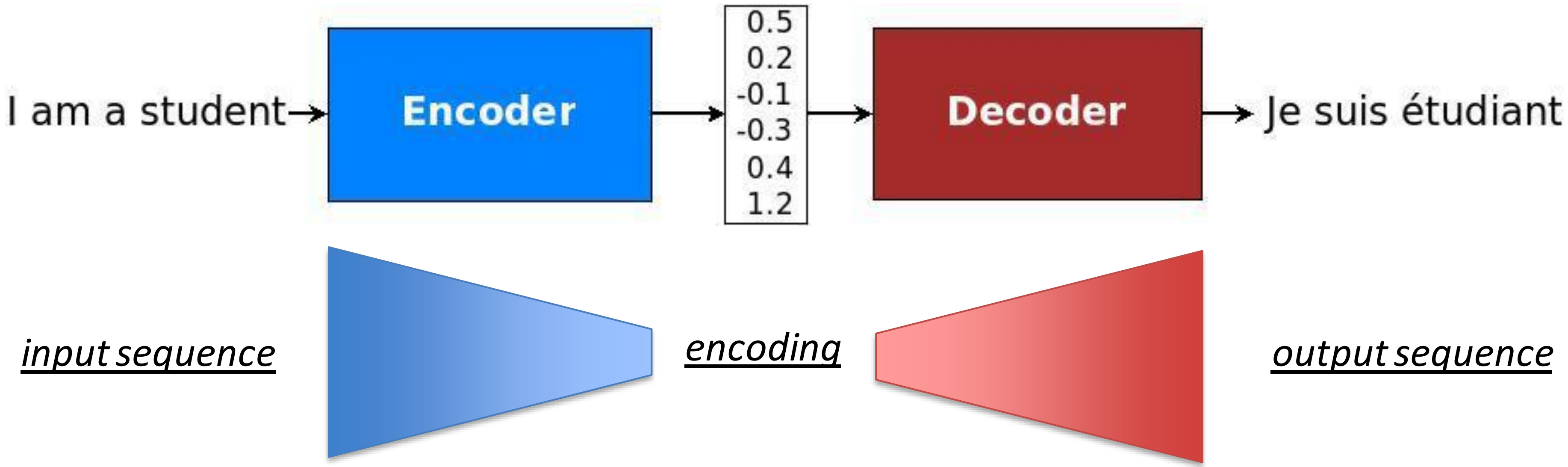
Artificial Neural Networks and Deep Learning

- Beyond Seq2Seq Architectures-

Matteo Matteucci, PhD (matteo.matteucci@polimi.it)
Artificial Intelligence and Robotics Laboratory
Politecnico di Milano

Extending Recurrent Neural Networks

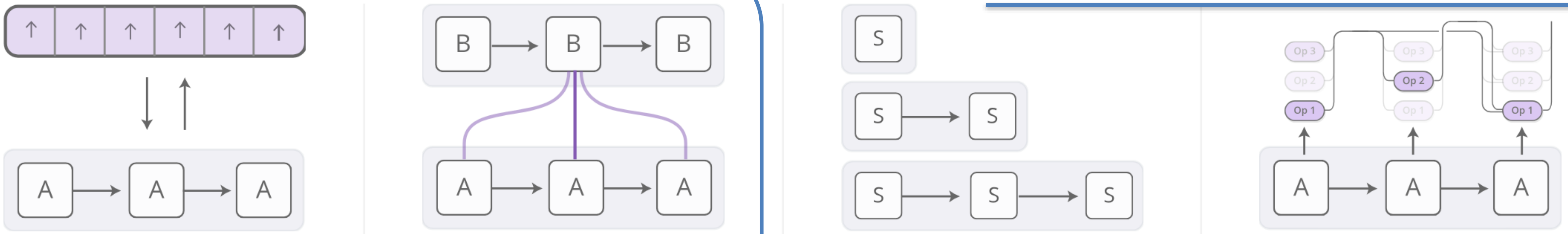
Recurrent Neural Networks have been extended with memory to cope with very long sequences and the encoding bottleneck ...



Extending Recurrent Neural Networks

Recurrent Neural Networks have been extended with memory to cope with very long sequences and the encoding bottleneck ...

https://distill.pub/2016/augmented-rnns/



Neural Turing Machines have external memory that they can read and write to.

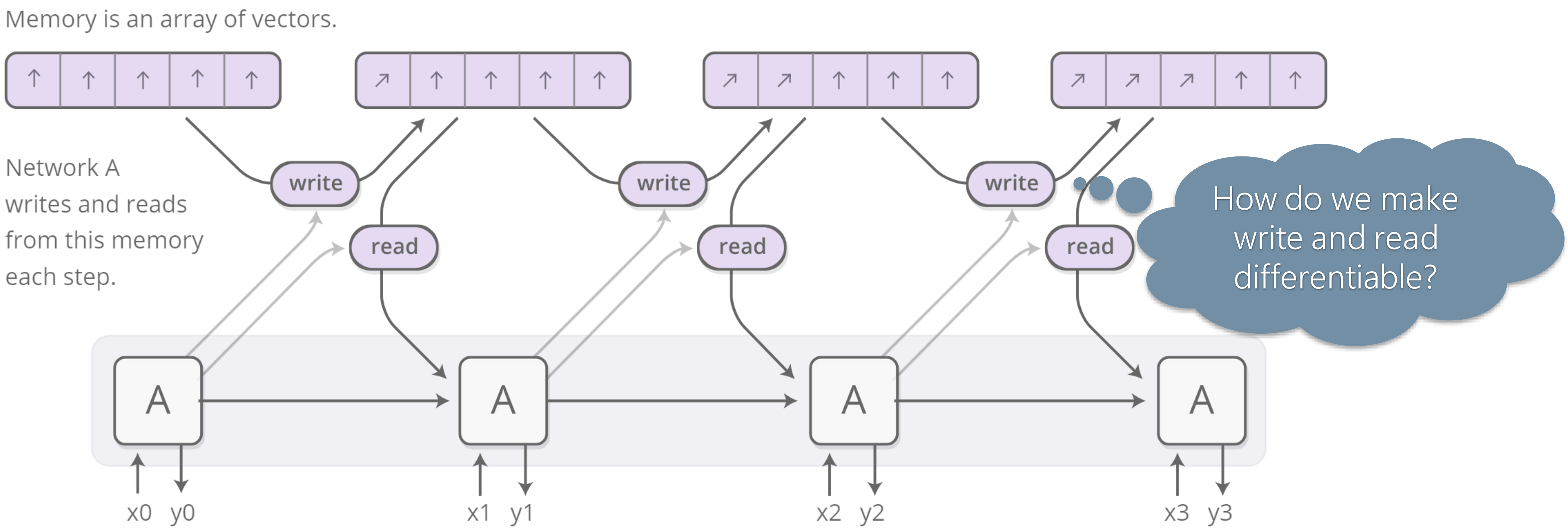
Attentional Interfaces allow RNNs to focus on parts of their input.

Adaptive Computation Time allows for varying amounts of computation per step.

Neural Programmers can call functions, building programs as they run.

Neural Turing Machines

Neural Turing Machines combine a RNN with an external memory bank.



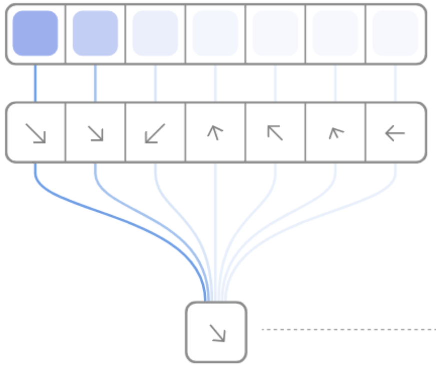
Neural Turing Machines Idea

Neural Turing Machines challenge:

- We want to learn what to write/read but also where to write it
- Memory addresses are be fundamentally discrete
- Write/read differentiable w.r.t the location we read from or write

Attention mechanism!

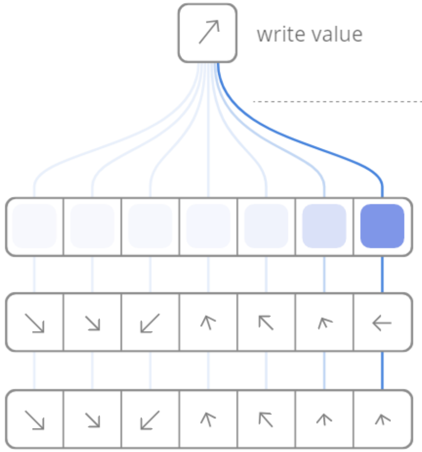
Solution: <u>*Every step, read and write everywhere, just to different extents*</u>.



The RNN gives an attention distribution which describe how we spread out the amount we care about different memory positions.

The read result is a weighted sum.

$$r \leftarrow \sum_i a_i M_i$$



Instead of writing to one location, we write everywhere, just to different extents.

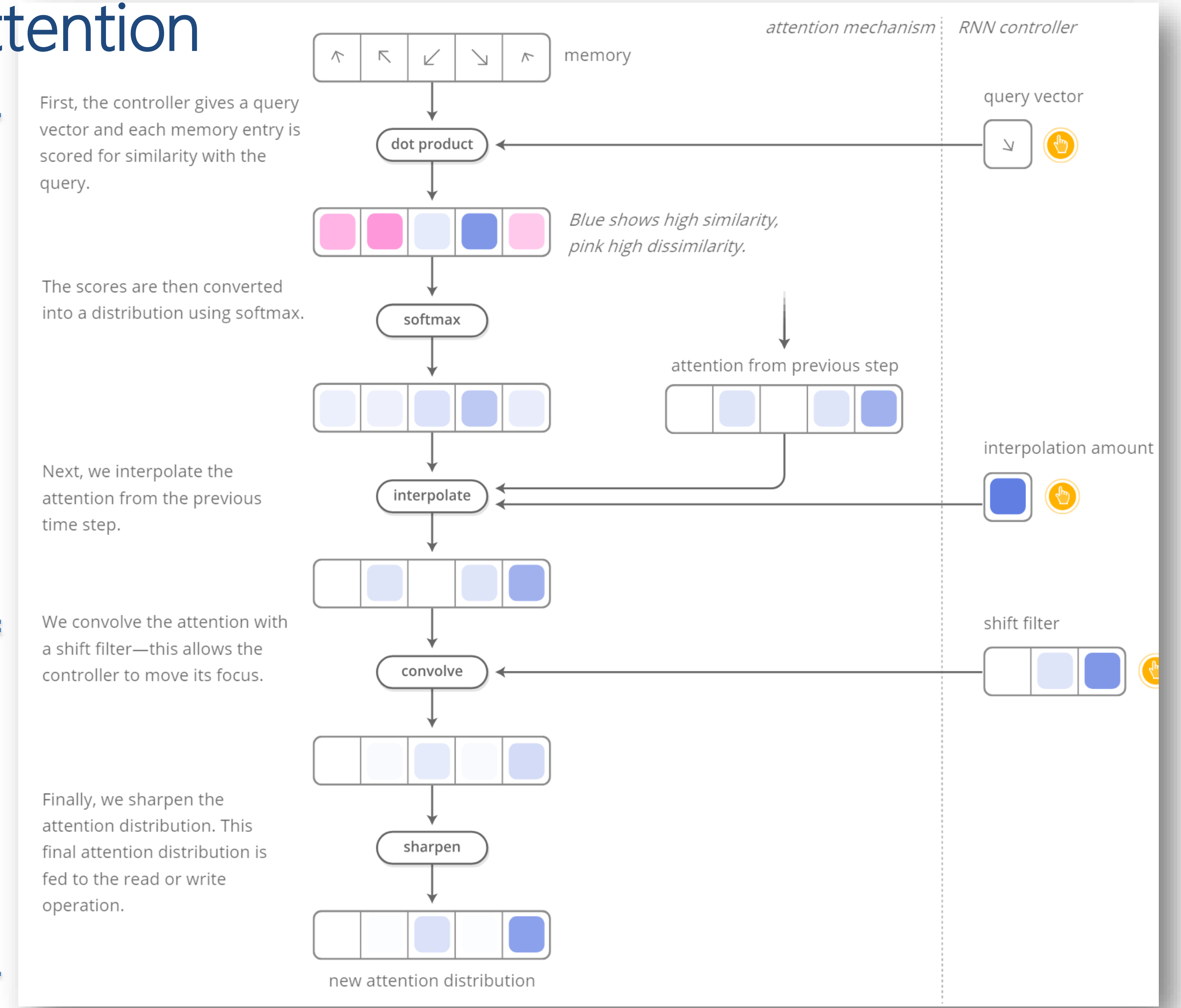
The RNN gives an attention distribution, describing how much we should change each memory position towards the write value.

$$M_i \leftarrow a_i w + (1 - a_i) M_i$$

Neural Turing Machines Attention

***<u>Content-based attention</u>**: searches memory and focus on places that match what they're looking for*

***<u>Location-based attention</u>**: allows relative movement in memory enabling the NTM to loop.*



Neural Turing Machines Extensions

NTM perform algorithms, previously beyond neural networks:

- Learn to store a long sequence in memory
- Learn to loop and repeat sequences back repeatedly
- Learn to mimic a lookup table
- Learn to sort numbers …

But the most interesting thing (to me) was the attention mechanism!

Some extension have been proposed to go beyond this:

- Neural GPU overcomes the NTM's inability to add and multiply numbers
- Zaremba & Sutskever train NTMs using reinforcement learning instead of the differentiable read/writes used by the original
- Neural Random Access Machines work based on pointers
- Others have explored differentiable data structures, like stacks and queues

Attention Mechanism in Seq2Seq Models

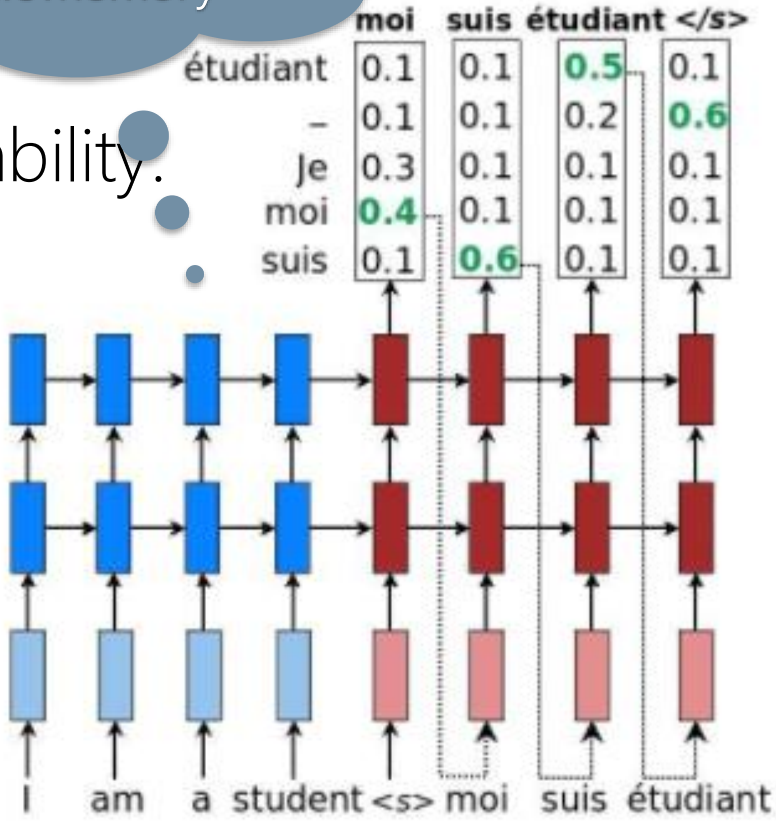
Attention on the past hidded states used as dynamic memory

Considering the sequential dataset:

$$\{((x_1, \ldots, x_n), (y_1, \ldots, y_m))\}_{i=1}^{N}$$

The decoder role is to model the generative probability:

$$P(y_1, \ldots, y_m | x)$$

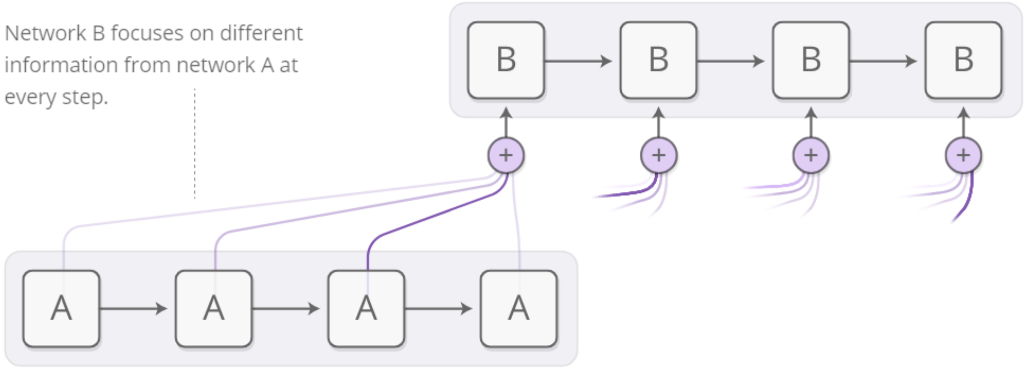


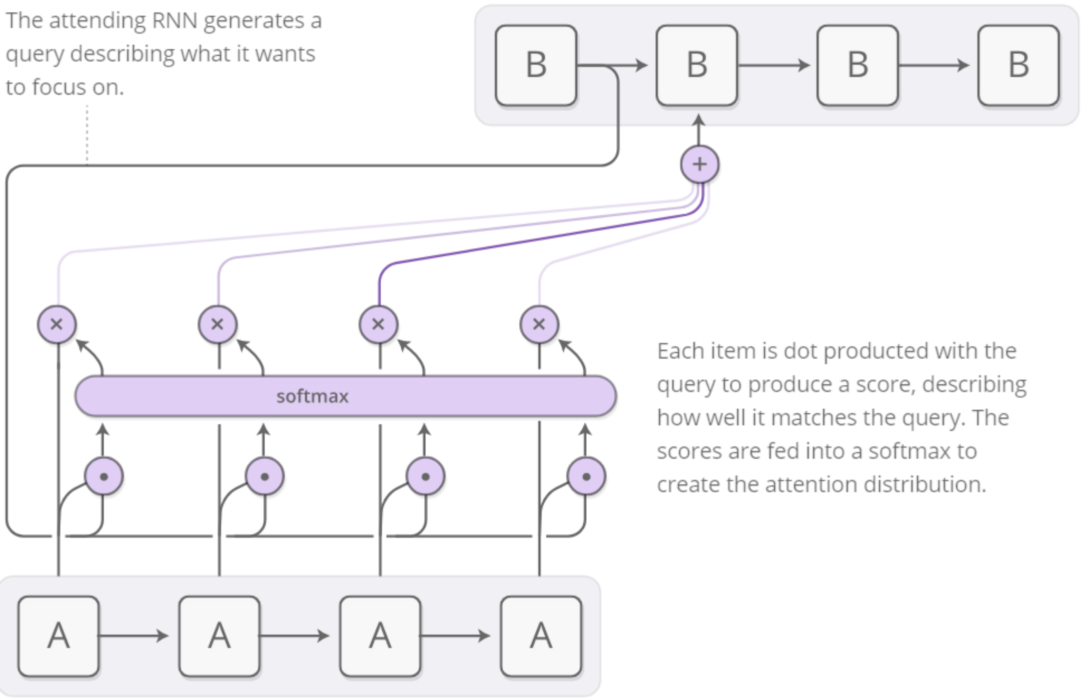
In "vanilla" seq2seq models, the decoder is conditioned initializing the initial state with last state of the encoder.

Works well for short and medium-length sentences; however, for long sentences, becomes a bottleneck

Attention Mechanism in Seq2Seq Models

Let’s use the same idea of Neural Turing Machines to get a differentiable attention and learn where to focus attention.





Attention distribution is usually generated with content-based attention.

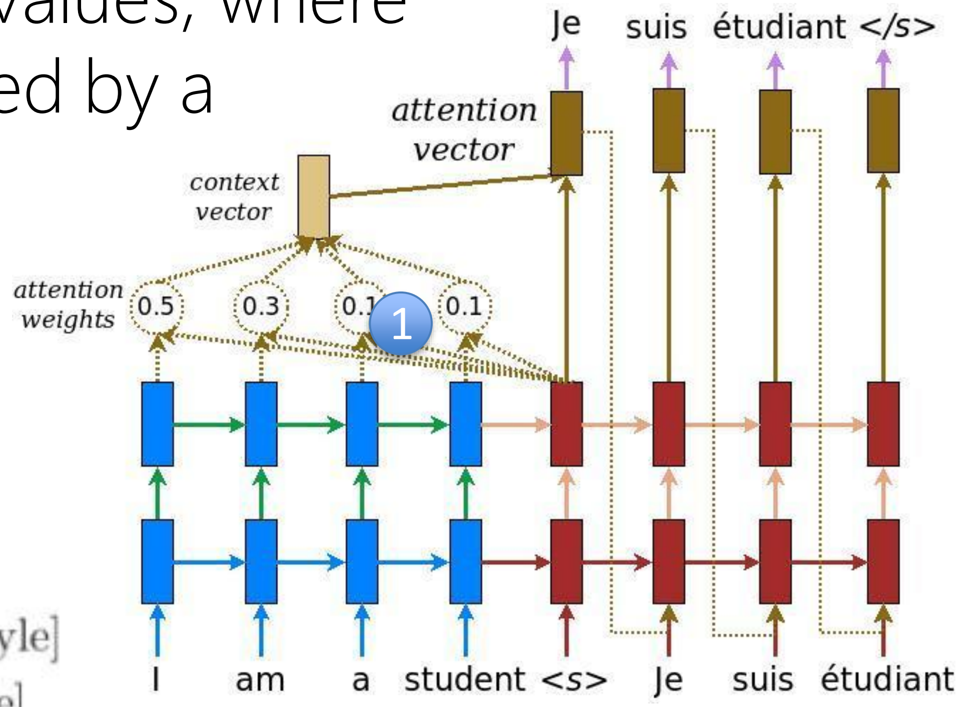
Each item is thus weighted with the query response to produce a score

Scores are fed into a softmax to create the attention distribution

Attention Mechanism in Seq2Seq Models

Attention function maps query and set of key-value pairs to an output.

Output computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function:



1. Compare current target hidden state $\boldsymbol{h_t}$, with source states $\boldsymbol{h_s}$ to derive attention

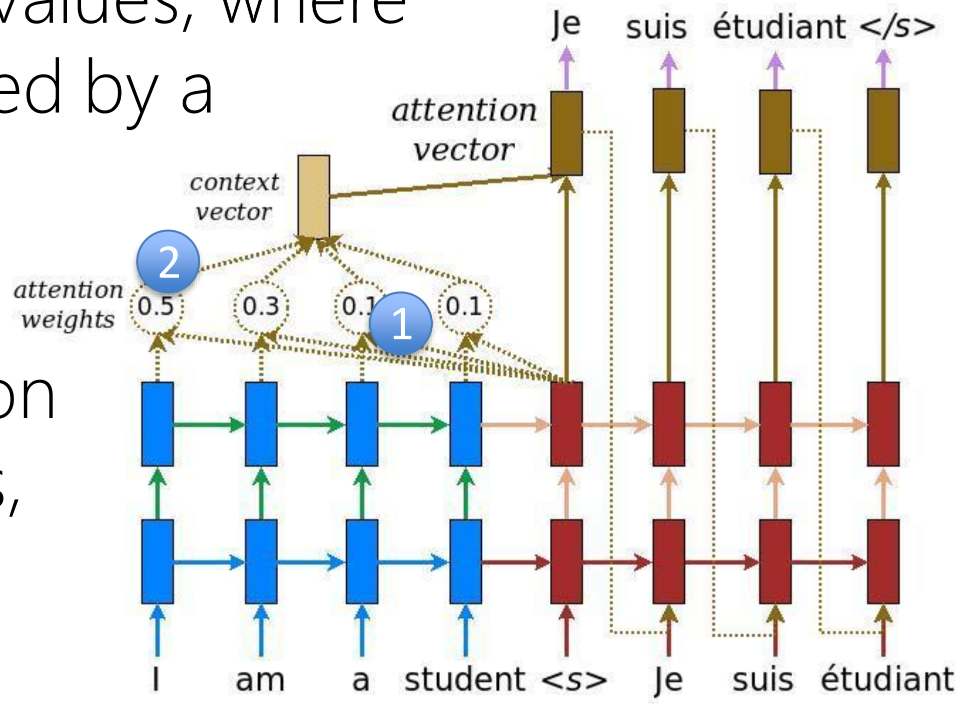
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_1 \boldsymbol{h}_t + \boldsymbol{W}_2 \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's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

Attention Mechanism in Seq2Seq Models

Attention function maps query and set of key-value pairs to an output.

Output computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function:

2. Apply the softmax function on the attention scores and compute the attention weights, one for each encoder token

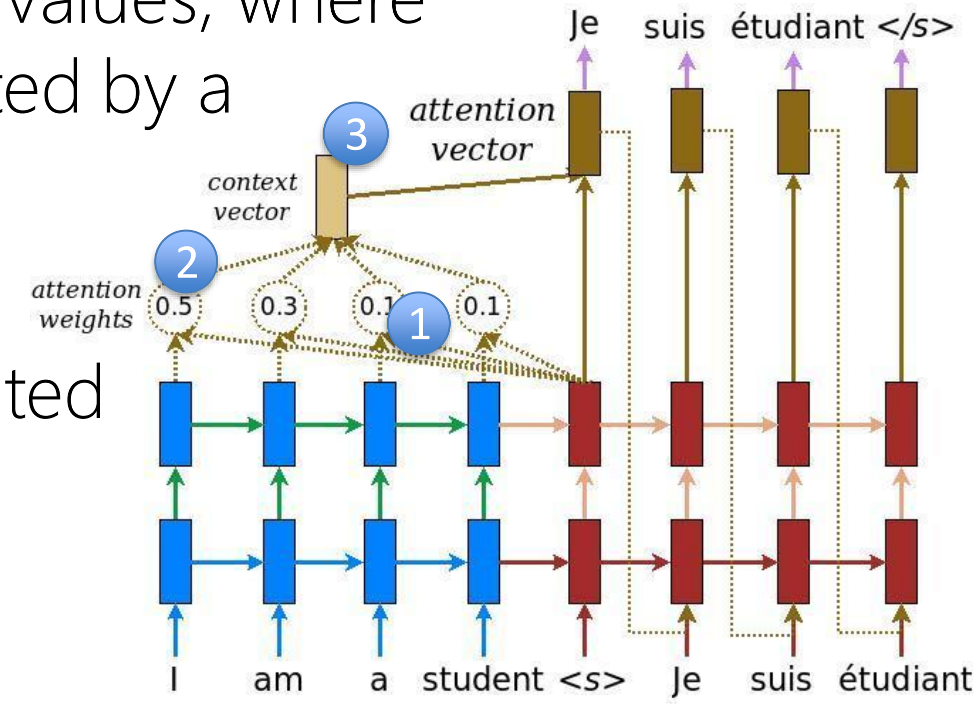


$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

Attention Mechanism in Seq2Seq Models

Attention function maps query and set of key-value pairs to an output.

Output computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function:



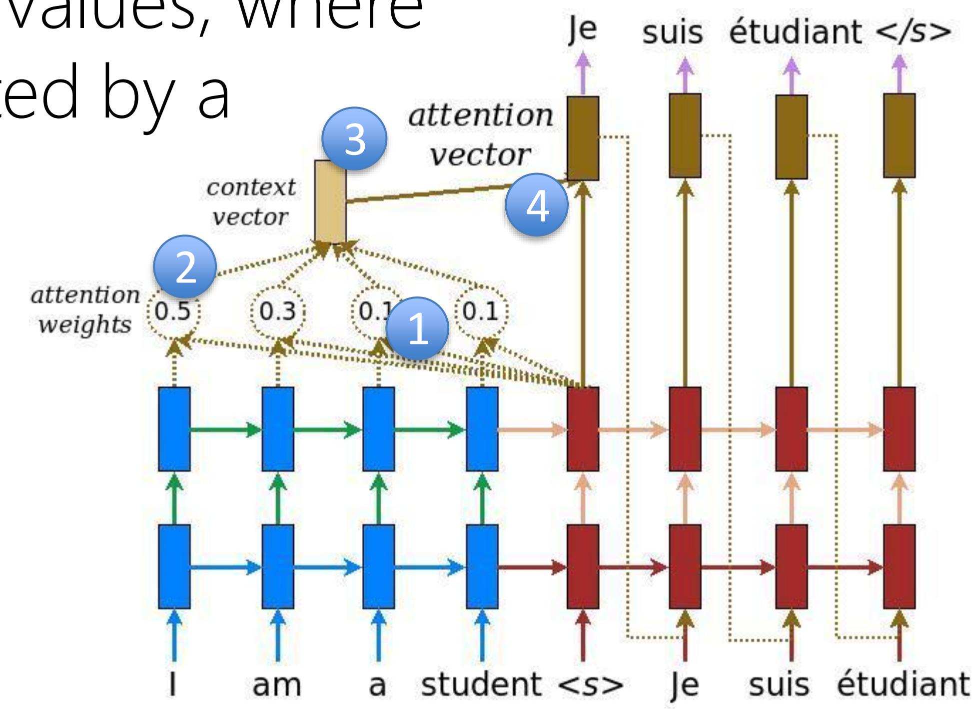
3. Compute the context vector as the weighted average of the source states

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s$$

Attention Mechanism in Seq2Seq Models

Attention function maps query and set of key-value pairs to an output.

Output computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function:



4. Combine the context vector with current target hidden state to yield the final attention vector

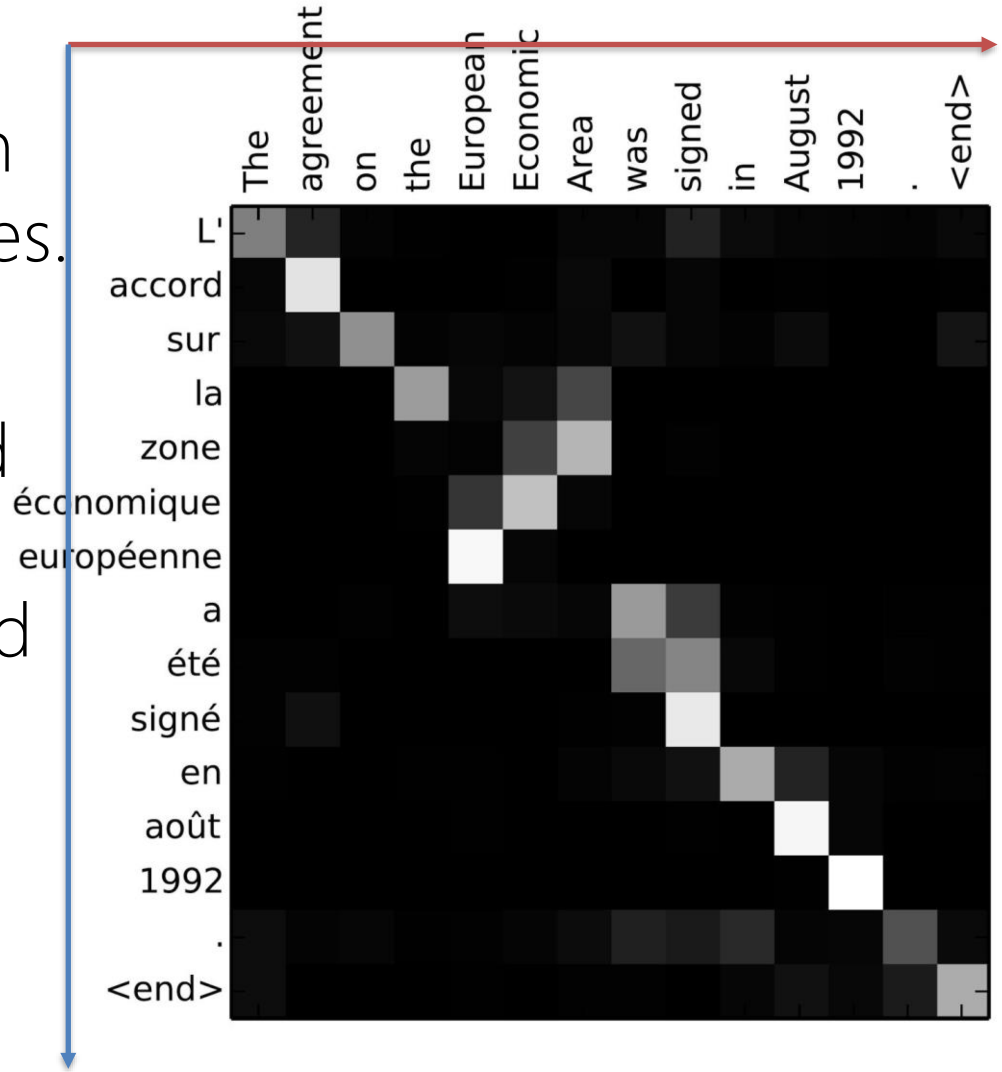
$$a_t = f(c_t, h_t) = \tanh(W_c[c_t; h_t])$$

Attention Visualization

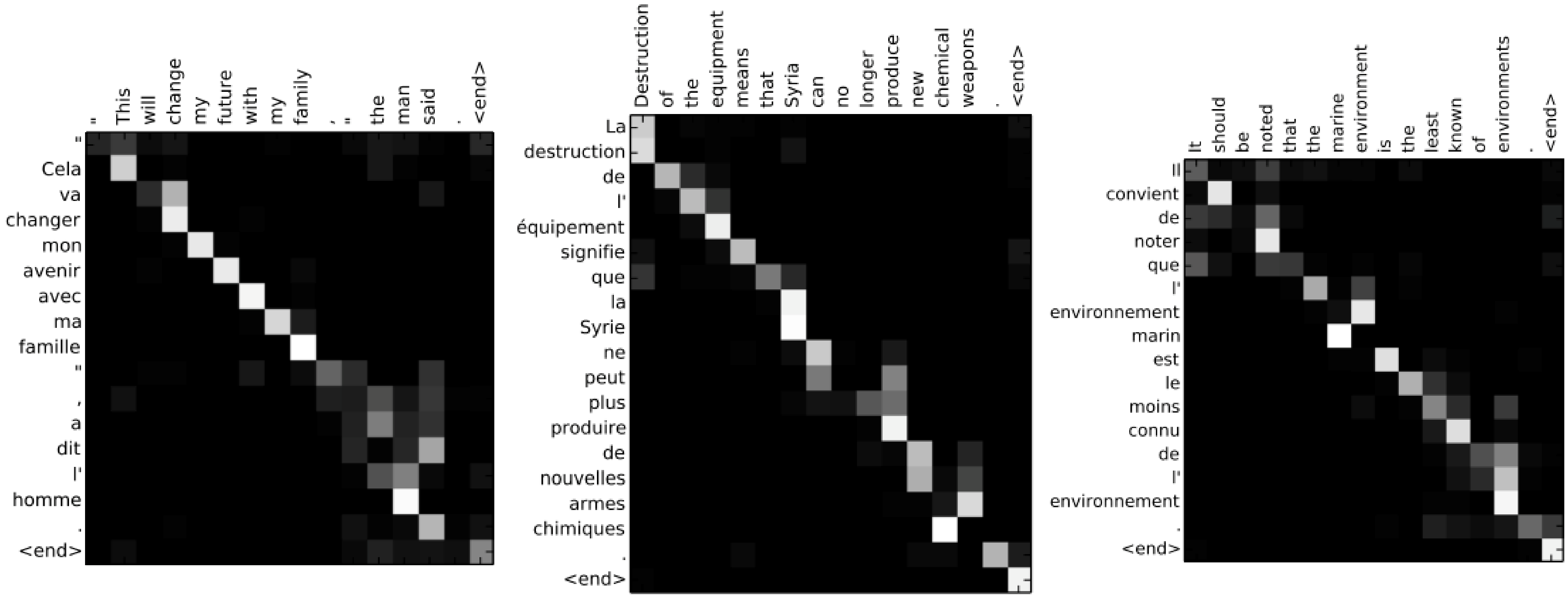
Alignment matrix is use to visualize attention weights between source and target sentences.

For each decoding step, i.e., each generated target token, describes which are the source tokens that are more present in the weighted sum that conditioned the decoding.

We can see attention as a tool in the network's bag that, while decoding, allows it to pay attention on different parts of the source sentence.



Attention Visualization



Attention Mechanism in Translation

Check the demo!!!

Attention allows processing the input to pass along information about each word it sees, and then for generating the output to focus on words

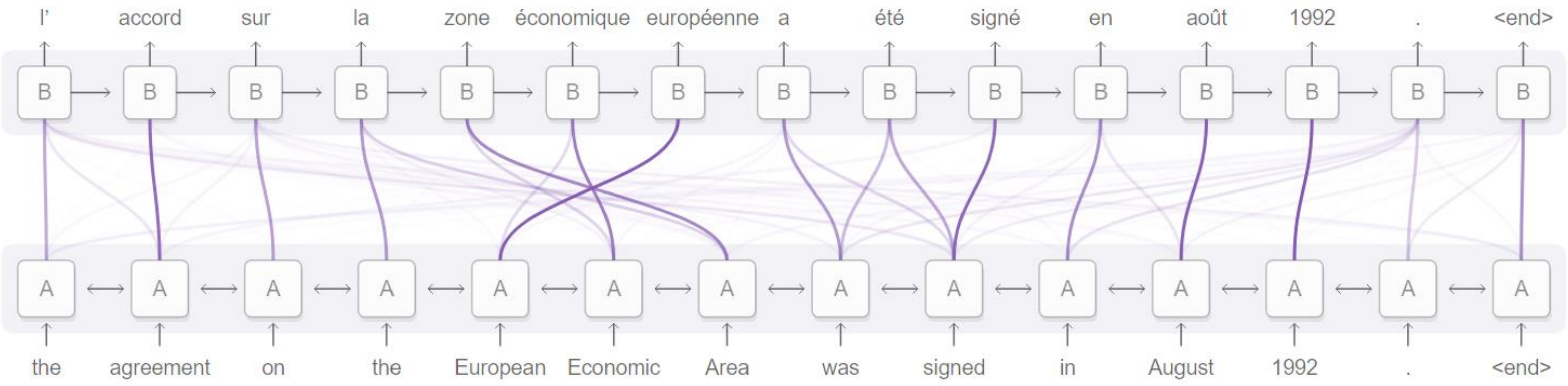


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

Attention Mechanism in Voice Recognition

Check the demo!!!

Attention allows one RNN to process the audio and then have another RNN skim over it, focusing on relevant parts as it generates a transcript.

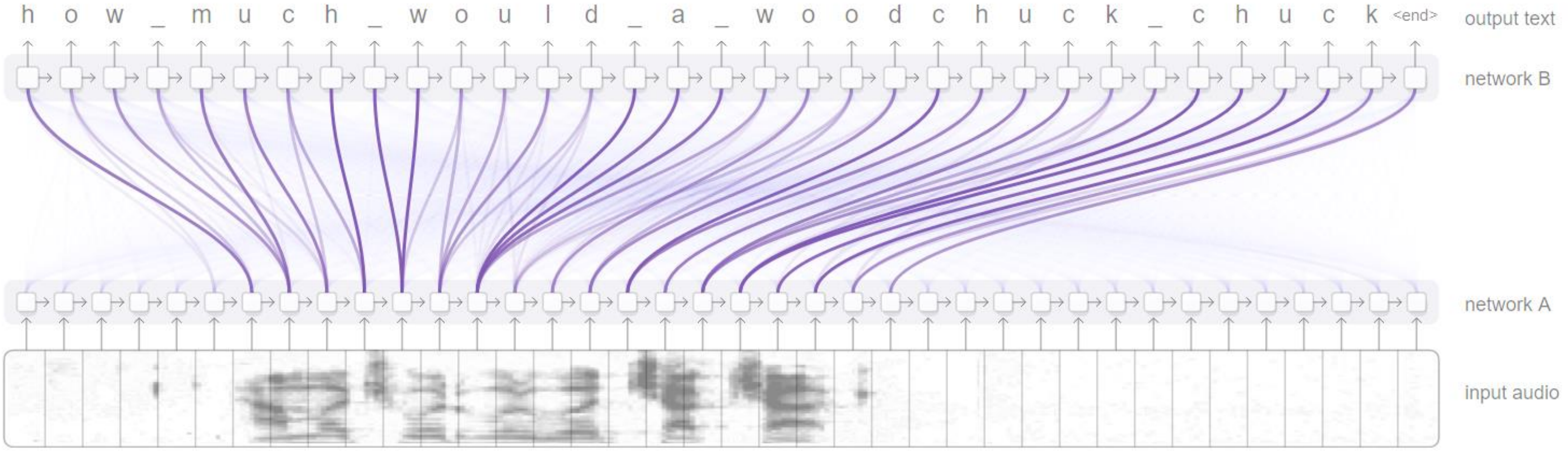


Figure derived from Chan, *et al.* 2015

Attention Mechanism in Image Captioning

A CNN processes the image, extracting high-level features. Then an RNN runs, generating a description of the image based on the features.

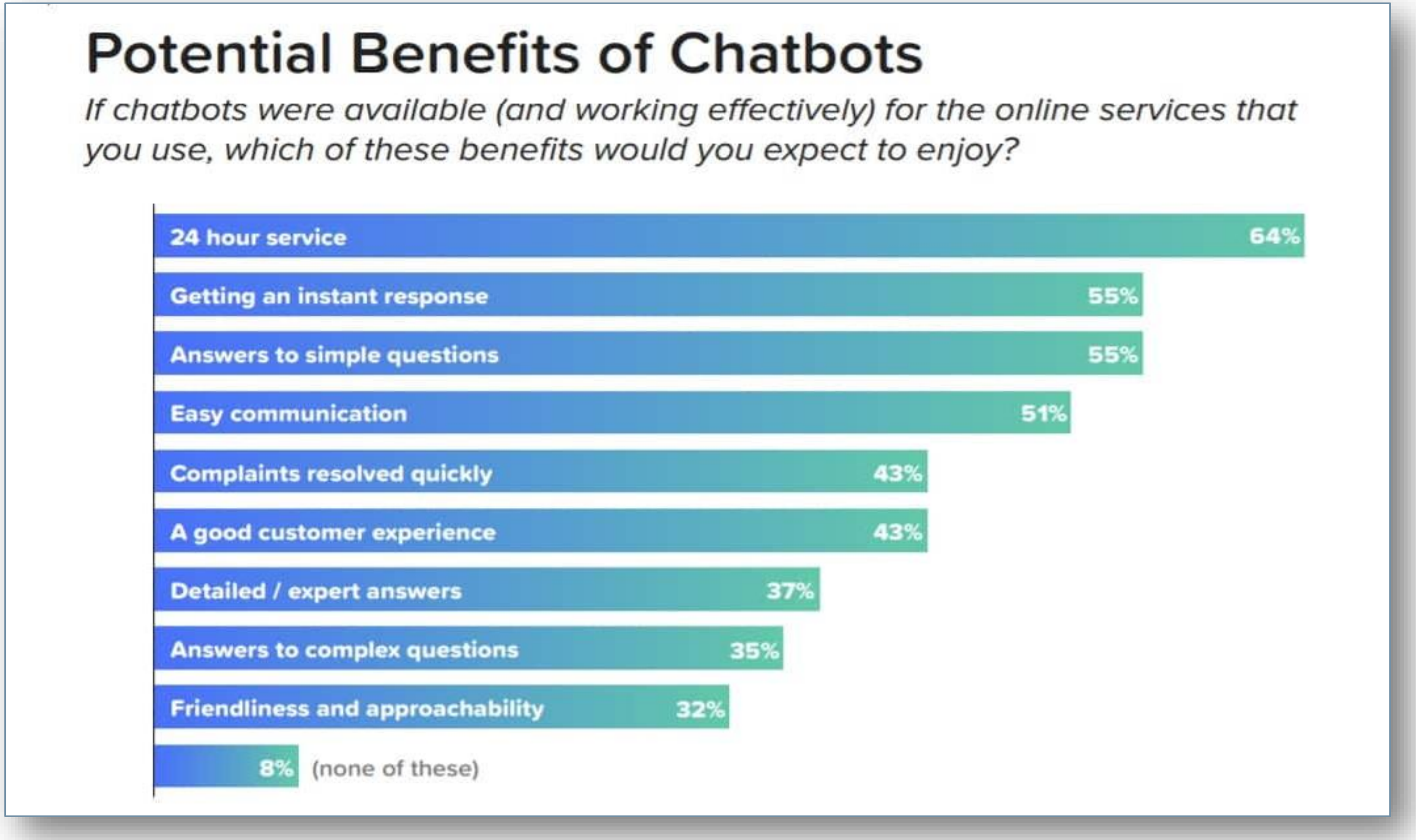
As it generates each word in the description, the RNN focuses on the CNN interpretation of the relevant parts of the image.

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

Attention in Response Generation (i.e., Chatbots)



Sources: *https://blog.appliedai.com/chatbot-benefits/*
https://blog.growthbot.org/chatbots-were-the-next-big-thing-what-happened

Attention in Response Generation (i.e., Chatbots)

Chatbots can be defined along at least two dimensions, *core algorithm* and context handling:

- Generative: encode the question into a context vector and generate the answer word by word using conditioned probability distribution over answer's vocabulary. E.g., an encoder-decoder model.
- Retrieval: rely on knowledge base of question-answer pairs. When a new question comes in, inference phase encodes it in a context vector and by using similarity measure retrieves the top-k neighbor knowledge base items.

Attention in Response Generation (i.e., Chatbots)

Chatbots can be defined along at least two dimensions, core algorithm and *context handling*:

- Single-turn: build the input vector by considering the incoming question. They may lose important information about the history of the conversation and generate irrelevant responses.

$$\{(q_i\,,a_i)\}$$

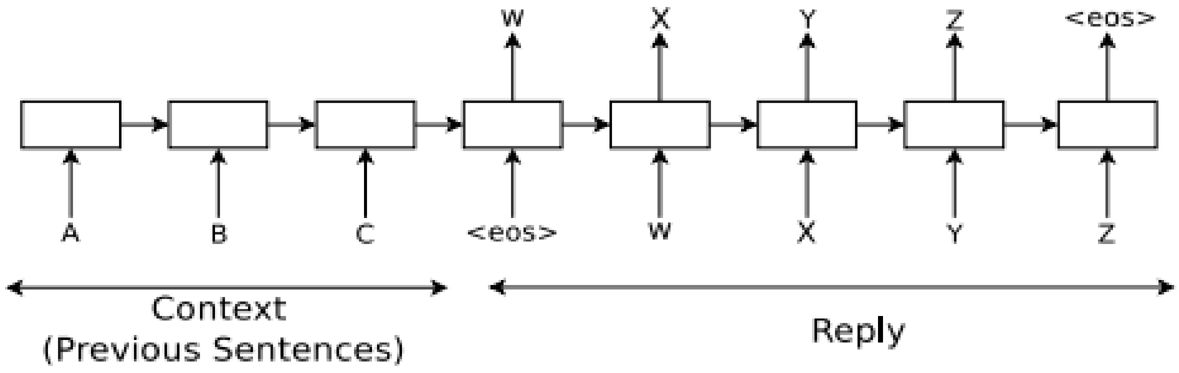
- Multi-turn: the input vector is built by considering a multi-turn conversational context, containing also incoming question.

$$\{([q_{i-2};a_{i-2};q_{i-1};a_{i-1};q_i],a_i)\}$$

Generative Chatbots

Vinyals and Le, 2015 and Shang et al., 2015 proposed to directly apply sequence to sequence models to the conversation between two agents:

- The first person utters "ABC"
- The second person replies "WXYZ"



Generative chatbots use an RNN and train it to map "ABC" to "WXYZ":

- We can borrow the model from machine translation
- A flat model simple and general
- Attention mechanisms apply as usual

How do we handle multi turns chat?

Generative Hierarchical Chatbots

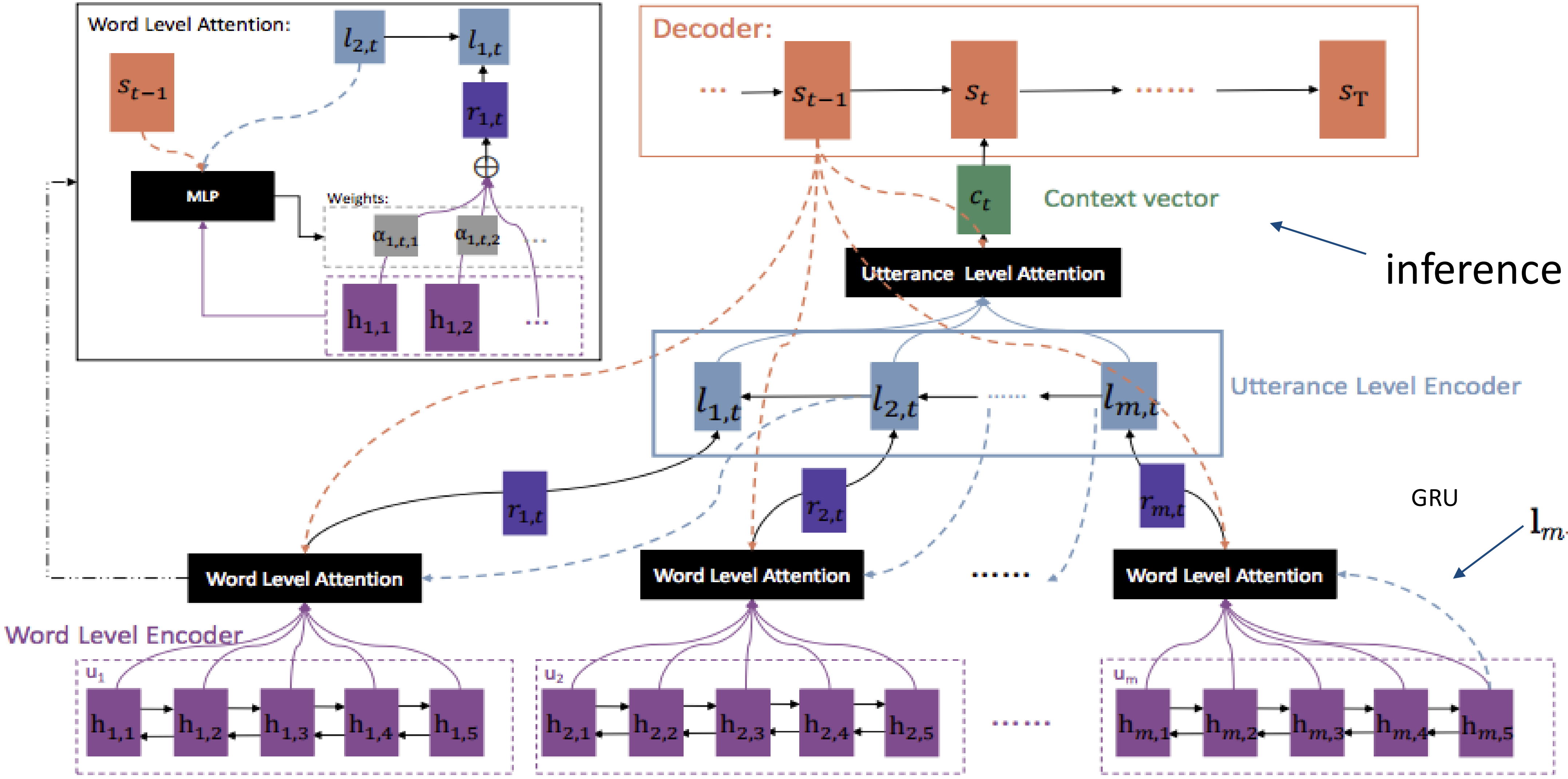
The idea could be concatenating multiple turns into a single long input sequence, but this probably results in poor performances.

- LSTM cells often fail to catch the long term dependencies within input sequences that are longer than 100 tokens
- No explicit representation of turns can be exploited by the attention mechanism

Xing et al., in 2017, extended attention mechanism from single-turn response generation to a hierarchical attention mechanism

- Hierarchical attention networks (e.g., characters -> words -> sentences)
- Generate hidden representation of a sequence from contextualized words

Hierarchical Generative Multi-turn Chatbots



inference !

GRU

$\mathbf{l}_{m+1,t}$

Hierarchical Generative Multi-turn Chatbots

$$\mathbf{U} = (u_1, \ldots, u_m),$$

$$u_i = (w_{i,1}, \ldots, w_{i,T_i})$$

$$\mathbf{h}_{i,k} = concat(\overrightarrow{\mathbf{h}}_{i,k}, \overleftarrow{\mathbf{h}}_{i,k}),$$

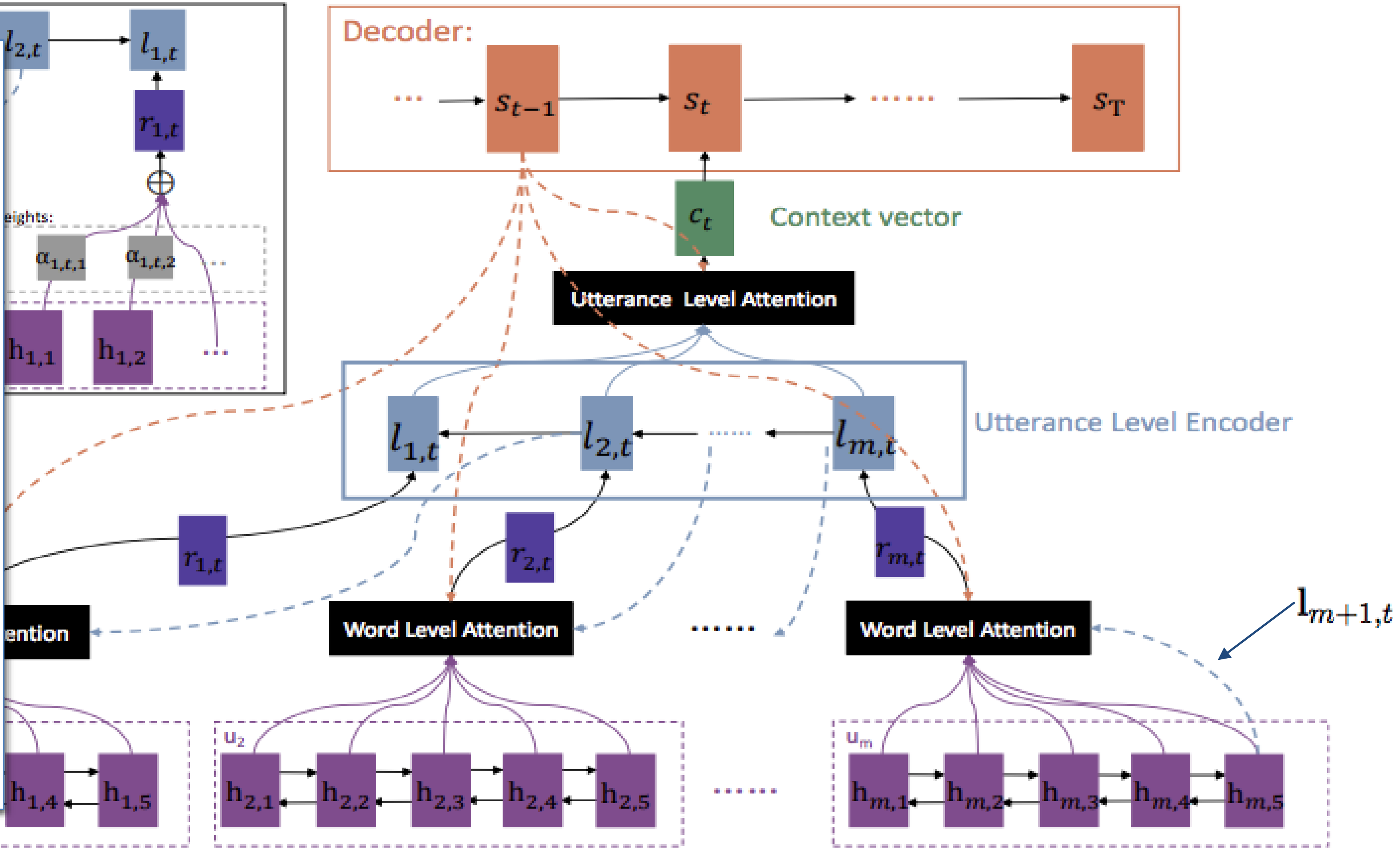
$$\mathbf{r}_{i,t} = \sum_{j=1}^{T_i} \alpha_{i,t,j} \mathbf{h}_{i,j}.$$

$$e_{i,t,j} = \eta(\mathbf{s}_{t-1}, \mathbf{l}_{i+1,t}, \mathbf{h}_{i,j});$$

$$\alpha_{i,t,j} = \frac{exp(e_{i,t,j})}{\sum_{k=1}^{T_i} exp(e_{i,t,k})},$$

$$(\mathbf{l}_{1,t}, \ldots, \mathbf{l}_{m,t})$$

$$\mathbf{c}_t = \sum_{i=1}^{m} \beta_{i,t} \mathbf{l}_{i,t}.$$



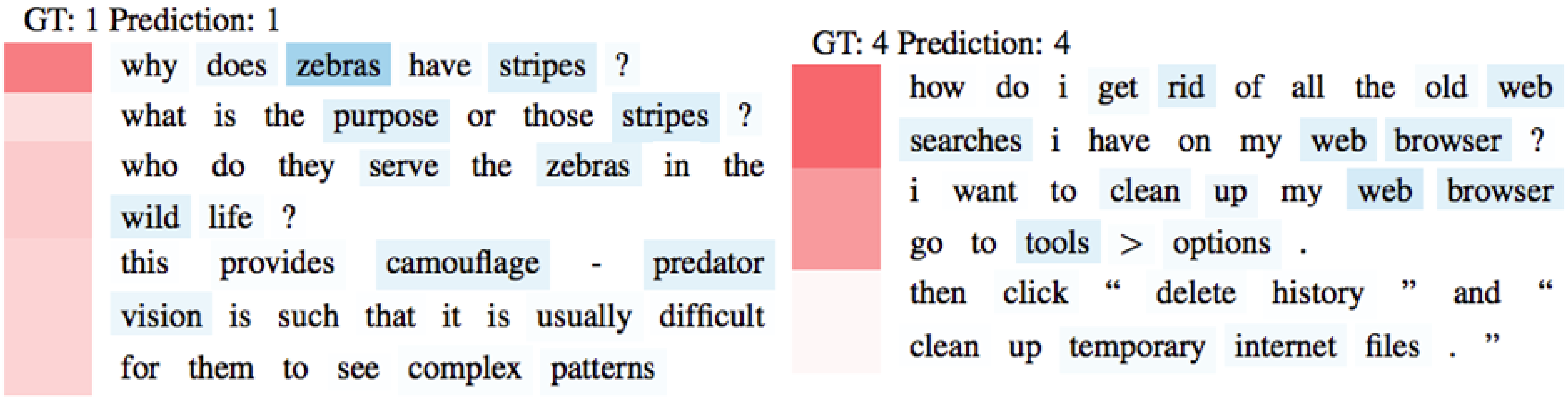
Hierarchical Generative Multi-turn Chatbots

We can visualize hierarchical attention weights, darker color means more important words or utterances.

Hierarchical Document Classification

Hierarchical attention networks have been used for topic classification (e.g., Yahoo Answer data set).

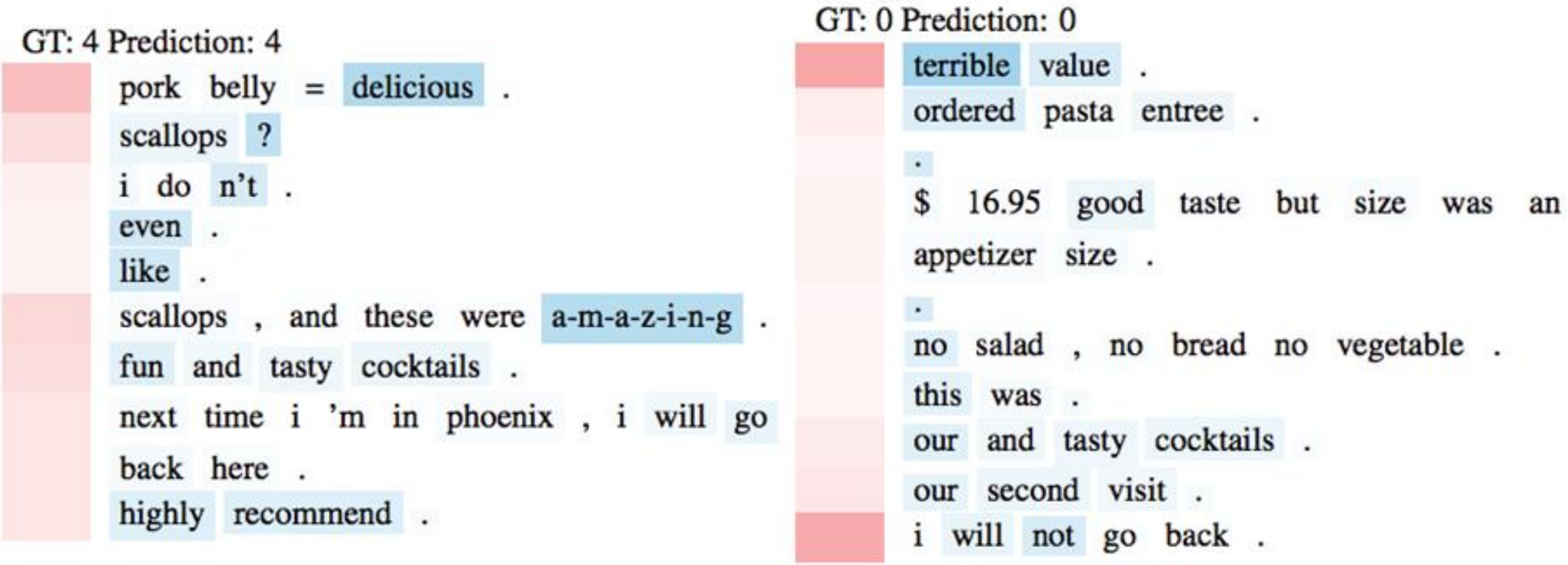
- Left document denotes Science and Mathematics; model accurately localizes the words *zebra*, *stripes*, *camouflage*, *predator* and corresponding sentences.
- Right document denotes Computers and Internet; the model focuses on *web*, *searches*, *browsers* and their corresponding sentences.



Hierarchical Document Classification

In Sentiment Analysis, the model can select words carrying strong sentiment like ***delicious***, ***amazing***, ***terrible*** and corresponding sentences.

Sentences containing useless words like cocktails, pasta, entree are disregarded.



Attention is all you need!

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*,†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*,‡]
illia.polosukhin@gmail.com

NIPS 2017

Attention is all you need!

Having seen attention is what makes things working you start wondering:

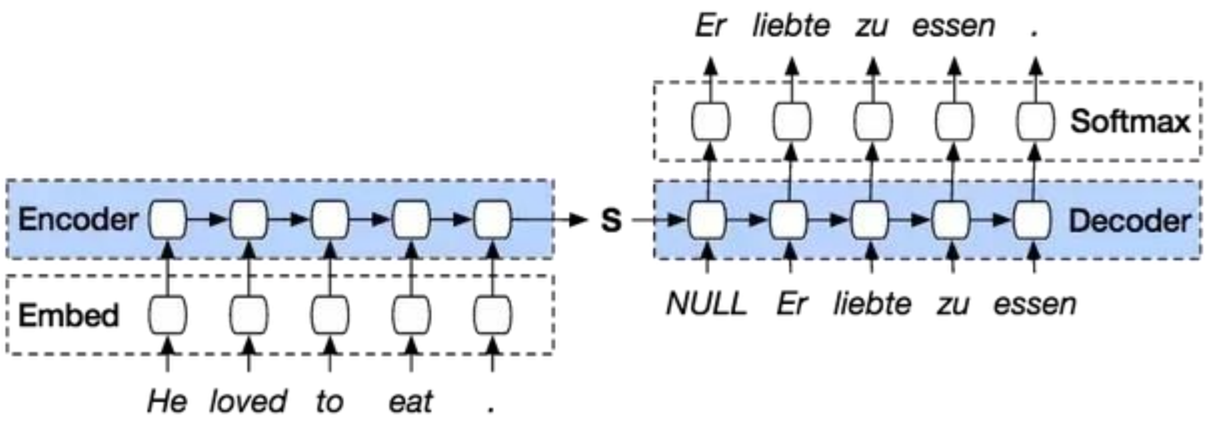
- Sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples.
- Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks. Can we base solely on attention mechanisms, dispensing with recurrence and convolutions entirely?
- Without recurrence, nor convolution, in order for the model to make use of the order of the sequence, we must **inject** some information about the relative or absolute position of the tokens in the sequence.

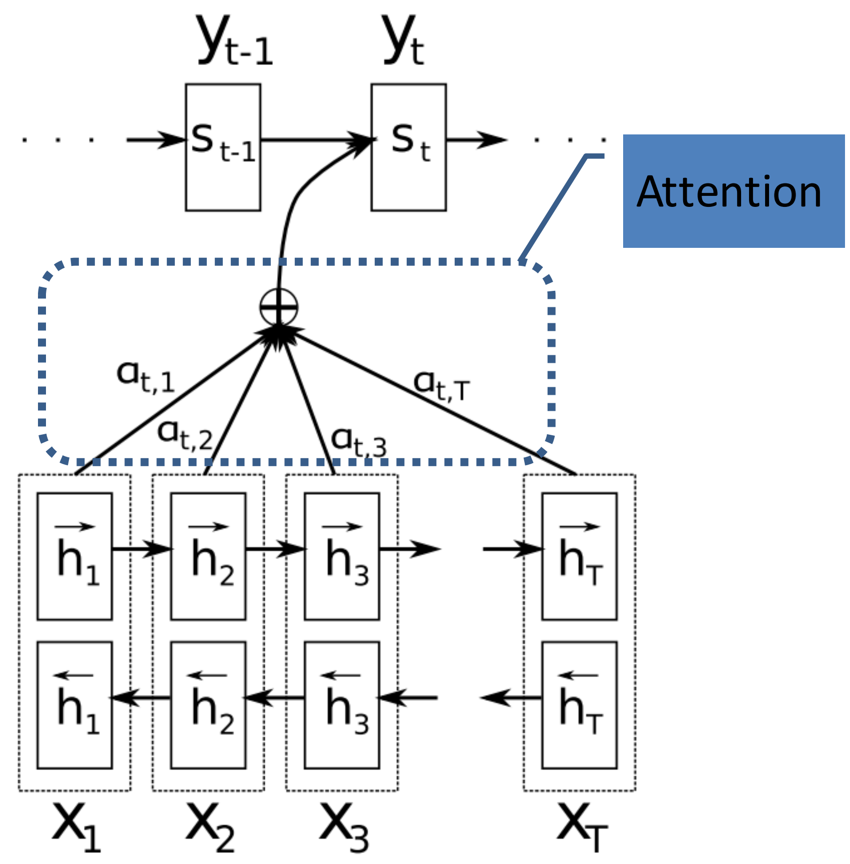
Current State of the Art

There has been a running joke in the NLP community that **an LSTM with attention** will yield state-of-the-art performance on any task.

Attention is built upon RNN …

Example: Neural Machine Translation





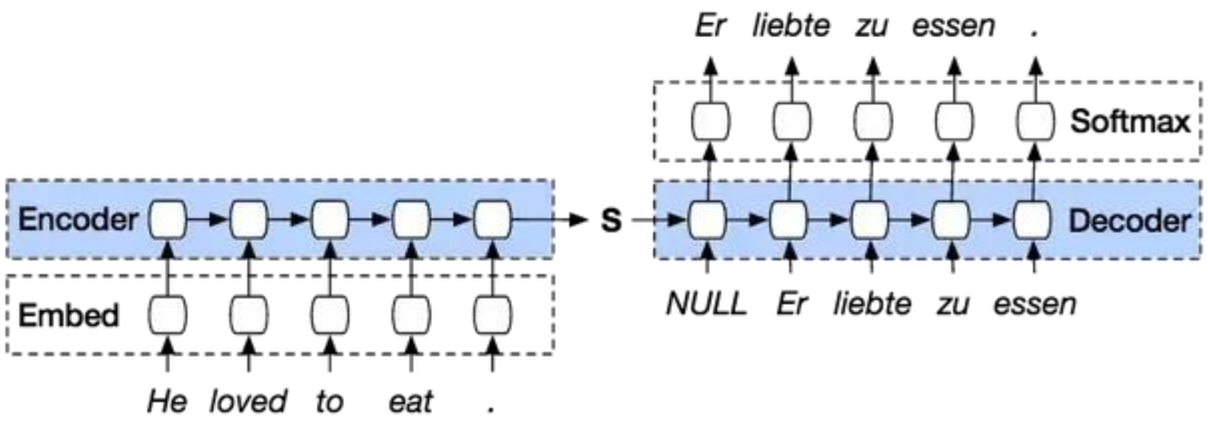
Current State of the Art

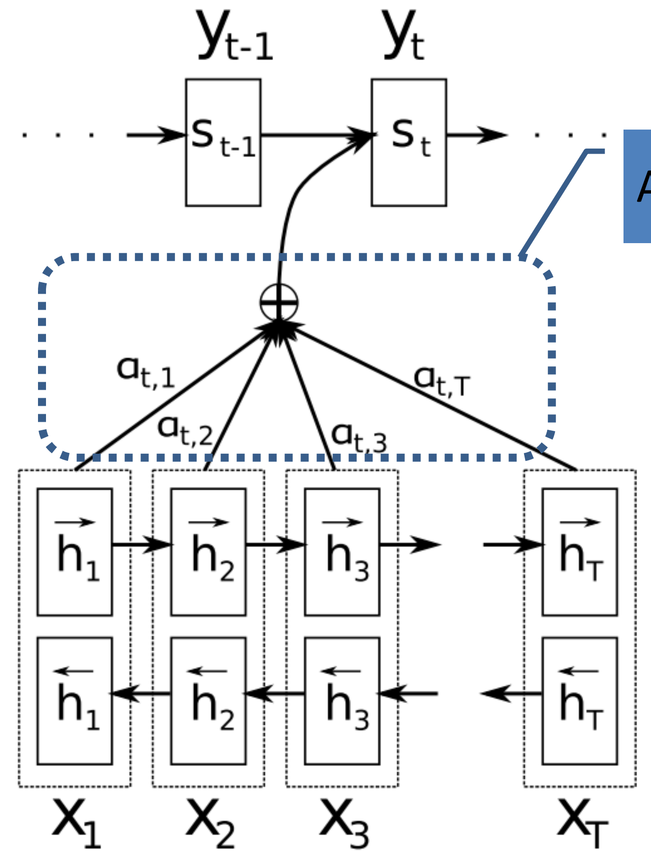
There has been a running joke in the NLP community that **an LSTM with attention** will yield state-of-the-art performance on any task.

<u>Attention is built upon RNN …</u>

The Transformer breaks this assumption!

Example: Neura



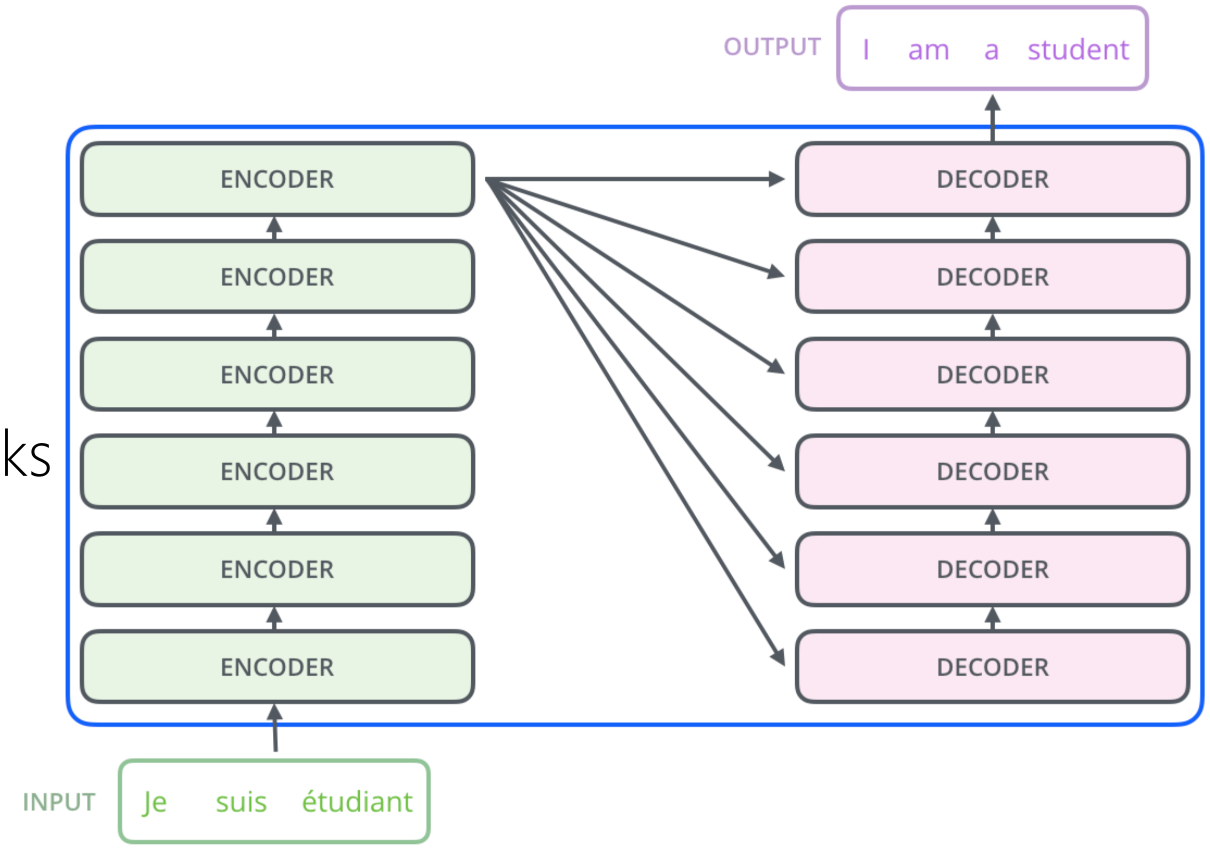


Attention

Transformer

A Transformer model is make out of:

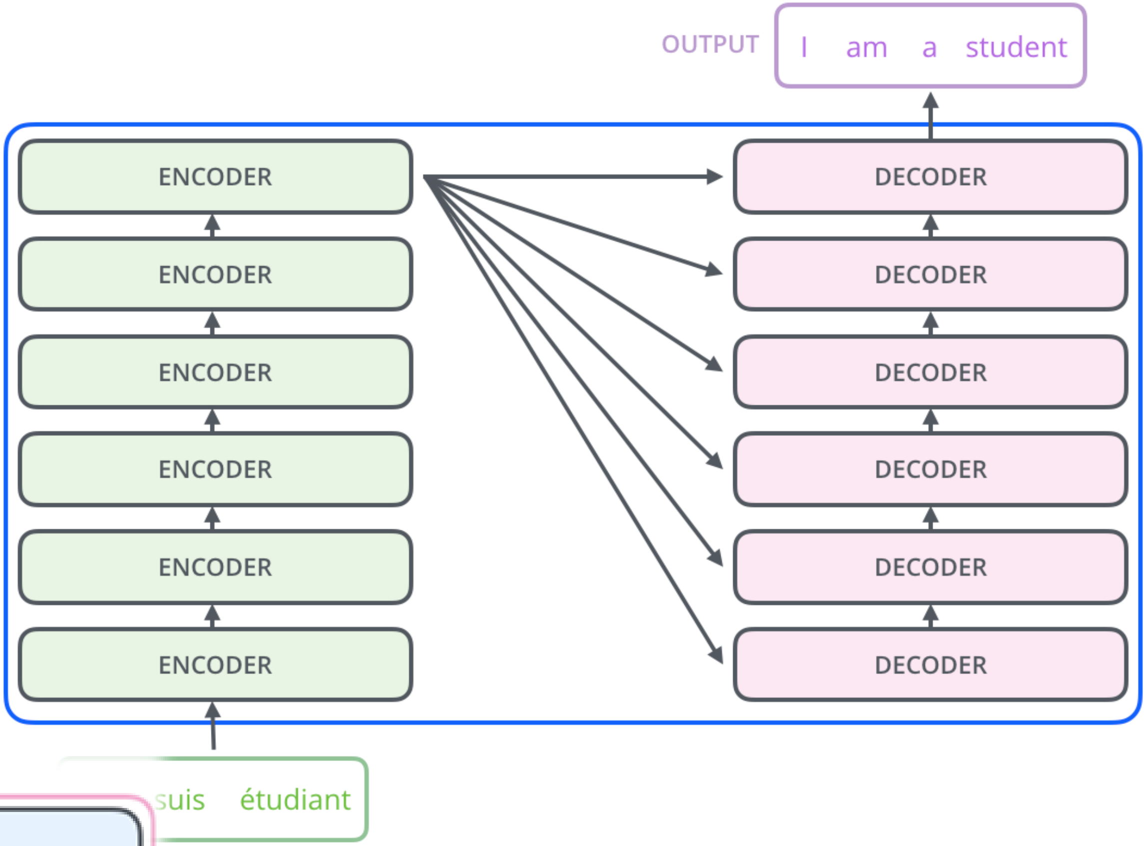
- Scaled Dot-Product Attention
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding

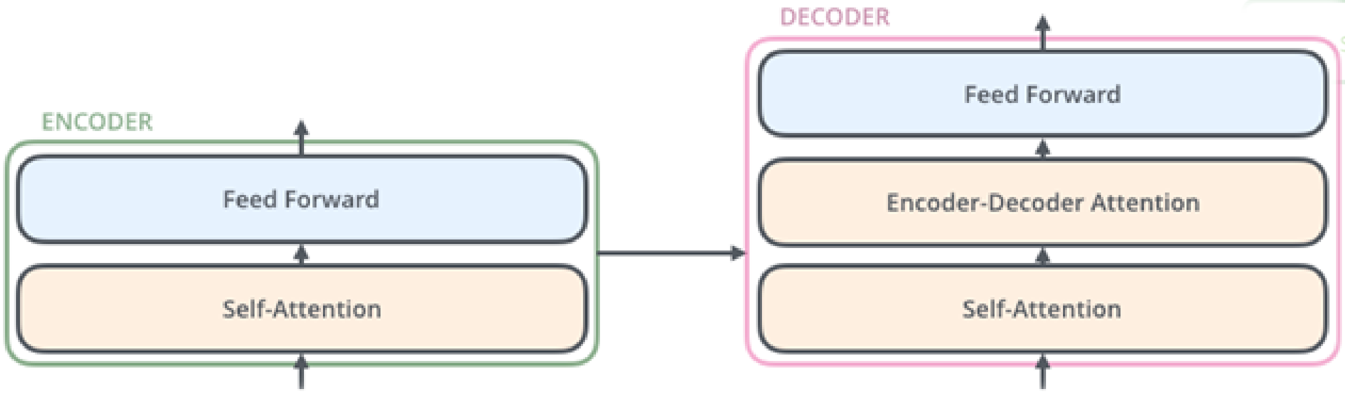


Transformer

A Transformer model is make out of:

- Scaled Dot-Product Attention
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding

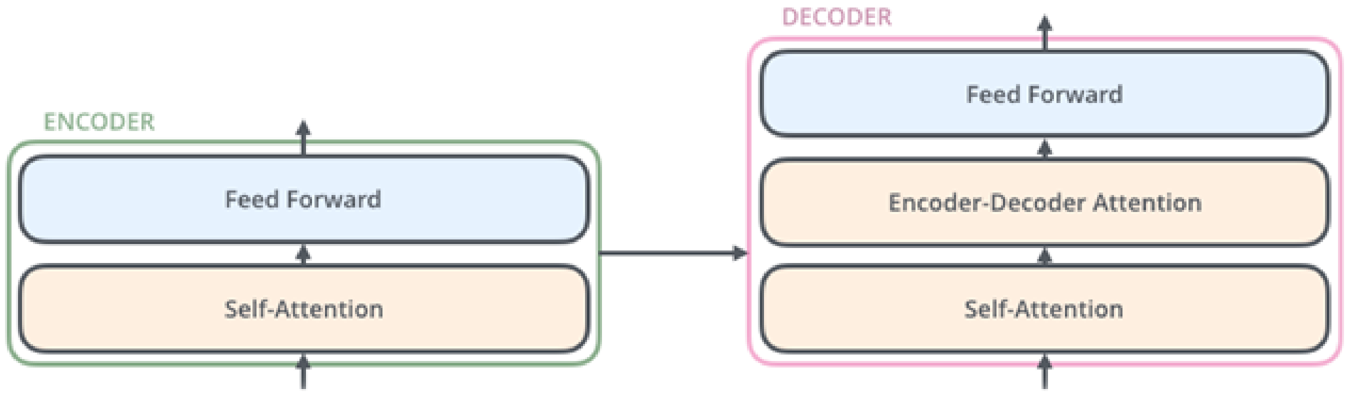




Transformer

A Transformer model is made out of:

- Scaled Dot-Product Attention
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding



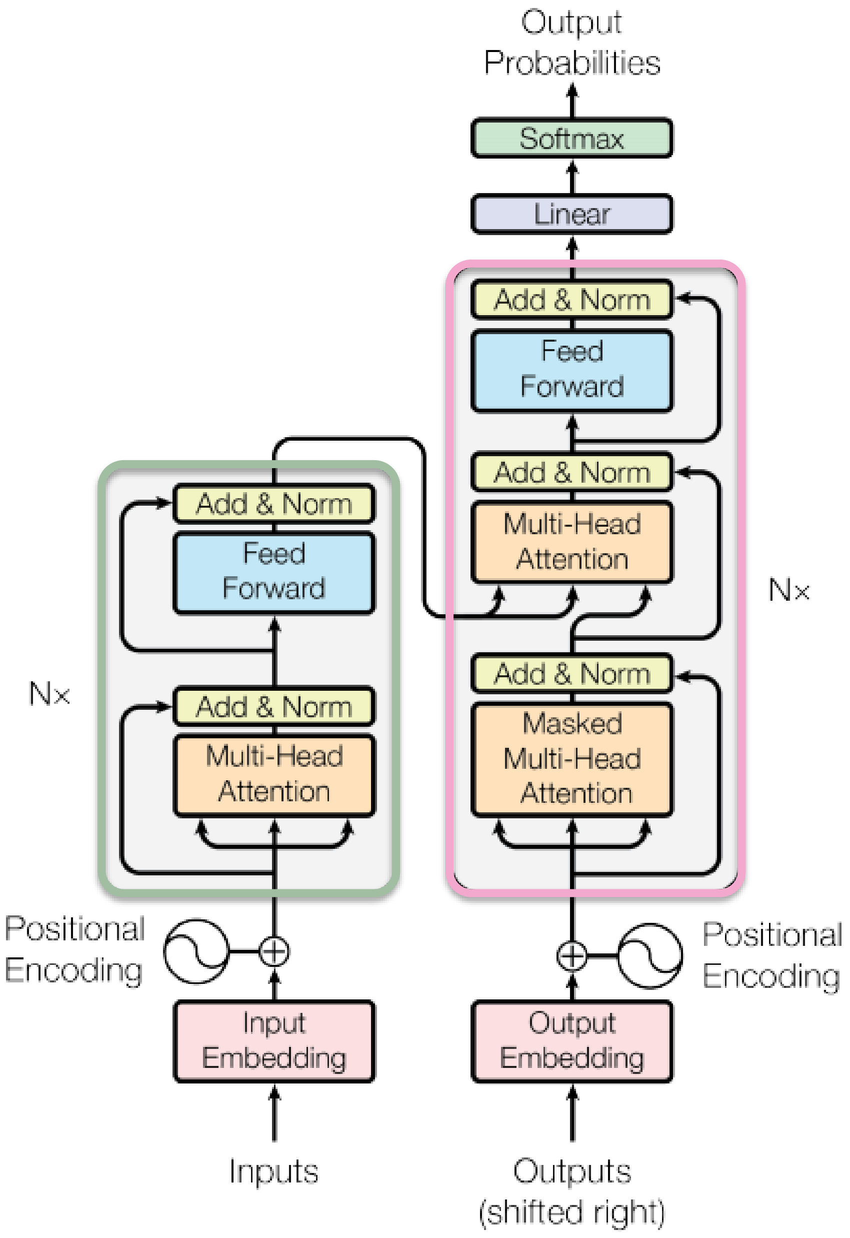


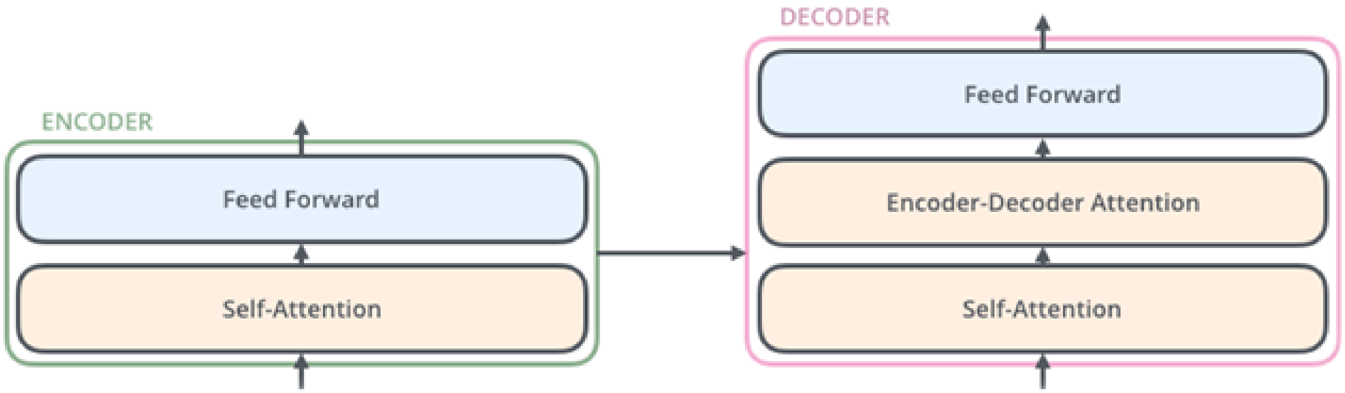
Figure 1: The Transformer - model architecture.

Transformer

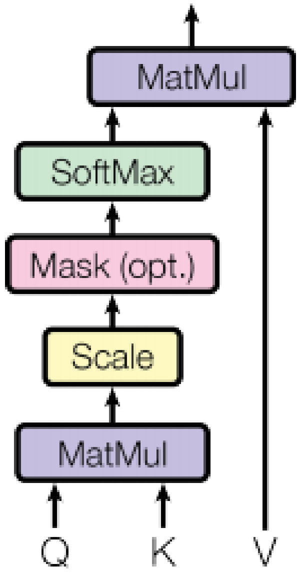
A Transformer model is made out of:

- Scaled Dot-Product Attention
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding

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



Scaled Dot-Product Attention

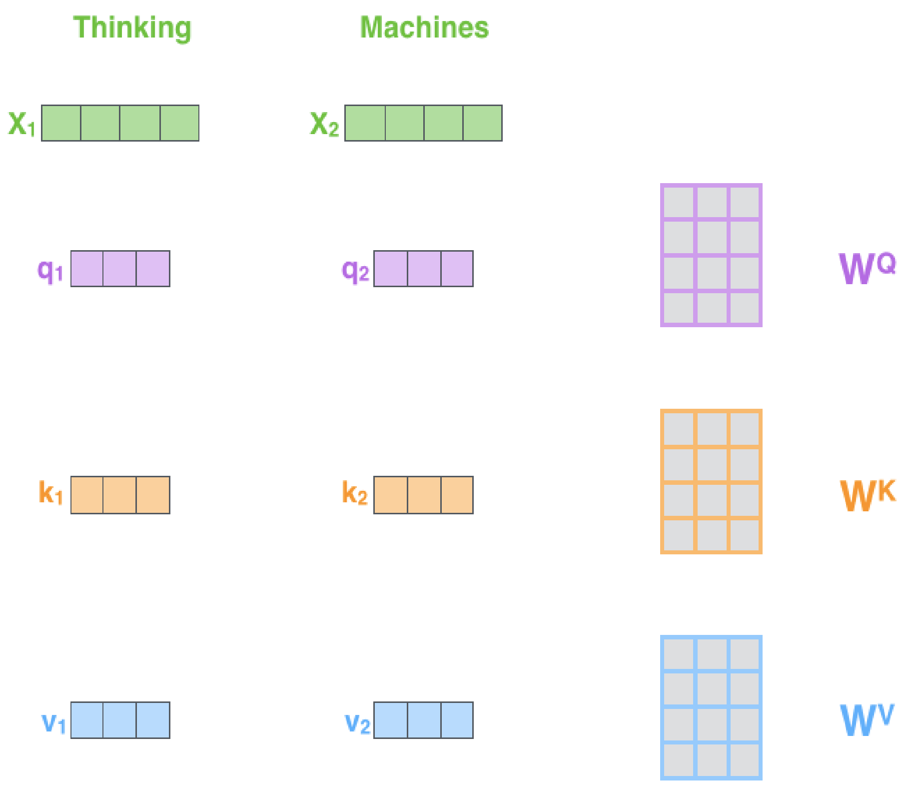


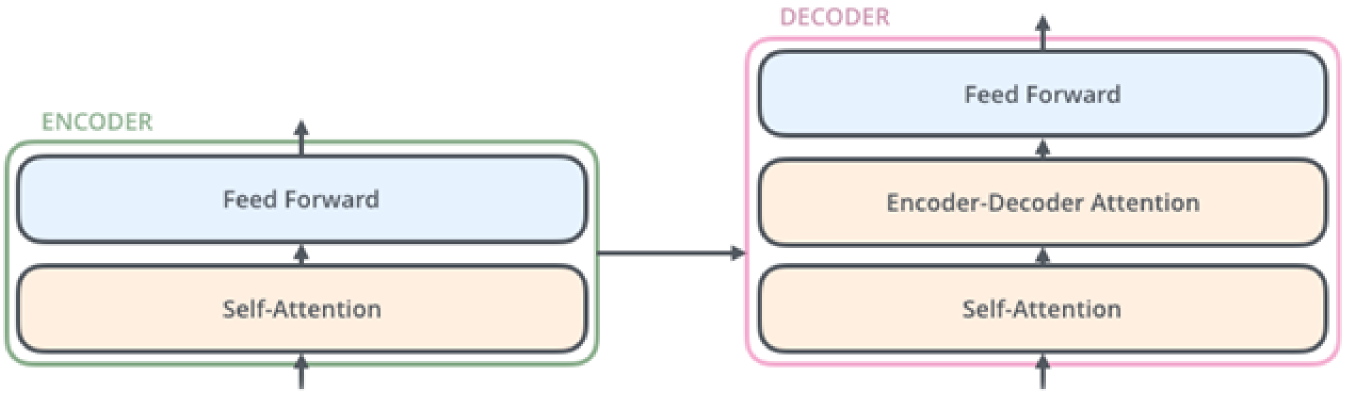
Transformer

A Transformer model is made out of:

- <u>Scaled Dot-Product Attention</u>
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding

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



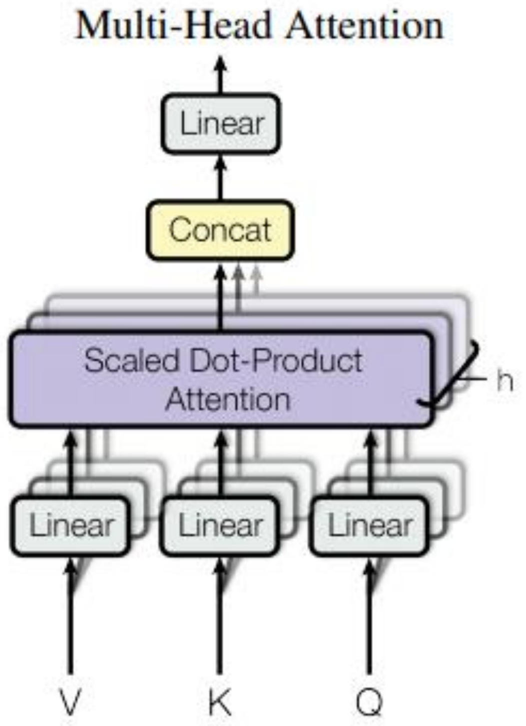


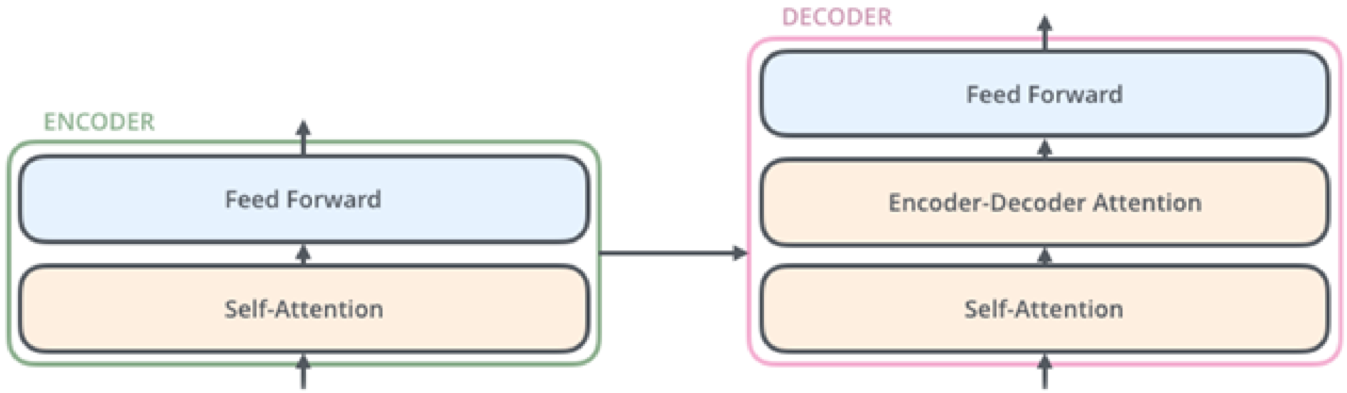
Transformer

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

A Transformer model is made out of:

- Scaled Dot-Product Attention
- <u>Multi-Head Attention</u>
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding



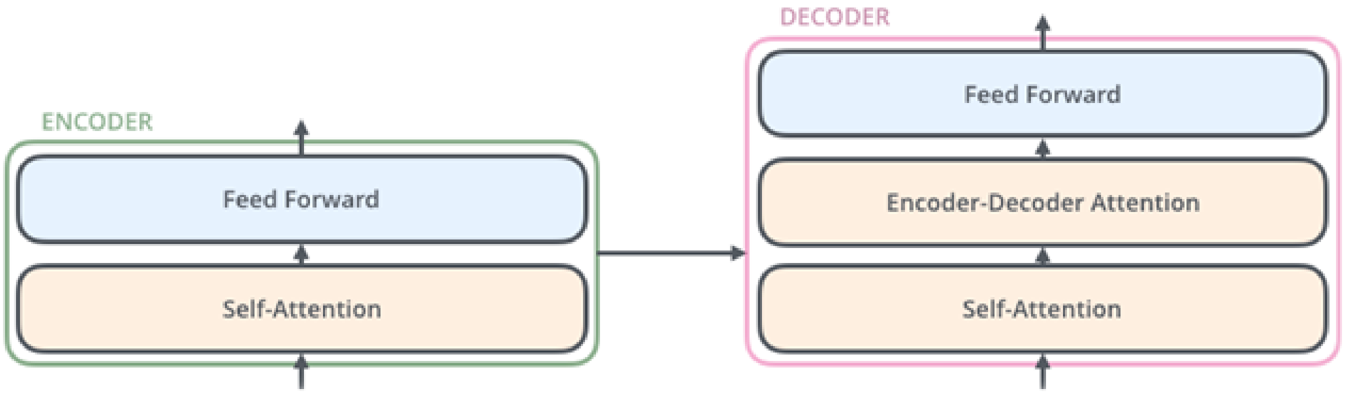


Transformer

A Transformer model is made out of:

- Scaled Dot-Product Attention
- Multi-Head Attention
- <u>Position-wise Feed-Forward Networks</u>
- Embeddings and Softmax
- Positional Encoding

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



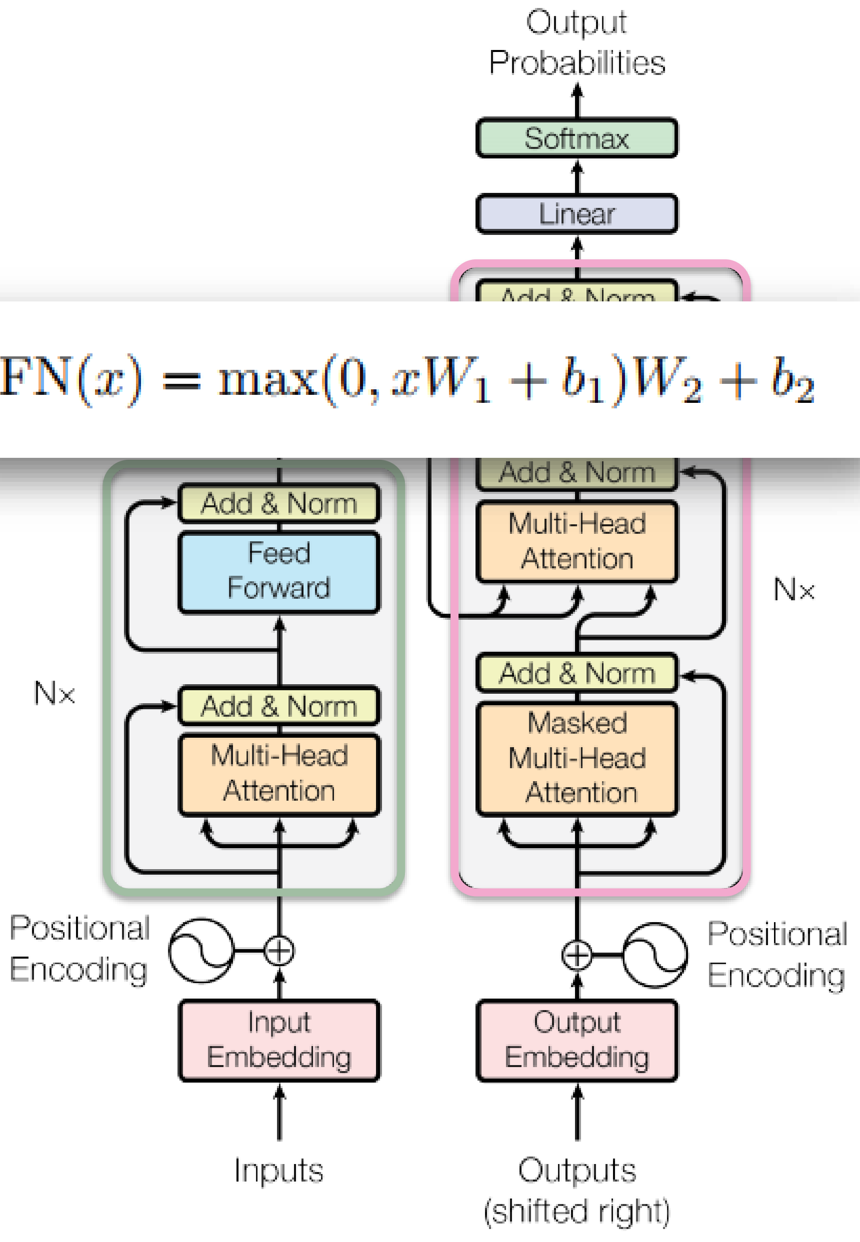
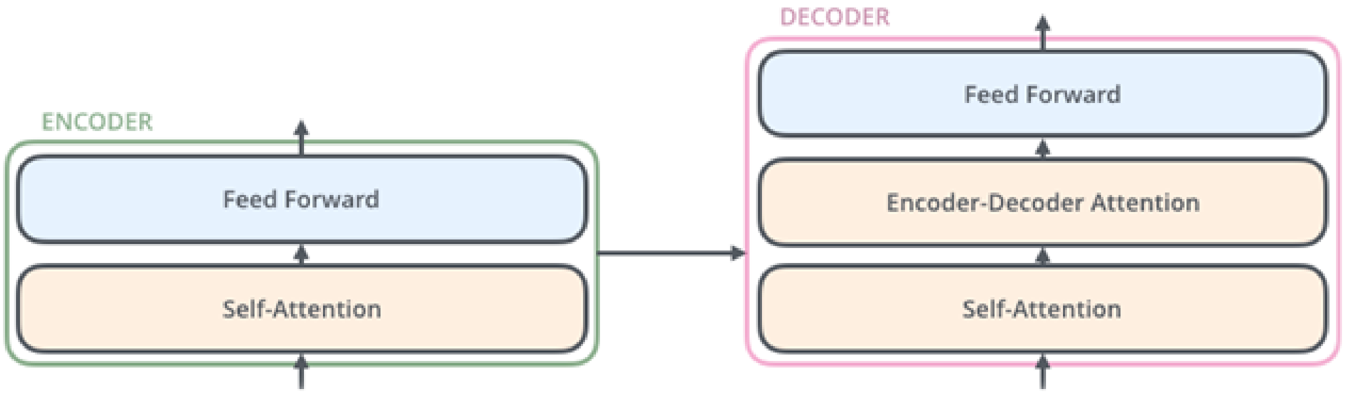


Figure 1: The Transformer - model architecture.

Transformer

A Transformer model is made out of:

- Scaled Dot-Product Attention
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding



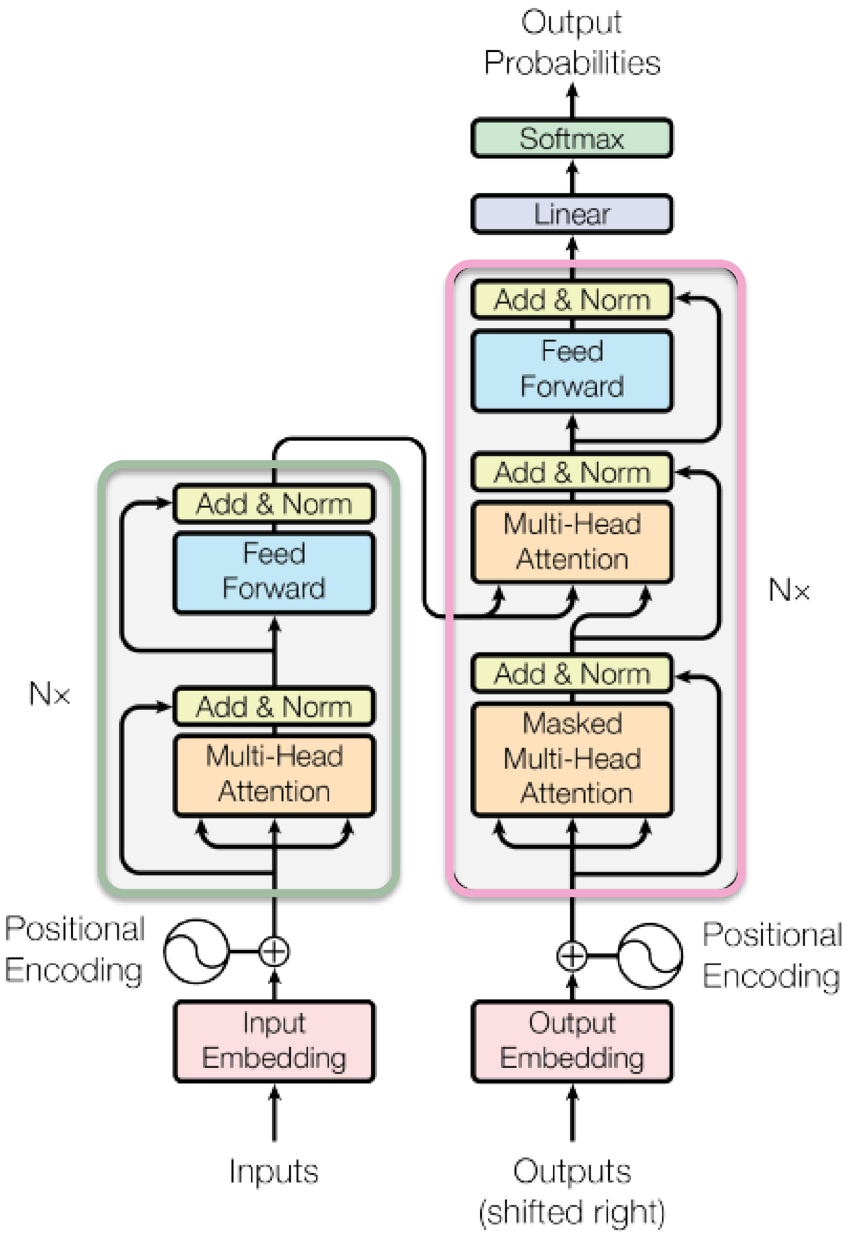


Figure 1: The Transformer - model architecture.

Transformer

A Transformer model is made out of:

- Scaled Dot-Product Attention
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding

Reason: no RNN to model the sequence position

Two types:

- learned positional embeddings (arXiv:1705.03122v2)
- Sinusoid:

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

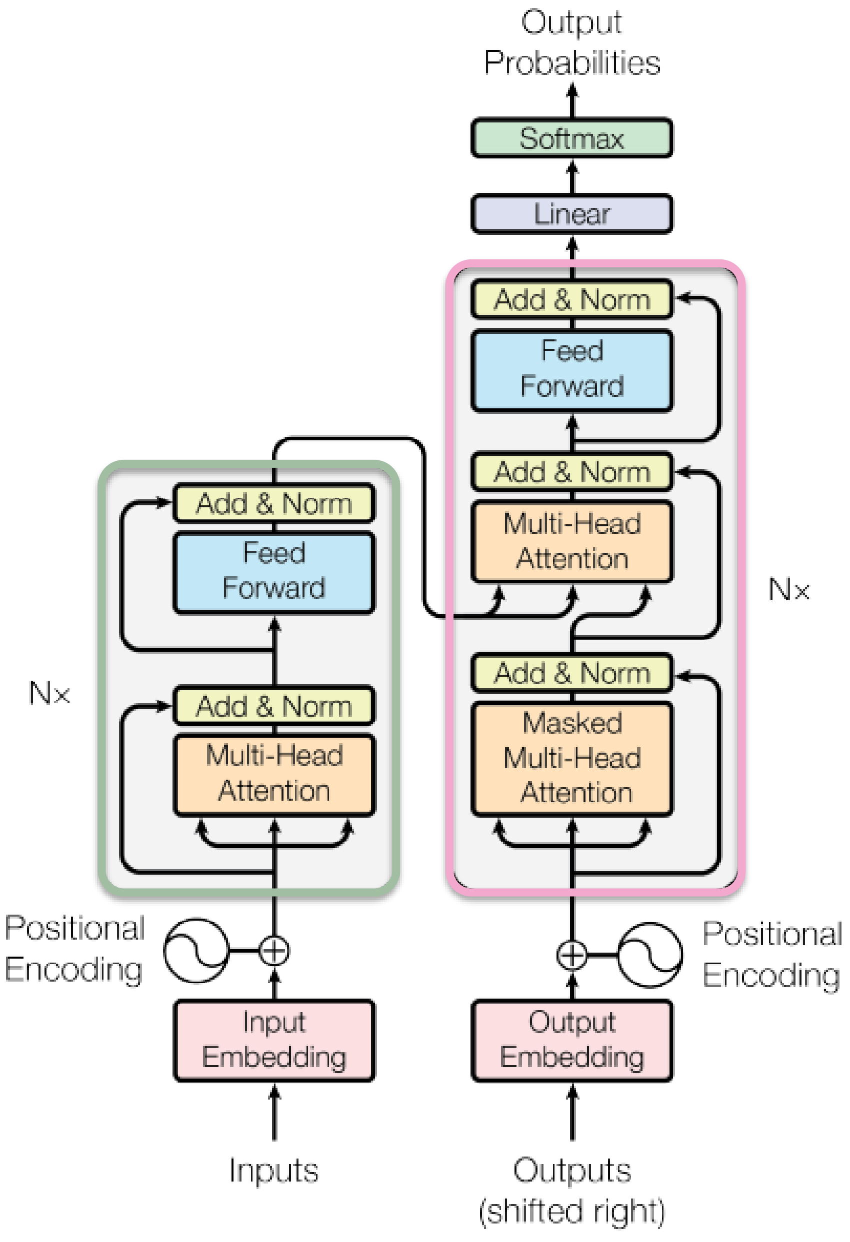


Figure 1: The Transformer - model architecture.

Transformer Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Observations:

- Self-Attention has O(1) maximum path length (capture long range dependency easily)
- When n<d, Self-Attention has lower complexity per layer

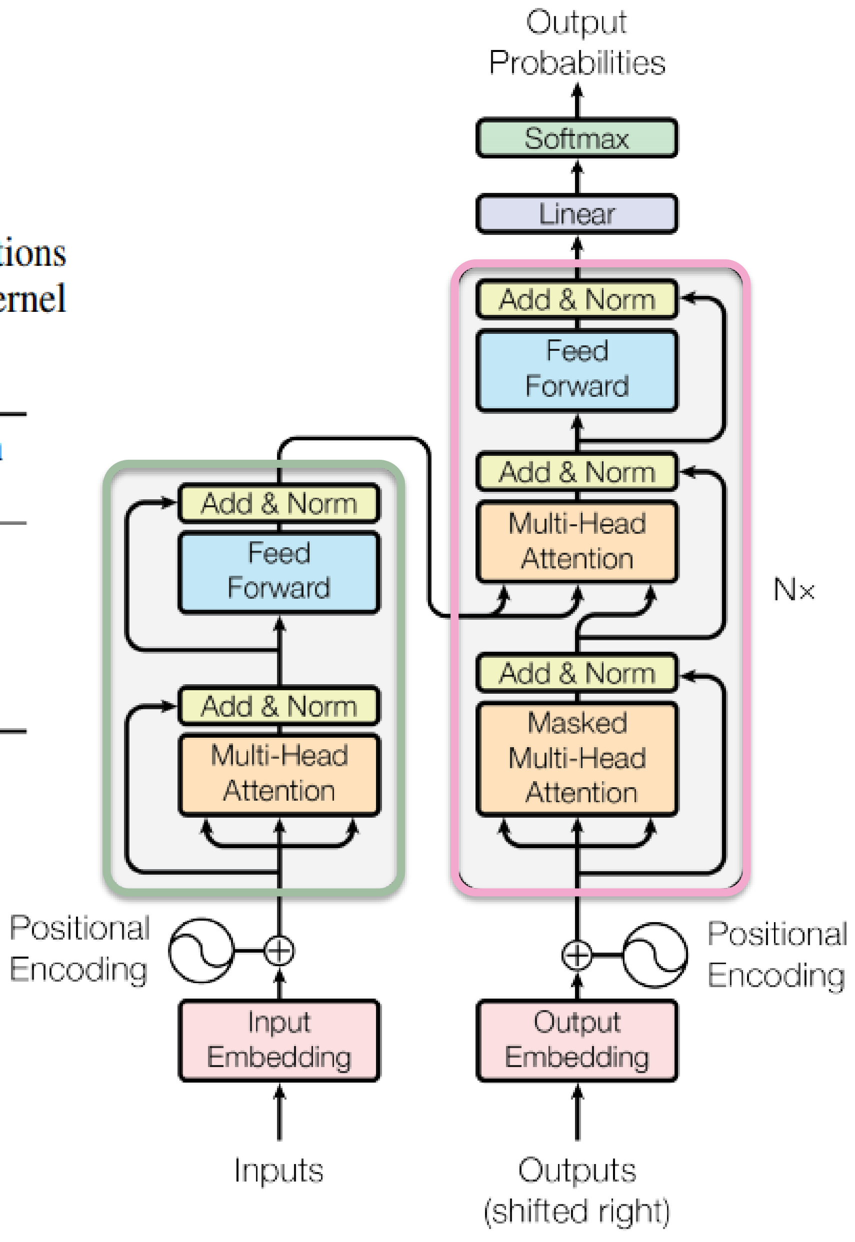


Figure 1: The Transformer - model architecture.

Transformer Performance

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 [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	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 [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

- Eng-to-De: new state-of-the-art
- Eng-to-Fr: new single-model state-of-the-art
- Less training cost

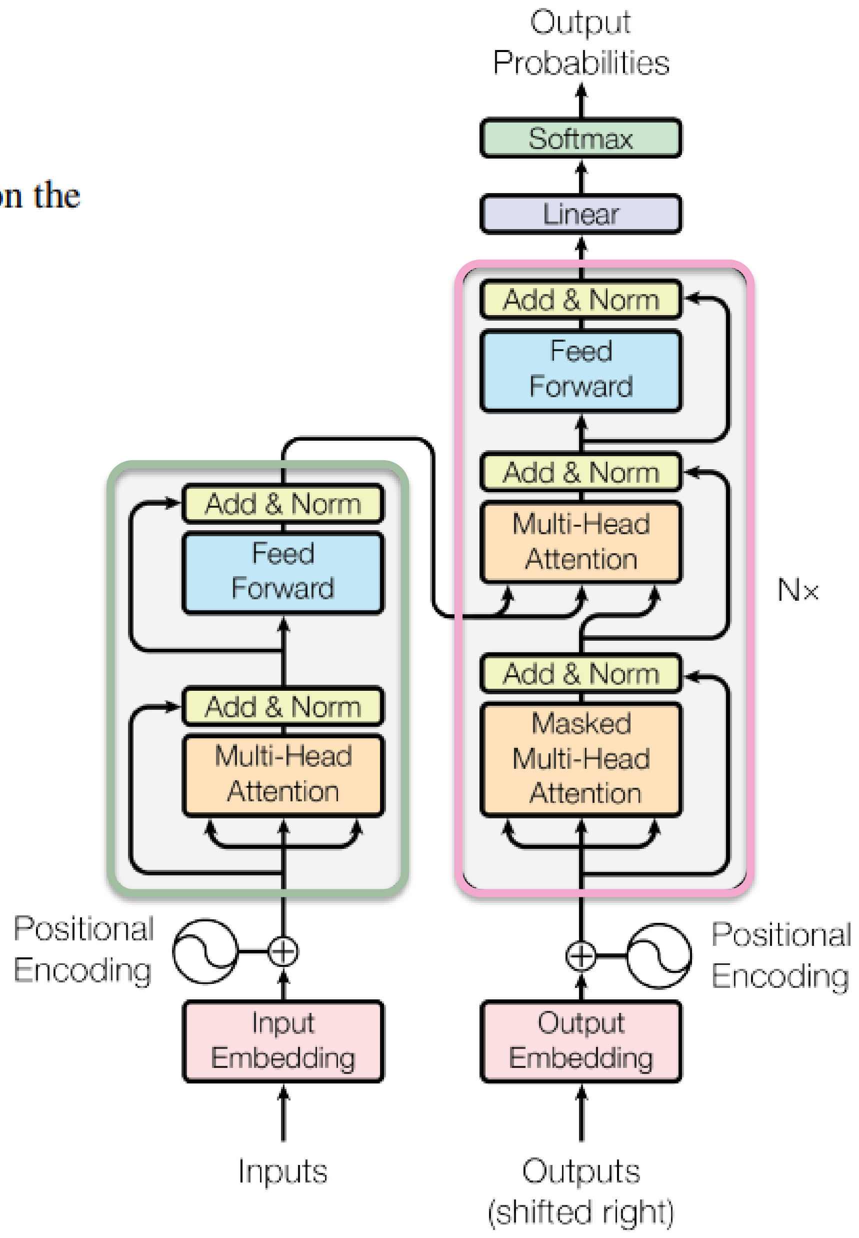


Figure 1: The Transformer - model architecture.

Transformer Performance

- source: Aber ich habe es nicht hingekriegt
- expected: But I didn't handle it
- got: But I didn't <UNK> it

- source: Wir könnten zum Mars fliegen wenn wir wollen
- expected: We could go to Mars if we want
- got: We could fly to Mars when we want

- source: Dies ist nicht meine Meinung Das sind Fakten
- expected: This is not my opinion These are the facts
- got: This is not my opinion These are facts

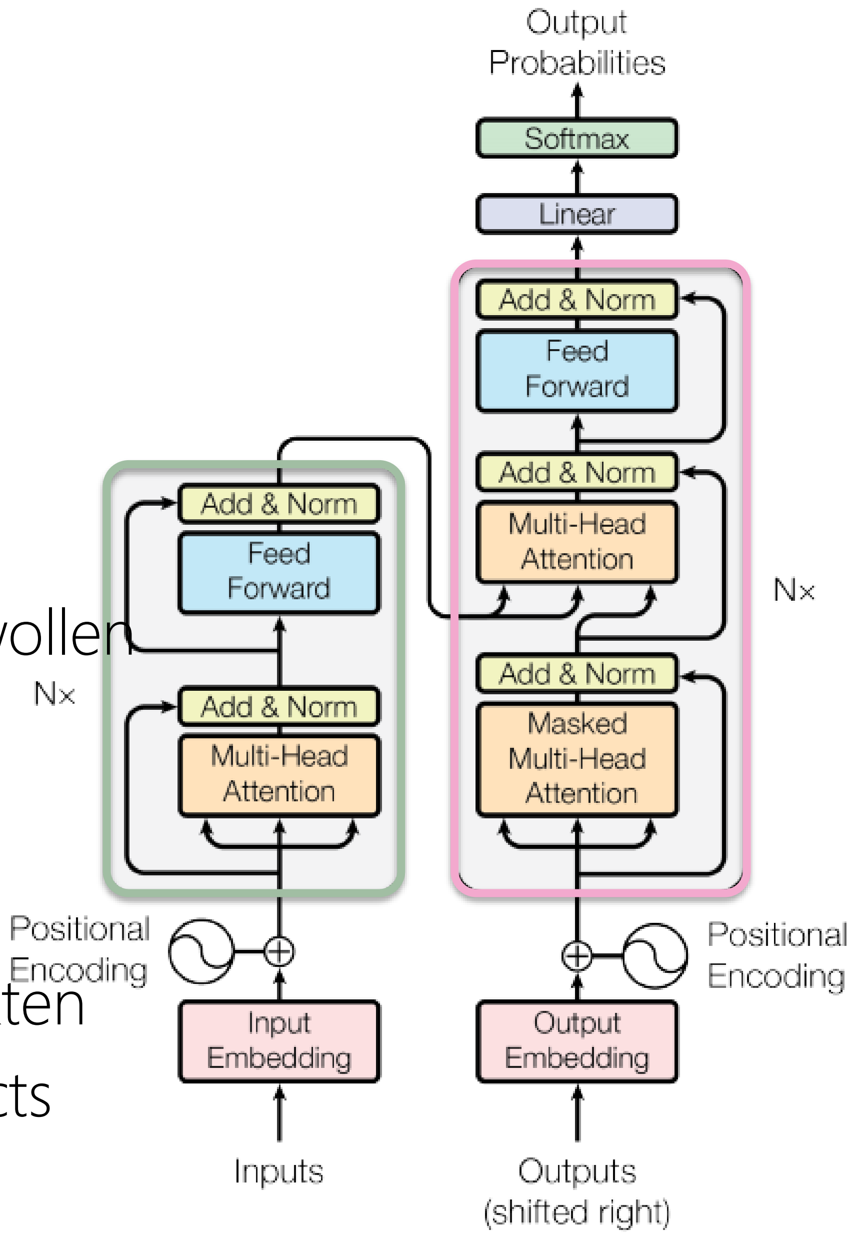


Figure 1: The Transformer - model architecture.

Acknowledgements

These slides are highly based on material taken from the following websites/blogs:

- https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/
- https://medium.com/@Aj.Cheng/seq2seq-18a0730d1d77
- https://distill.pub/2016/augmented-rnns/
- http://jalammar.github.io/illustrated-transformer/

Amazing images, and part of content, about attention mechanisms from

Olah & Carter, "Attention and Augmented Recurrent Neural Networks", Distill, 2016. http://doi.org/10.23915/distill.00001