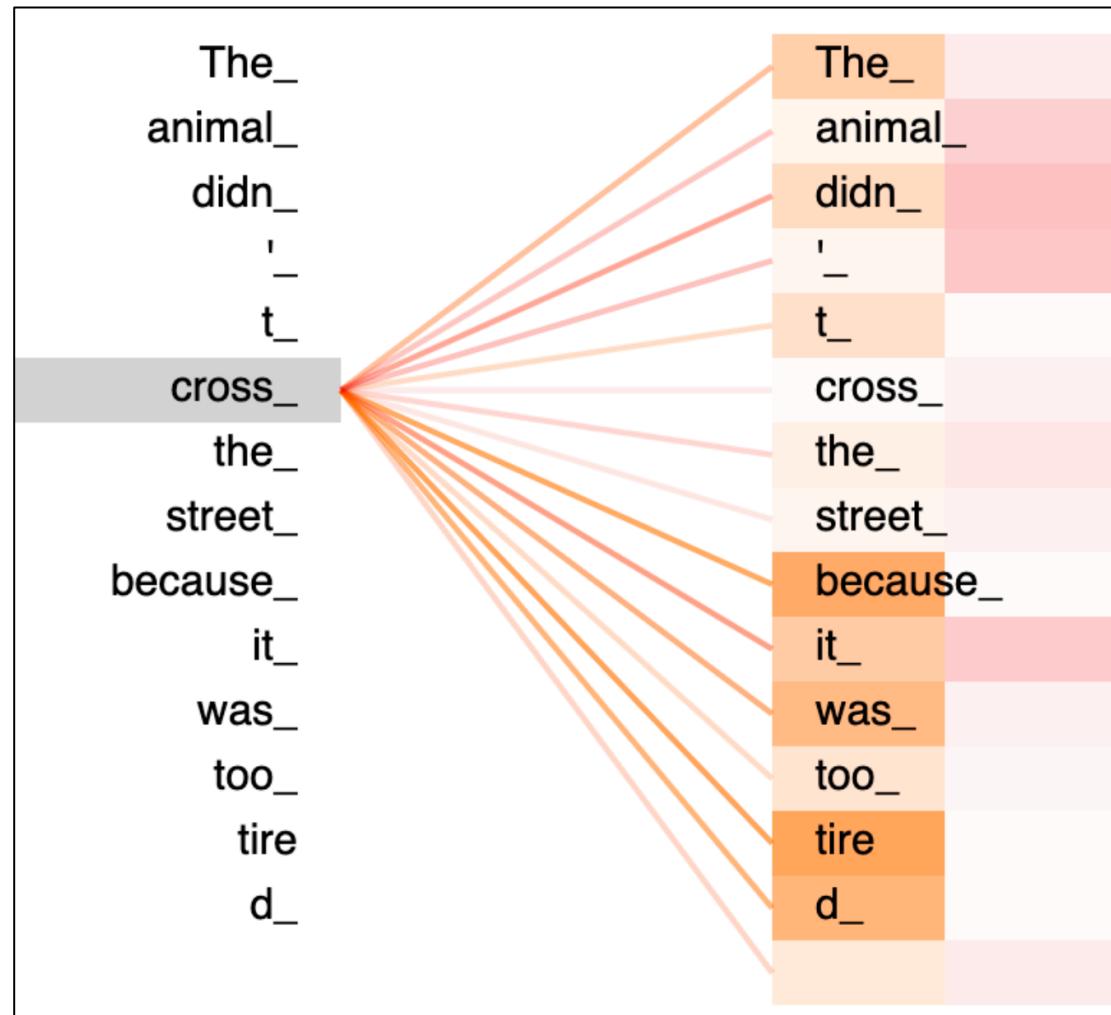
$$w_{21} = \frac{e^{w_{21}}}{\sum_j e^{w_{ij}}}$$

$$w_{22} = \frac{e^{w_{22}}}{\sum_j e^{w_{ij}}}$$

$$w_{23} = \frac{e^{w_{23}}}{\sum_j e^{w_{ij}}}$$



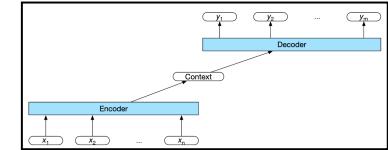
Multiple attention vectors



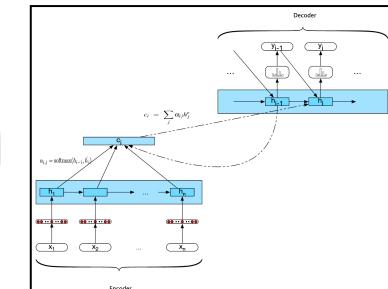
We discuss five Attention Models

1. Basic encoder-decoder RNN model

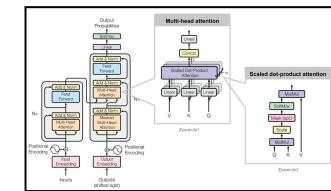
- Without attention



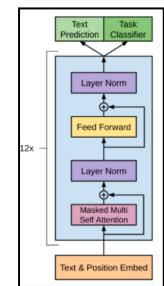
2. Encoder-decoder RNN with Attention



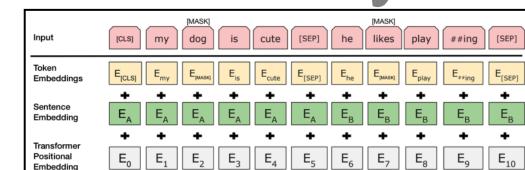
3. Transformer model with no recurrence



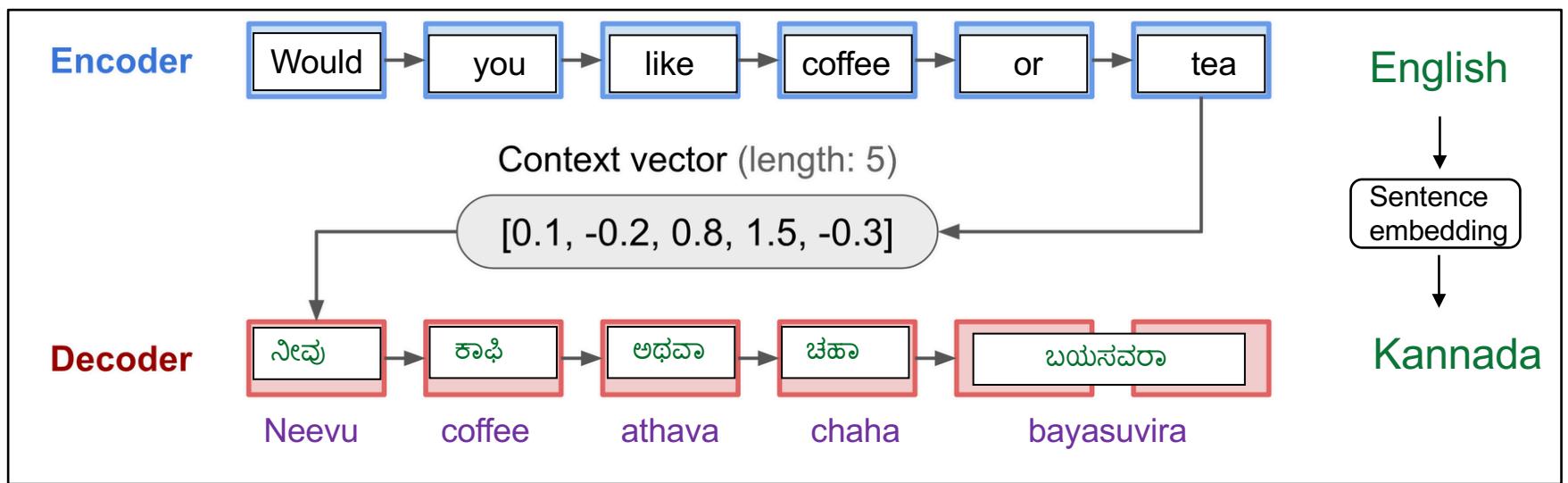
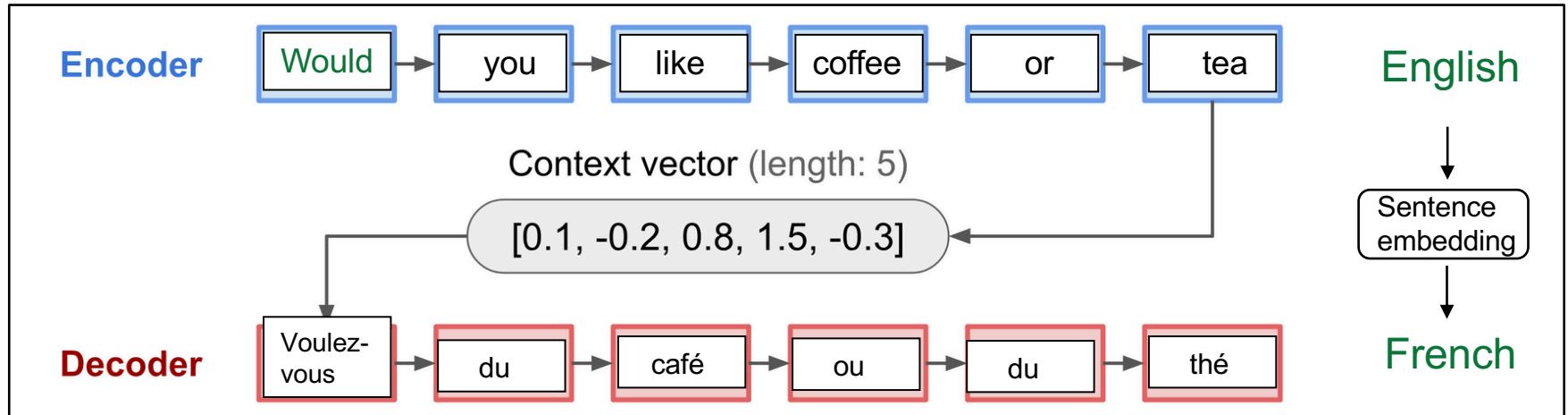
4. Generative pre-training of Attention model



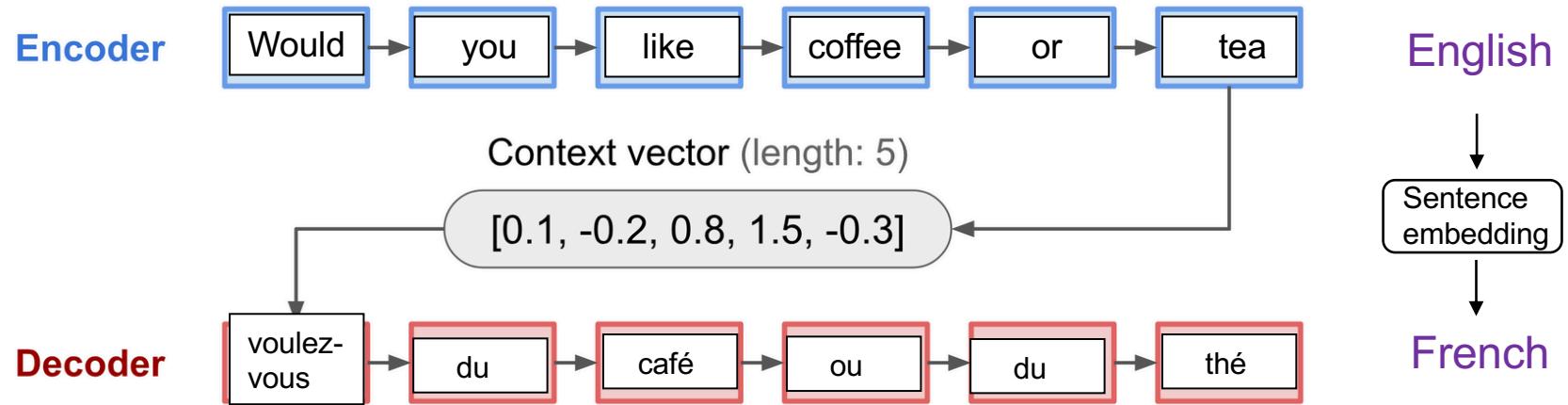
5. BERT with no decoder, using bidirectionality



Encoder-decoder without attention



Roles of Encoder-decoder in NMT



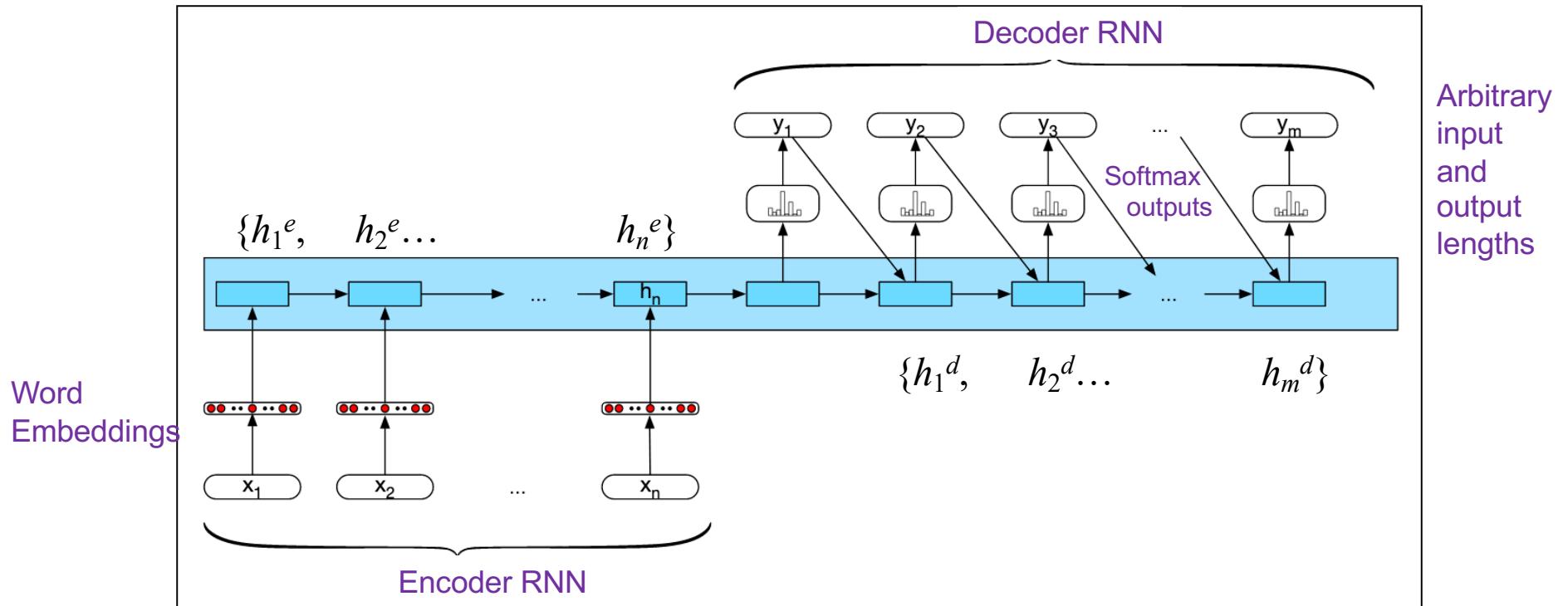
Encoder compresses input sequence into a context vector

It is a “sentence embedding” or “thought” vector of a *fixed length*.

It is expected to be a good summary of the meaning of the *whole* sequence

Decoder is initialized with the context vector to emit the transformed output

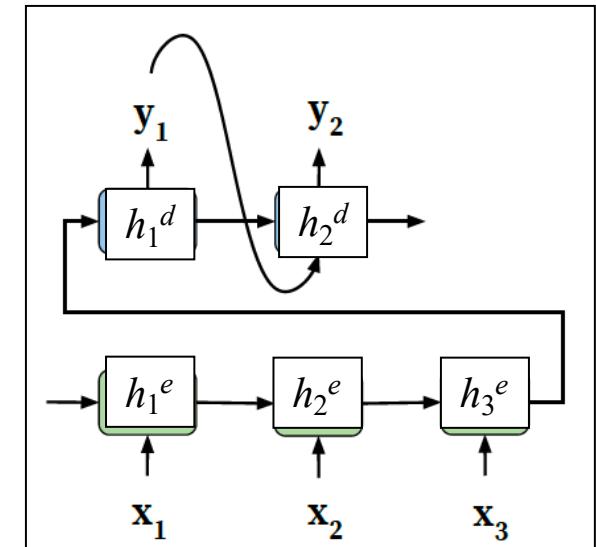
Basic RNN-based Encoder-Decoder



- Notation: At position t , h_t^e and h_t^d are the hidden states of the encoder and decoder RNN respectively
- Final encoder hidden state h_n^e serves as context for h_1^d in decoder
- Both the encoder and decoder are RNNs
e.g., [LSTM or GRU](#) units

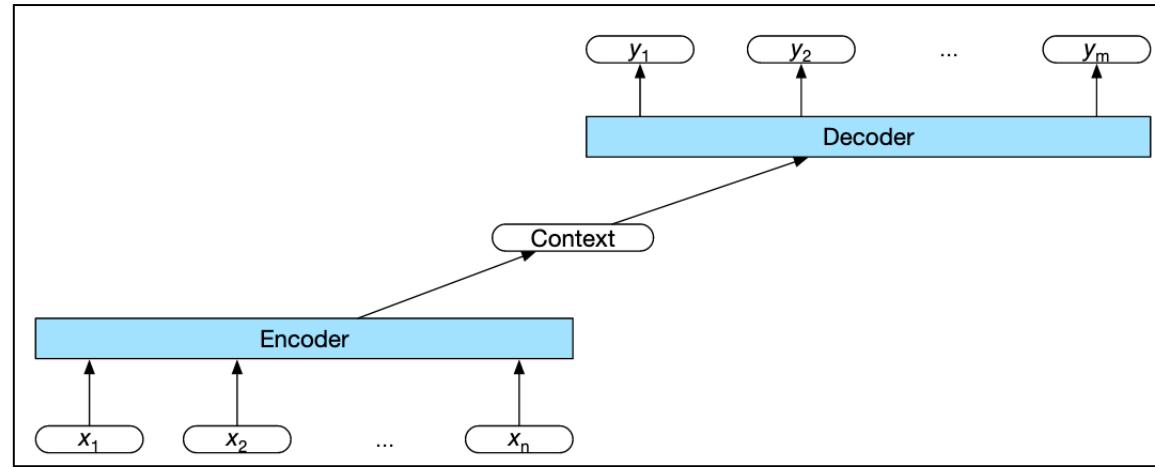
Sequence-to-sequence RNN model

- Encoder & decoder have the same structure (RNN)
 - Encoder encodes input sequence $\{x_1, x_2 \dots x_n\}$ into fixed length vectors $\{h_1^e, h_2^e \dots h_n^e\}$
 - Decoder takes single fixed length vector $\{h_n^e\}$ as input and generates output $\{y_1, y_2 \dots y_m\}$, token by token
 - Uses Autoregression
 - Consumes previous output



Summary of Encoder-Decoder

Encoder



Simple RNNs, LSTMs, GRUs, CNNs, transformer networks.

Only a single network layer shown.

In a stacked architecture output states from
the top layer are taken as the final representation.

A widely used encoder design makes use of stacked Bi-LSTMs

Decoder

Autoregressive generation is used to produce an output sequence,
an element at a time, until an end-of-sequence marker is generated.

This incremental process is guided by the context provided by the encoder
as well as any items generated for earlier states by the decoder.

Again, a typical approach is to use an LSTM or GRU-based RNN

Deficiencies of traditional encoder-decoder

1. Encoder compresses all input into a single fixed length vector h_n to be passed to decoder
 - Leads to loss of information
2. Unable to model alignment between input and output sequences
 - Which is essential for structured output tasks such as translation or summarization

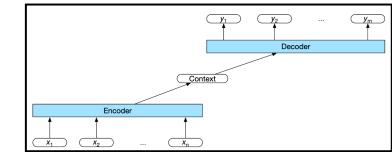
Intuition of what is needed

- In sequence-to-sequence tasks, each output token is expected to be more influenced by some specific parts of the input sequence
- However, decoder lacks any mechanism to selectively focus on relevant input tokens while generating each output token

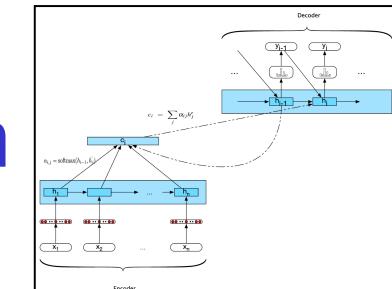
Types of Attention Models

1. Basic encoder-decoder RNN model

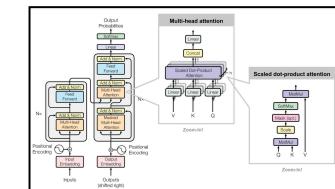
- Without attention



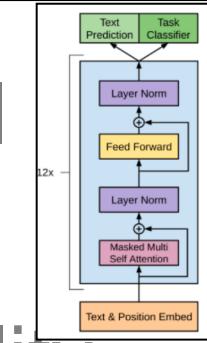
2. Encoder-decoder RNN with Attention



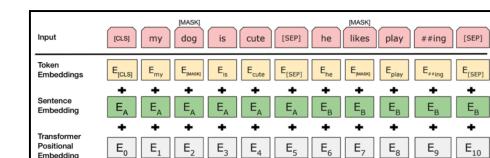
3. Transformer model with no recurrence



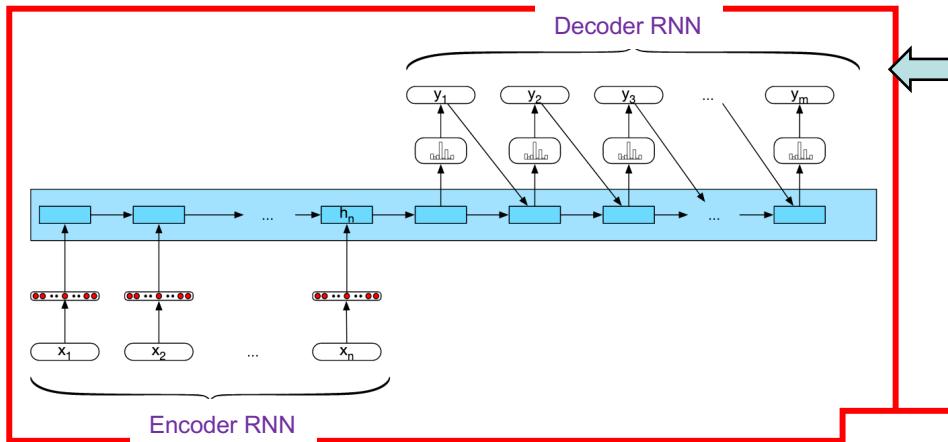
4. Generative pre-training of Attention model



5. BERT with no decoder, using bidirectionality



Encoder-Decoder without and with attention

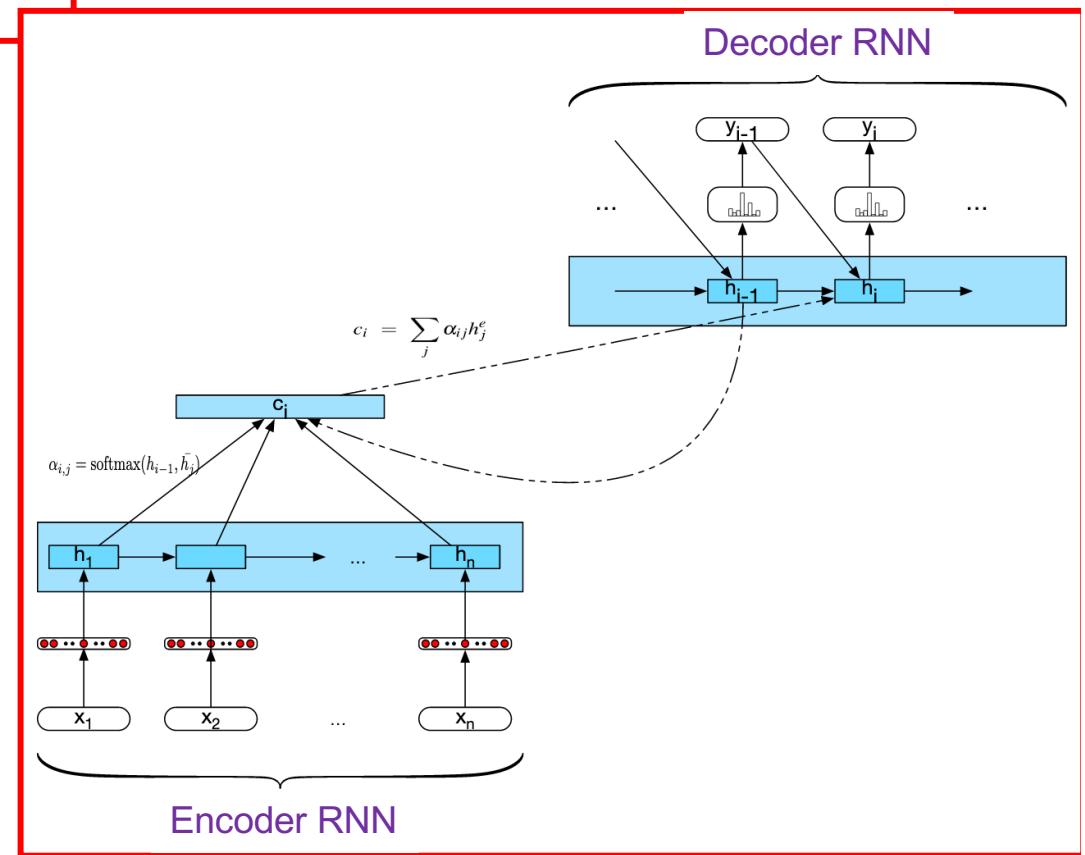


Basic Encoder-Decoder

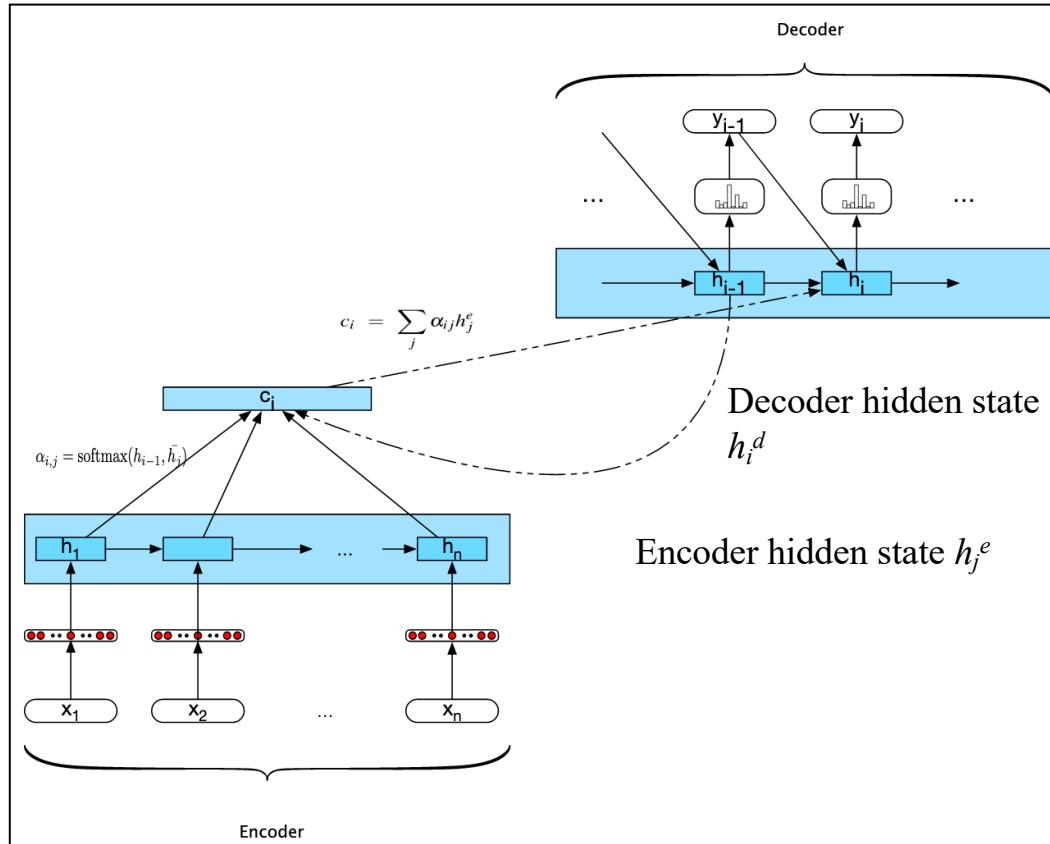
Final encoder hidden state h_n^e serves as context for h_1^d in decoder

Attention model

1. Accesses entire encoded input $\{h_1, h_2, \dots, h_n\}$
2. Induces attention weights over input to prioritize positions for generating next output
3. Dynamically updates during decoding
4. Can be embodied in a fixed-size vector



Encoder-Decoder with Attention



Decoder hidden state h_i^d is based on previous hidden state h_{i-1}^d , previous word generated y_{i-1} , and current context vector c_i

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

Context vector c_i obtained by comparing the previous hidden state h_{i-1}^d to all of the encoder hidden states h_j^e . Score is a measure of similarity of the decoder hidden state to each encoder hidden state

Alignment score functions

Dot Product

$$\text{score}(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$$

General Weight Matrix

$$\text{score}(h_{i-1}^d, h_j^e) = h_{i-1}^d W_s h_j^e$$

Neural network determines attention weights α_{ij}

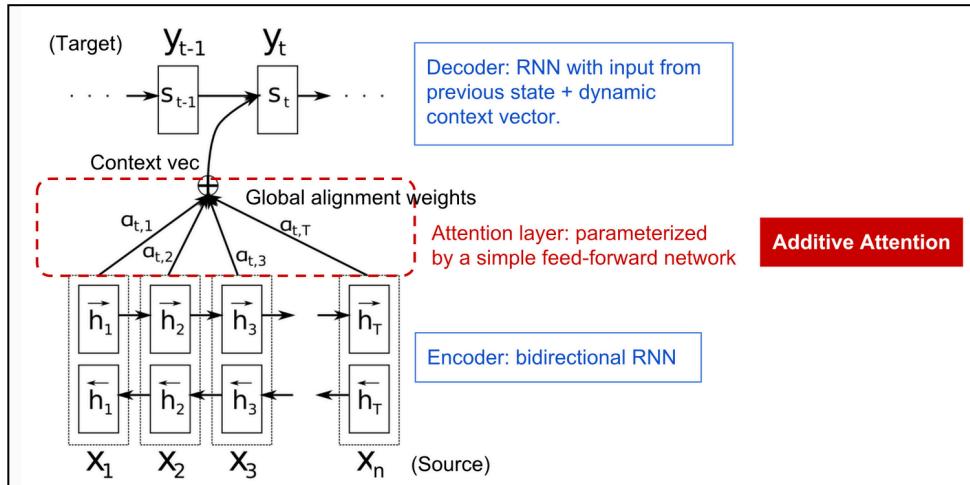
$$\begin{aligned} \alpha_{ij} &= \text{softmax}(\text{score}(h_{i-1}^d, h_j^e) \quad \forall j \in e) \\ &= \frac{\exp(\text{score}(h_{i-1}^d, h_j^e))}{\sum_k \exp(\text{score}(h_{i-1}^d, h_k^e))} \end{aligned}$$

$$c_i = \sum_j \alpha_{ij} h_j^e$$

c_i : Context vector for output y_i

α_{ij} : How well two words y_i and x_j are aligned

Bidirectional Encoder



Encoder is a bidirectional RNN with forward state $h_i \rightarrow i$ and backward $h_i \leftarrow$

A simple concatenation of two represents the encoder state.

Motivation:

Include both preceding and following words in annotating one word.

$$\mathbf{h}_i = [h_i^{\rightarrow T}; h_i^{\leftarrow T}]^T, i = 1, \dots, n$$

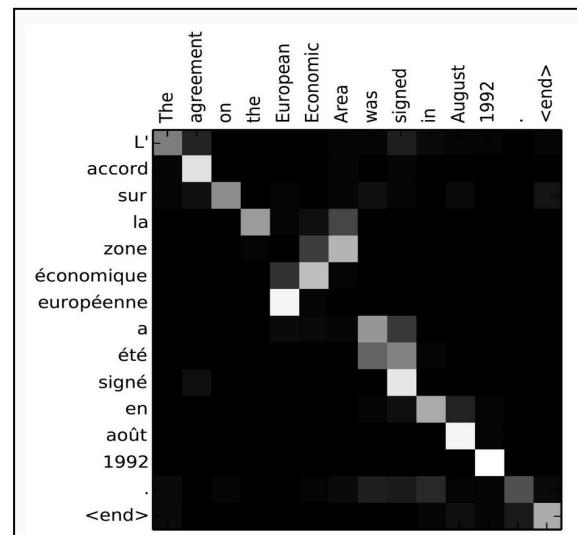
The decoder network has hidden state $s_t = f(s_{t-1}, y_{t-1}, \mathbf{c}_t)$ for the output word at position t , $t=1, \dots, m$, where the context vector \mathbf{c}_t is a sum of hidden states of the input sequence, weighted by alignment scores:

$$\mathbf{c}_t = \sum_{i=1}^n \alpha_{t,i} \mathbf{h}_i ; \text{ Context vector for output } y_t$$

$$\begin{aligned} \alpha_{t,i} &= \text{align}(y_t, x_i) &&; \text{How well two words } y_t \text{ and } x_i \text{ are aligned.} \\ &= \frac{\exp(\text{score}(s_{t-1}, \mathbf{h}_i))}{\sum_{i'=1}^n \exp(\text{score}(s_{t-1}, \mathbf{h}_{i'}))} &&; \text{Softmax of some predefined alignment score..} \end{aligned}$$

Example of Alignment Matrix α_{ij}

- Matrix of alignment scores is a byproduct to explicitly show the correlation between source and target words.
- Alignment matrix of French input
“L'accord sur la zone économique européenne a été signé en août 1992”
- Its English translation
“The agreement on the European Economic Area was signed in August 1992”



Alignment score functions

$$\begin{aligned}\alpha_{ij} &= \text{softmax(score}(h_{i-1}^d, h_j^e) \forall j \in e) \\ &= \frac{\exp(\text{score}(h_{i-1}^d, h_j^e))}{\sum_k \exp(\text{score}(h_{i-1}^d, h_k^e))}\end{aligned}$$

Name	Alignment score function	Citation
Content-base attention	$\text{score}(s_t, \mathbf{h}_i) = \text{cosine}[s_t, \mathbf{h}_i]$	Graves2014
Additive(*)	$\text{score}(s_t, \mathbf{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[s_t; \mathbf{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a s_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(s_t, \mathbf{h}_i) = s_t^\top \mathbf{W}_a \mathbf{h}_i$ where \mathbf{W}_a is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(s_t, \mathbf{h}_i) = s_t^\top \mathbf{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(s_t, \mathbf{h}_i) = \frac{s_t^\top \mathbf{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Self-Attention

- Self-attention, also known as *intra-attention*
 - Relates different positions of a single sequence to compute a representation of the same sequence
 - As opposed to attention over entire input space



- Useful in machine reading, text summarization, image description generation
- Broad categories of Attention:

Name	Definition	Citation
Self-Attention(&)	Relating different positions of the same input sequence. Theoretically the self-attention can adopt any score functions above, but just replace the target sequence with the same input sequence.	Cheng2016
Global/Soft	Attending to the entire input state space.	Xu2015
Local/Hard	Attending to the part of input state space; i.e. a patch of the input image.	Xu2015; Luong2015

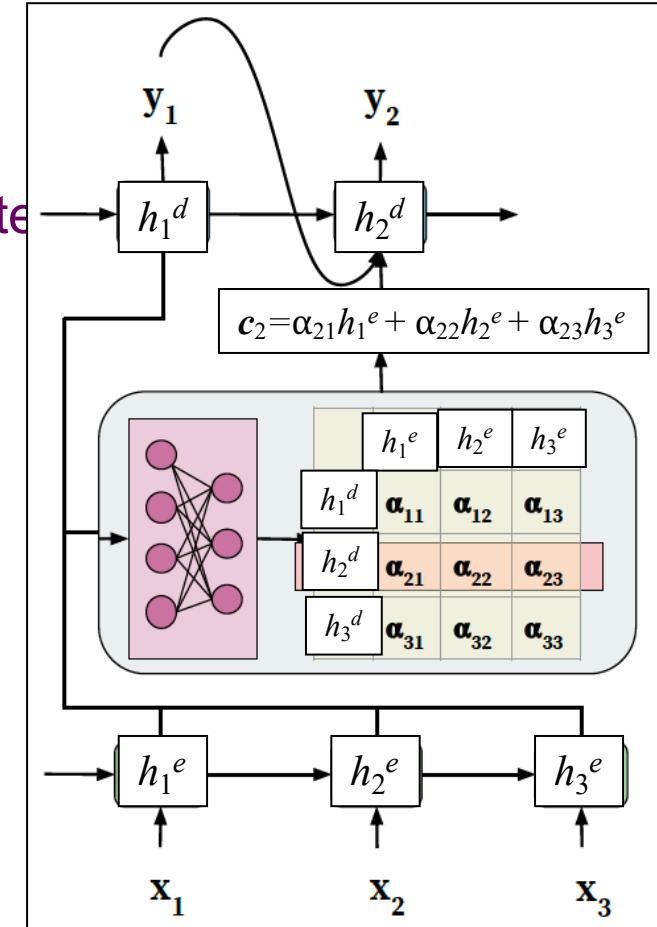
Neural network for learning α_{ij}

- α_{ij} capture relevance between
 - h_i^d (decoder hidden state, or *query state*)
 - h_j^e (encoder hidden state, or *candidate state*)
- α_{ij} used to build a context vector c_i at each decoding position i ,
 - A weighted sum of encoder hidden states and attention weights

$$c_i = \sum_j \alpha_{ij} h_j^e$$

- Context vectors c_i determine next decoder hidden states

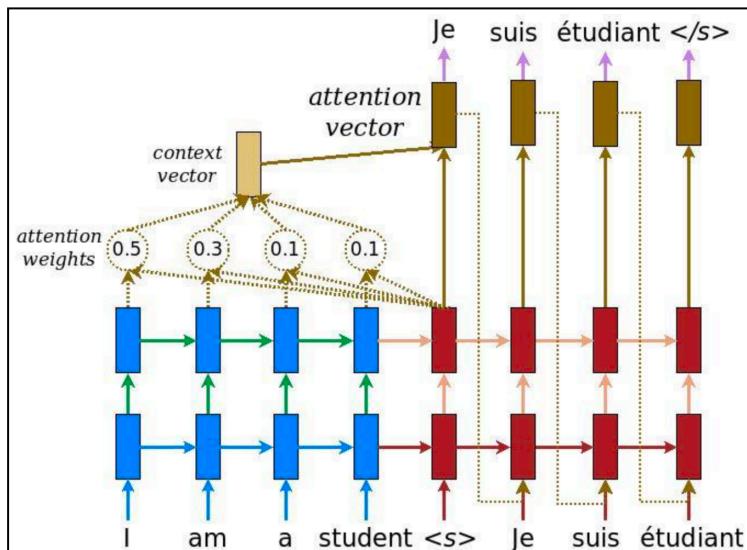
$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$



Context vector is how the decoder can access entire input sequence and also focus on its relevant positions

Tensorflow: NMT with Attention

https://www.tensorflow.org/tutorials/text/nmt_with_attention



Input: <start> hace mucho frio aqui . <end>
Predicted translation: it s very cold here . <end>

[Run in Google Colab](#) [View source on GitHub](#) [Download notebook](#)

$$\alpha_{ts} = \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'=1}^S \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))} \quad [\text{Attention weights}] \quad (1)$$

$$\mathbf{c}_t = \sum_s \alpha_{ts} \bar{\mathbf{h}}_s \quad [\text{Context vector}] \quad (2)$$

$$\mathbf{a}_t = f(\mathbf{c}_t, \mathbf{h}_t) = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]) \quad [\text{Attention vector}] \quad (3)$$

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \mathbf{W} \bar{\mathbf{h}}_s \\ \mathbf{v}_a^\top \tanh(\mathbf{W}_1 \mathbf{h}_t + \mathbf{W}_2 \bar{\mathbf{h}}_s) \end{cases} \quad [\text{Luong's multiplicative style}] \quad [\text{Bahdanau's additive style}] \quad (4)$$

Plot of attention weights

