POLITECNICO MILANO 1863

Advances in Deep Learning with Applications in Text and Image Processing

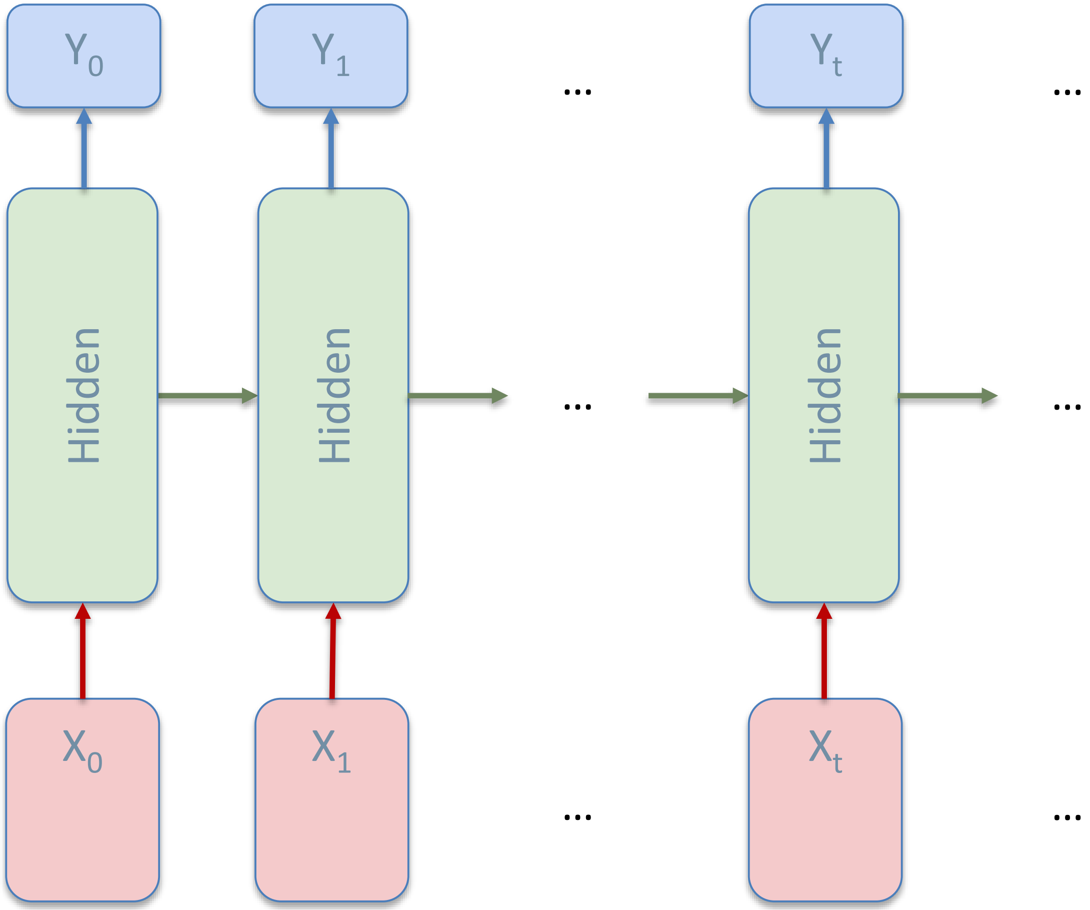
- Seq2Seq Learning Architectures -

Prof. Matteo Matteucci – *matteo.matteucci@polimi.it*

Department of Electronics, Information and Bioengineering
Artificial Intelligence and Robotics Lab - Politecnico di Milano

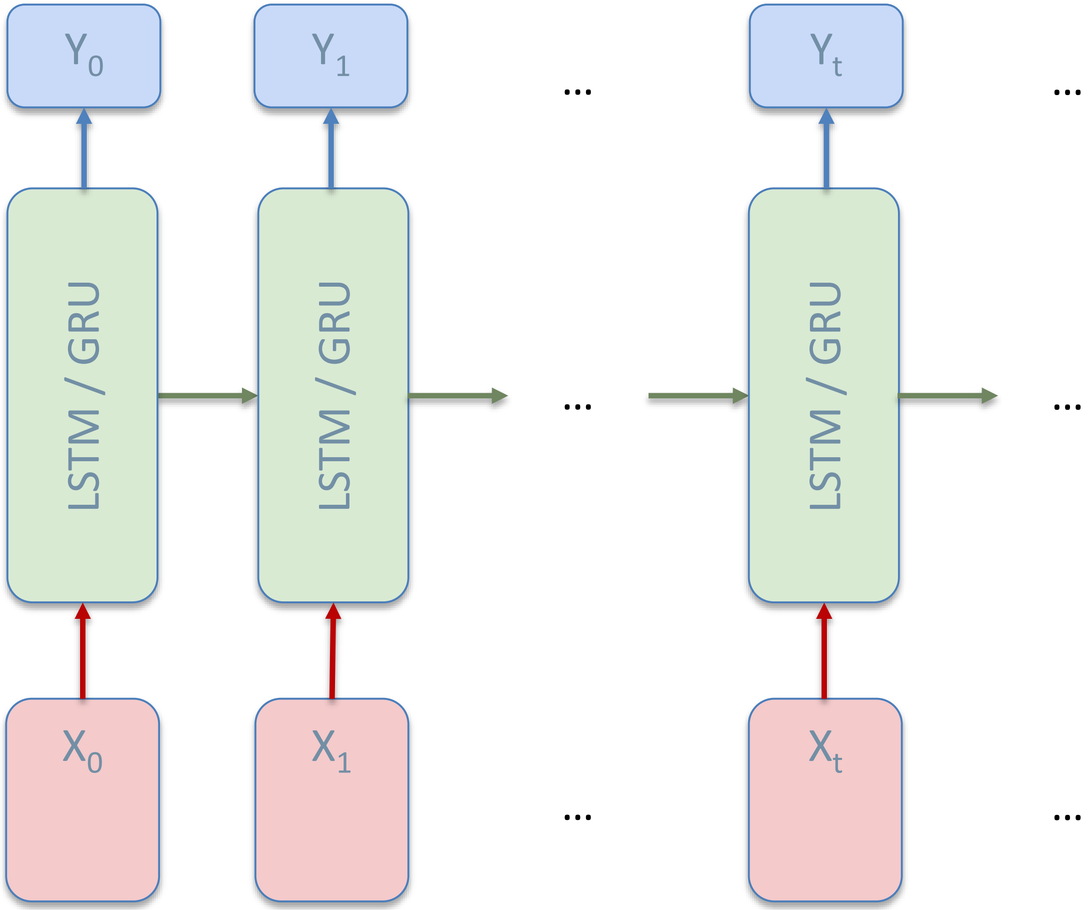
Let’s Recall LSTM Networks

From feed forward architecture to recurrent one

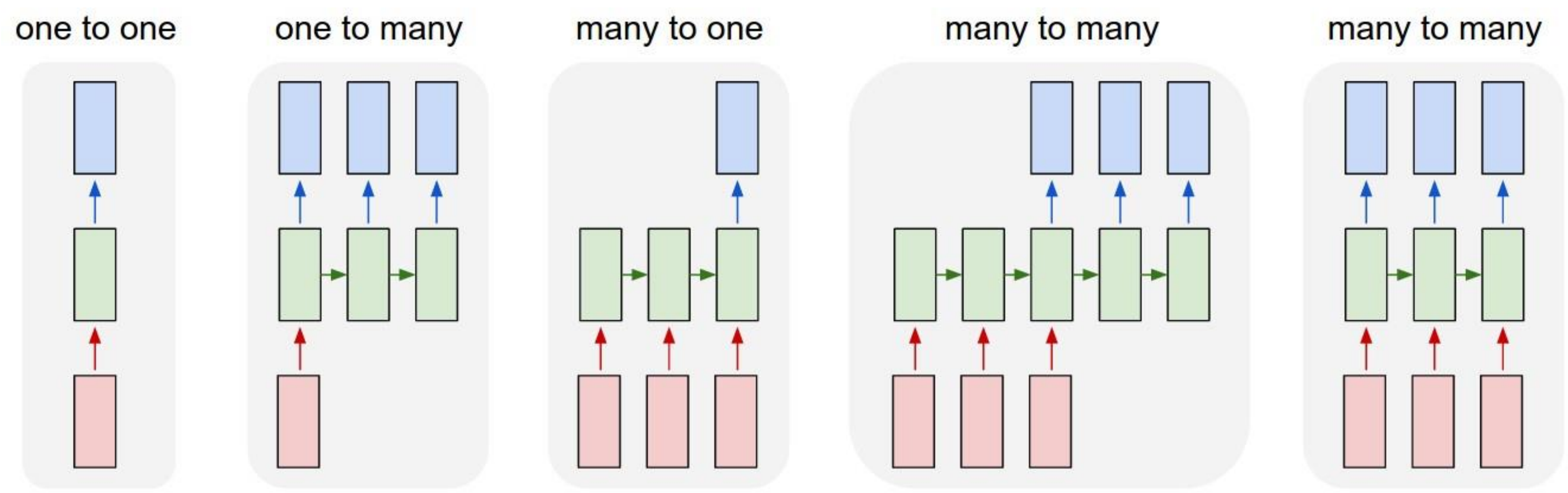


Let's Recall LSTM Networks

From feed forward architecture to recurrent one



Sequential Data Problems



Fixed-sized input to fixed-sized output (e.g. image classification)

Sequence output (e.g. image captioning takes an image and outputs a sentence of words).

Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

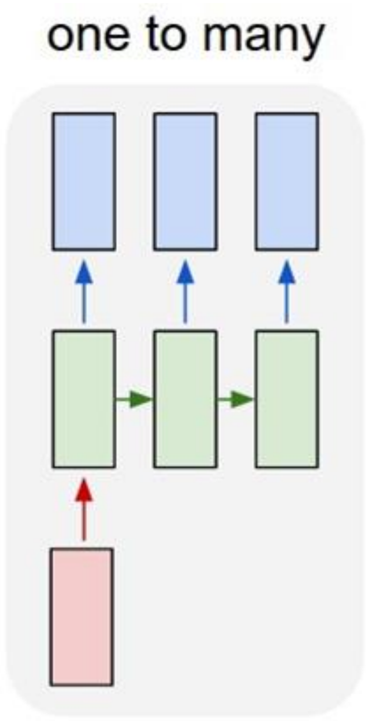
Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French)

Synced sequence input and output (e.g. video classification where we wish to label each frame of the video)

Credits: Andrej Karpathy

Sequence to Sequence Learning Examples (1/3)

Image Captioning: input a single image and get a series or sequence of words as output which describe it. The image has a fixed size, but the output has varying length.



A person riding a motorcycle on a dirt road.

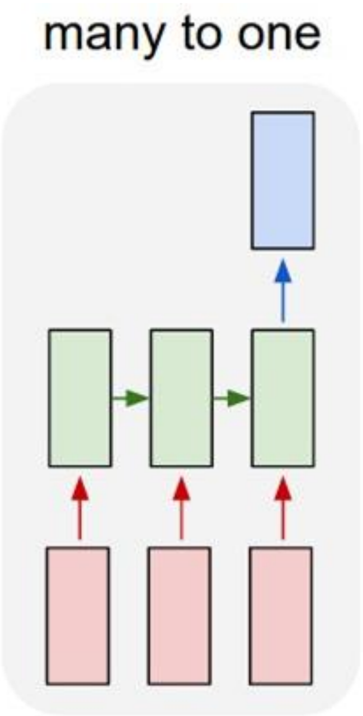
Two dogs play in the grass.

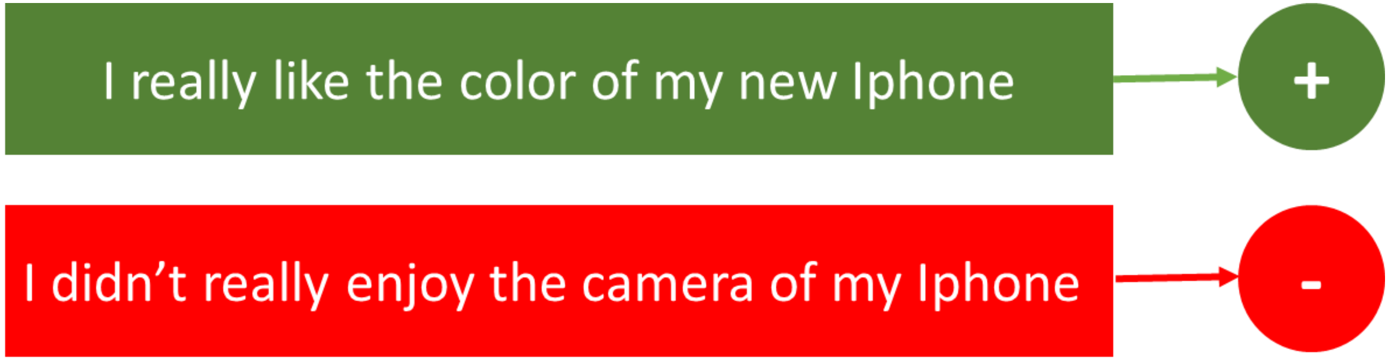
A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

Sequence to Sequence Learning Examples (2/3)

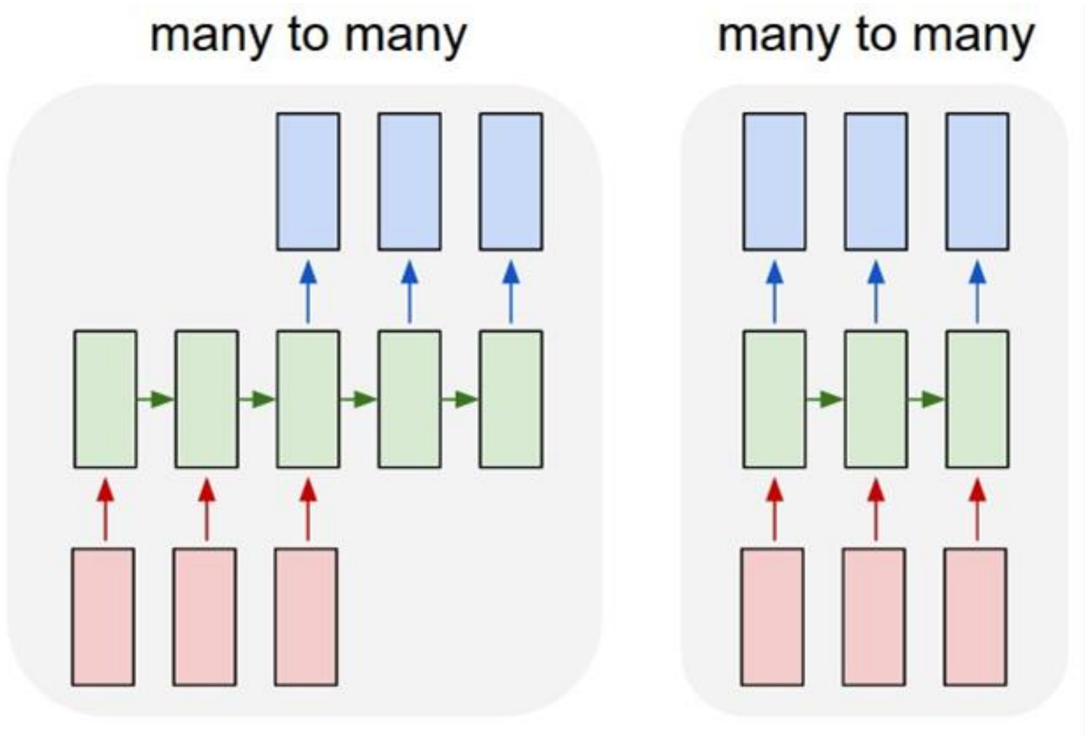
Sentiment Classification/Analysis: input a sequence of characters or words, e.g., a tweet, and classify the sequence into positive or negative sentiment. The input has a varying lengths, while output is of a fixed type and size.





Sequence to Sequence Learning Examples (3/3)

Language Translation: having some text in a particular language, e.g., English, we wish to translate it in another, e.g., French. Each language has it’s own semantics and would have varying lengths for the same sentence.



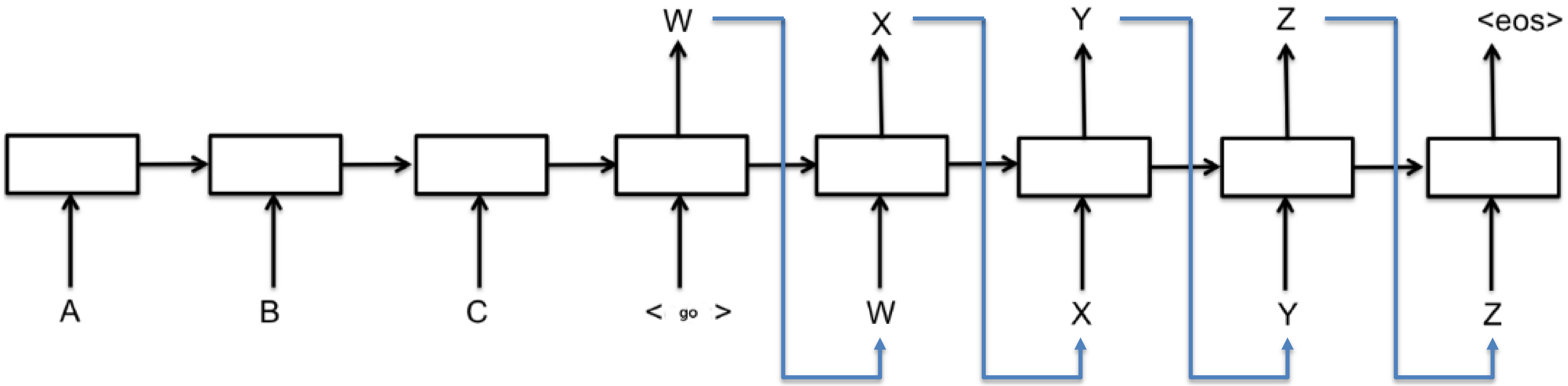
French was the official language of the colony of French Indochina, comprising modern-day Vietnam, Laos, and Cambodia. It continues to be an administrative language in Laos and Cambodia, although its influence has waned in recent years.

Le français était la langue officielle de la colonie de l'Indochine française, comprenant le Vietnam d'aujourd'hui, le Laos et le Cambodge. Il continue d'être une langue administrative au Laos et au Cambodge, bien que son influence a décliné au cours des dernières années.

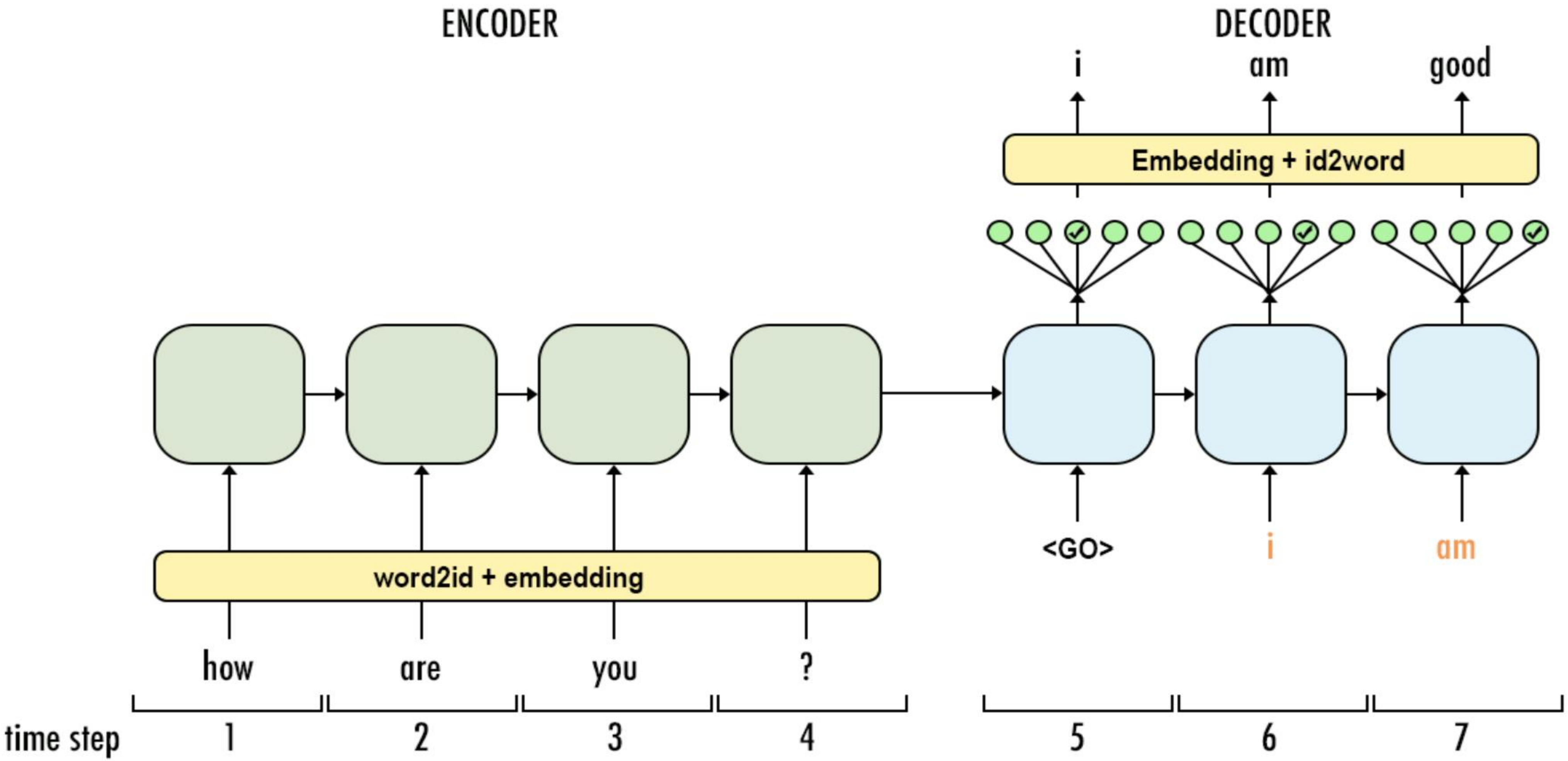
Seq2Seq Model Anatomy

The Seq2Seq model follows the classical encoder decoder architecture

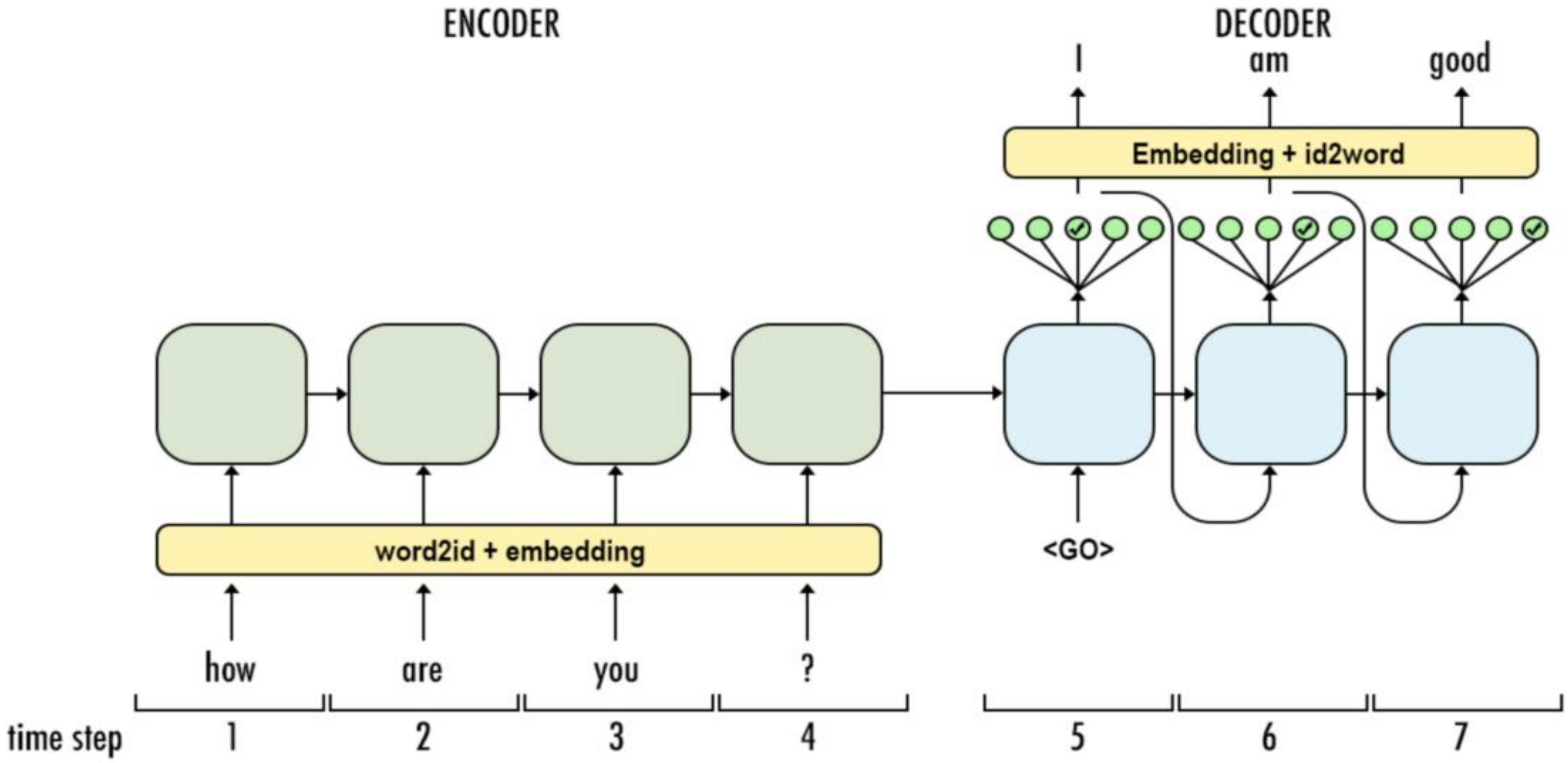
- At trainng time the decoder **does not** feed the output of each time step to the next; the inputs to the decoder time steps are the target from the training dataset
- At inference time the decoder feeds the output of each time step as an input to the next one



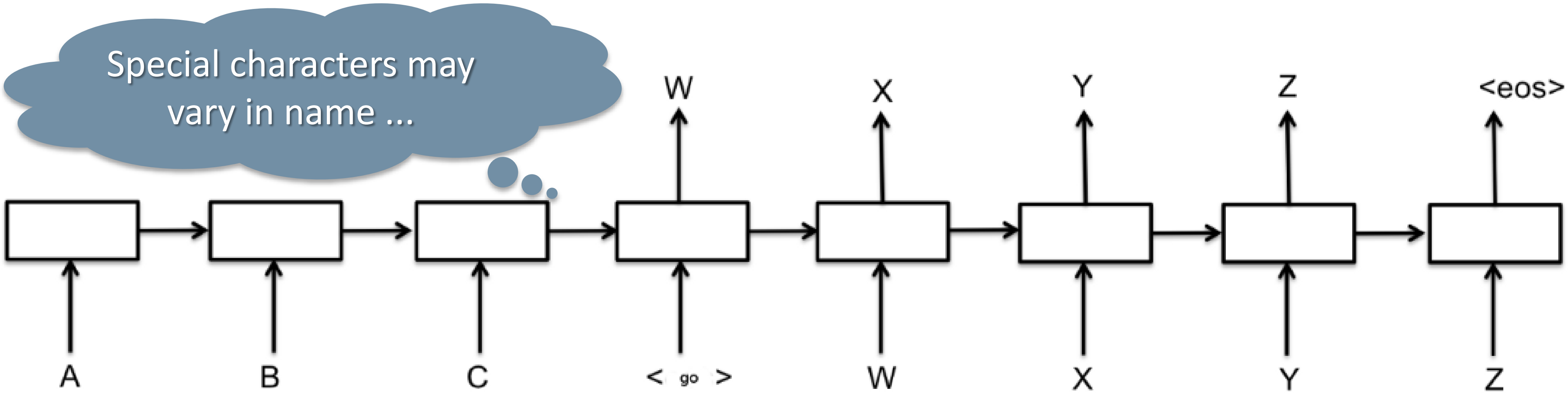
Seq2Seq Training Process



Seq2Seq Inference Process



Special Characters



<PAD>: During training, examples are fed to the network in batches. The inputs in these batches need to be the same width. This is used to pad shorter inputs to the same width of the batch

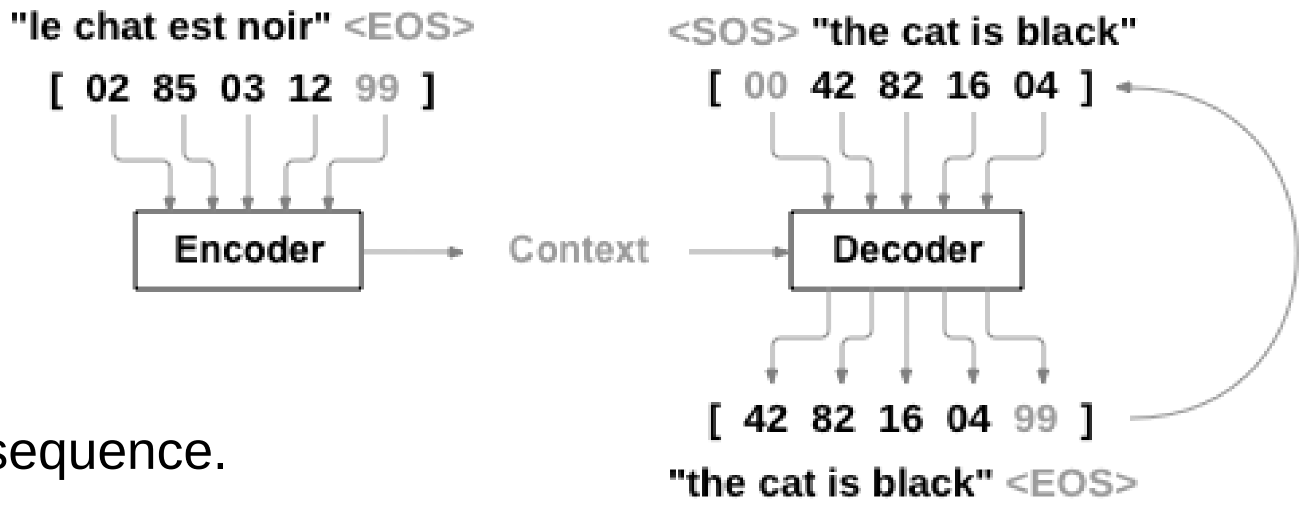
<EOS>: Needed for batching on the decoder side. It tells the decoder where a sentence ends, and it allows the decoder to indicate the same thing in its outputs as well.

<UNK>: On real data, it can vastly improve the resource efficiency to ignore words that do not show up often enough in your vocabulary by replace those with this character.

<SOS>/<GO>: This is the input to the first time step of the decoder to let the decoder know when to start generating output.

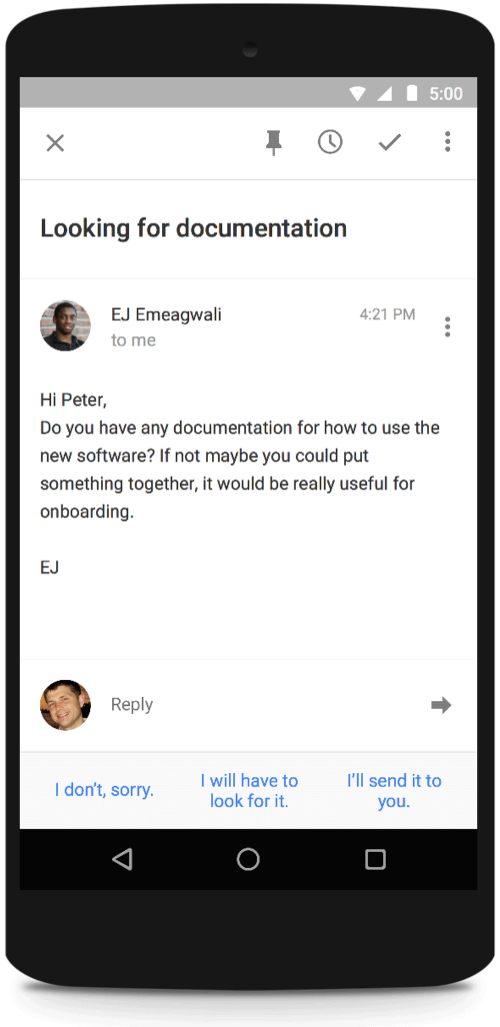
Dataset Batch Preparation

1. Sample batch_size pairs of (source_sequence, target_sequence).
2. Append <EOS> to the source_sequence
3. Prepend <SOS> to the target_sequence to obtain the target_input_sequence and append <EOS> to obtain target_output_sequence.
4. Pad up to the max_input_length (max_target_length) within the batch using the <PAD> token.
5. Encode tokens based of vocabulary (or embedding)
6. Replace out of vocabulary (OOV) tokens with <UNK>.
   Compute the length of each input and target sequence in the batch.

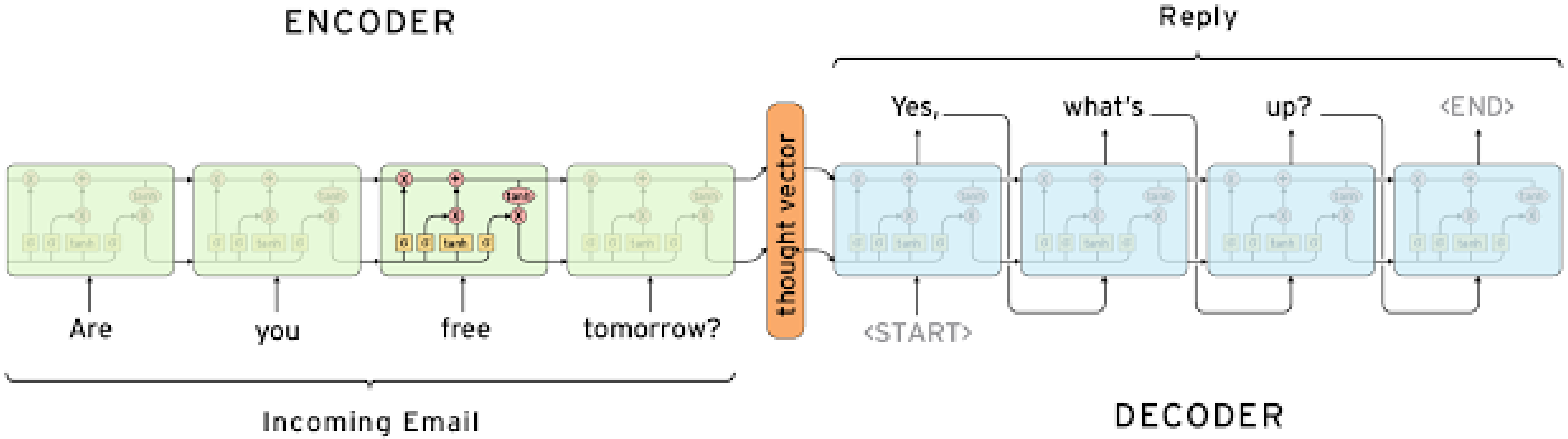



```
Vocabulary = {"<SOS>": 00,
              "<EOS>": 99,
              "<UNK>": 01,
              "<PAD>": 03,
              "the": 42,
              "is": 16,
                ...      }
```

Sequence to Sequence Modeling



Given <S, T> pairs, read S, and output T' that best matches T

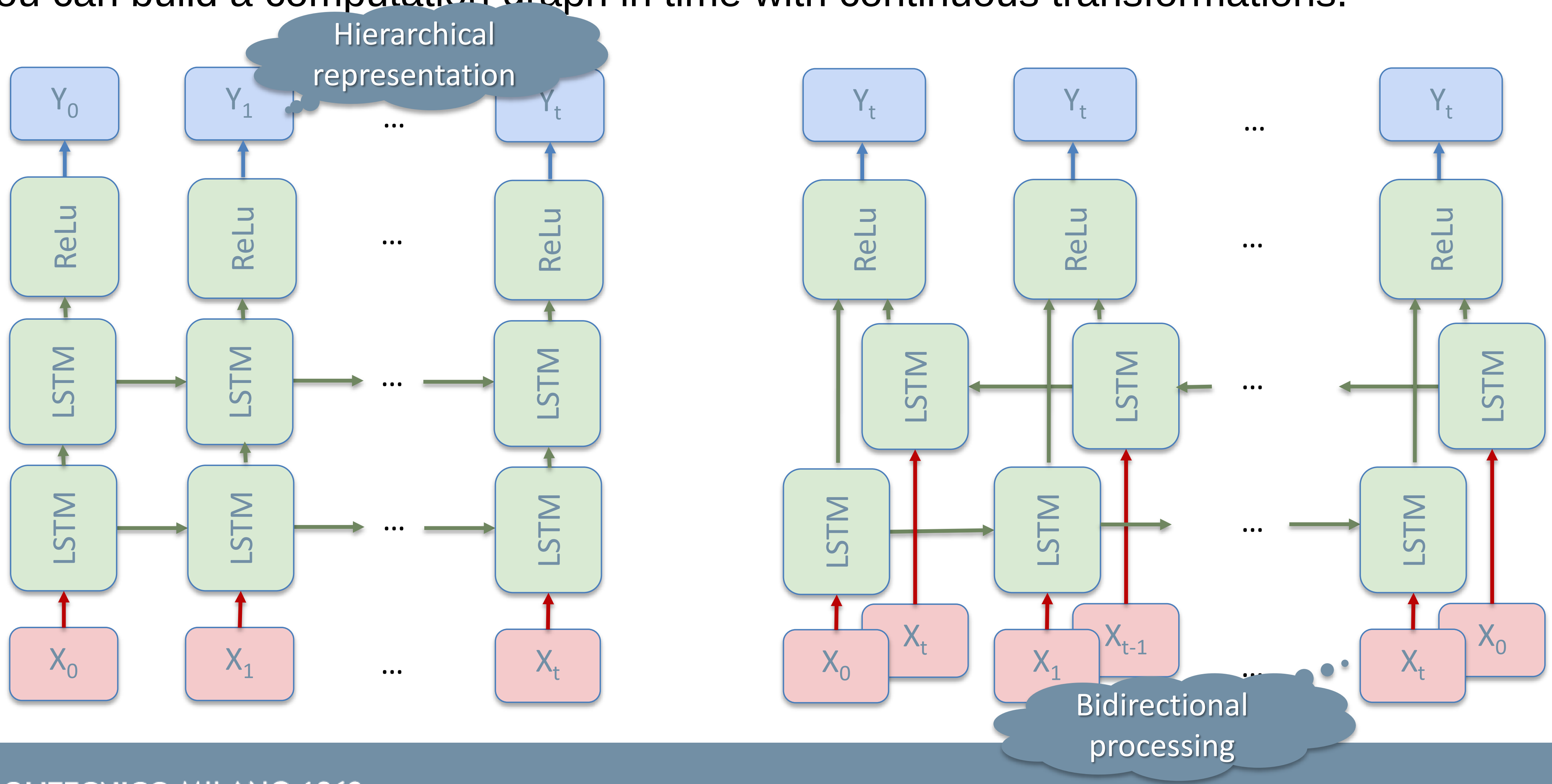


$$p(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots, y_{t-1})$$

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

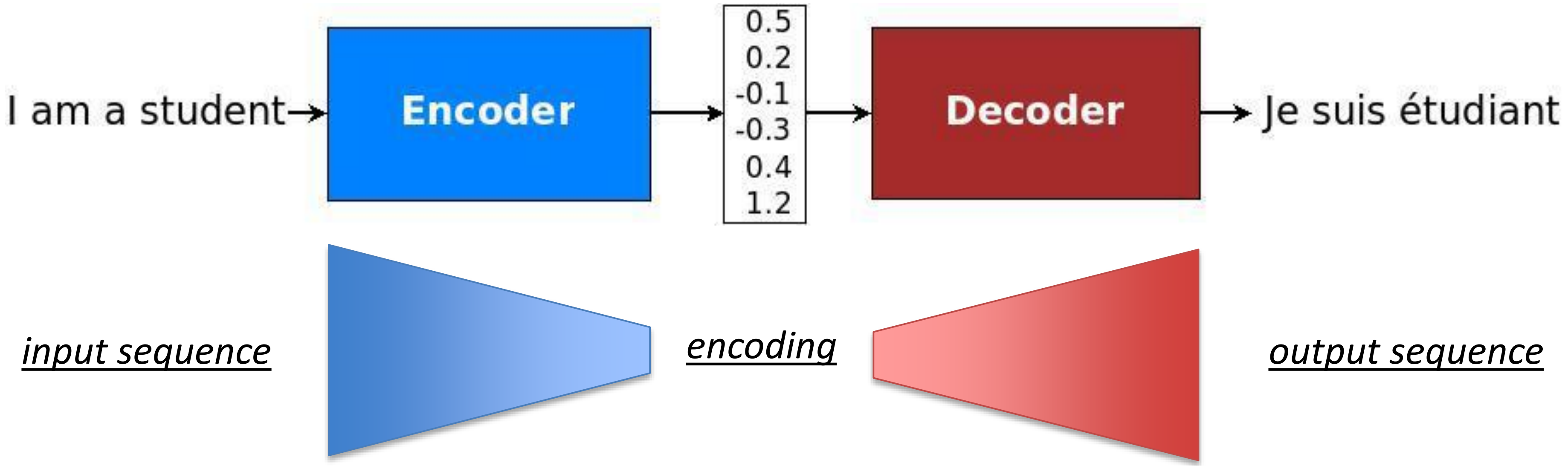
Multiple Layers and Bidirectional LSTM Networks

You can build a computation graph in time with continuous transformations.



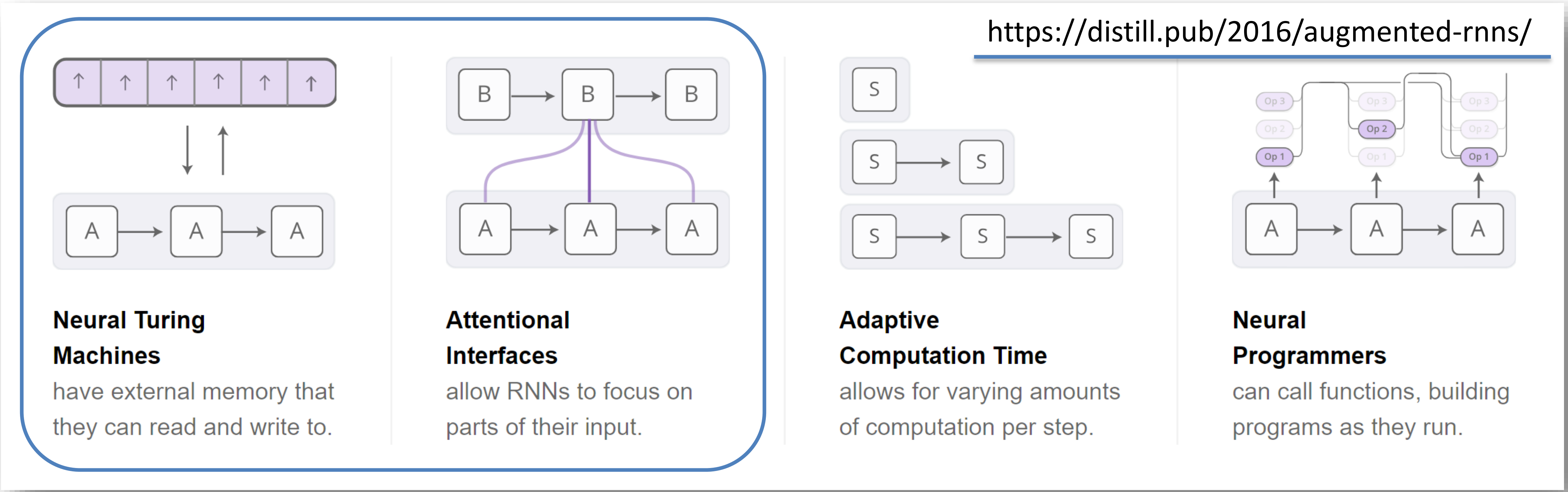
Extending Recurrent Neural Networks

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



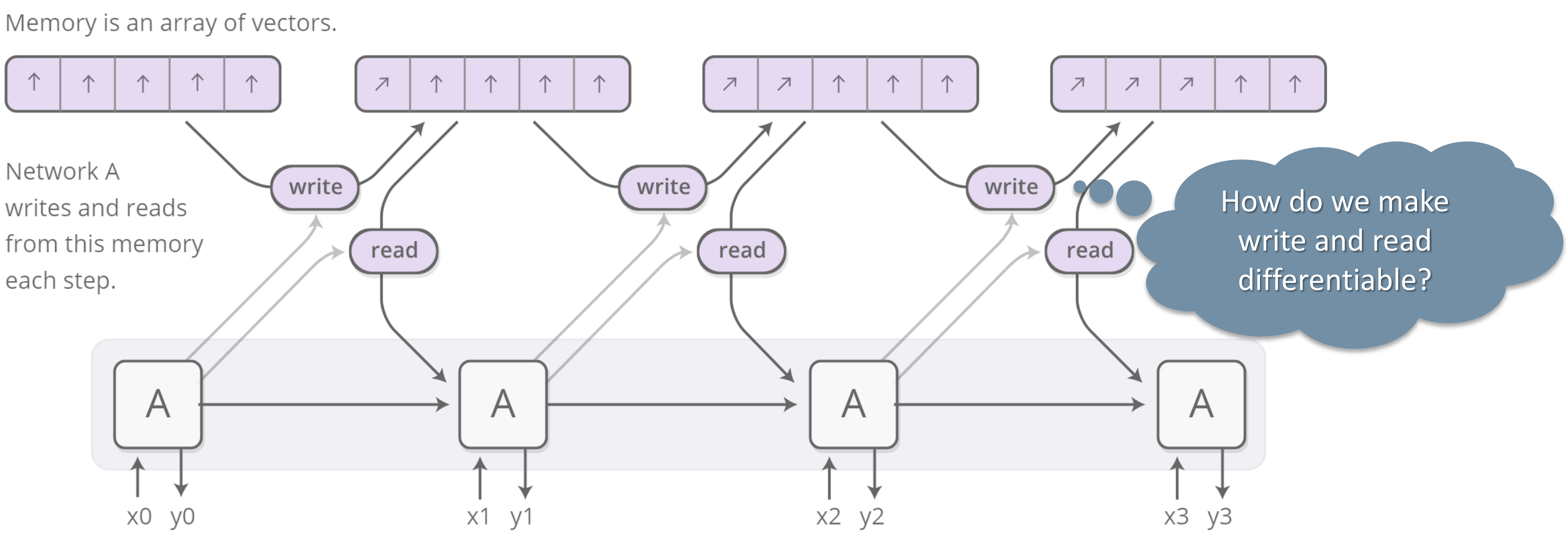
Extending Recurrent Neural Networks

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



Neural Turing Machines

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



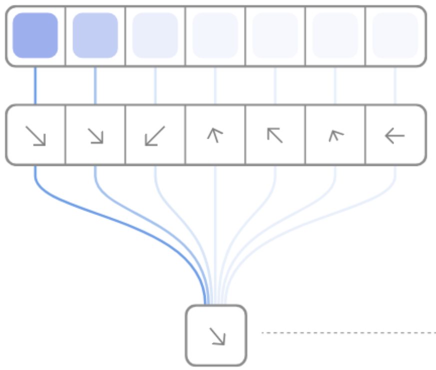
Neural Turing Machines Idea

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 to

Attention mechanism!

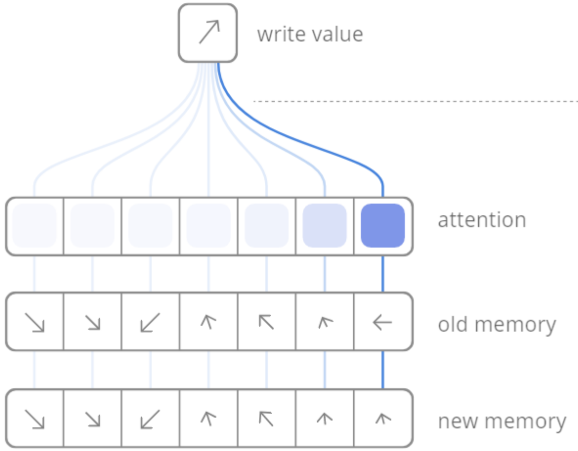
NTM solution: ***<u>Every step, they read and write everywhere, just to different extents</u>***.



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.

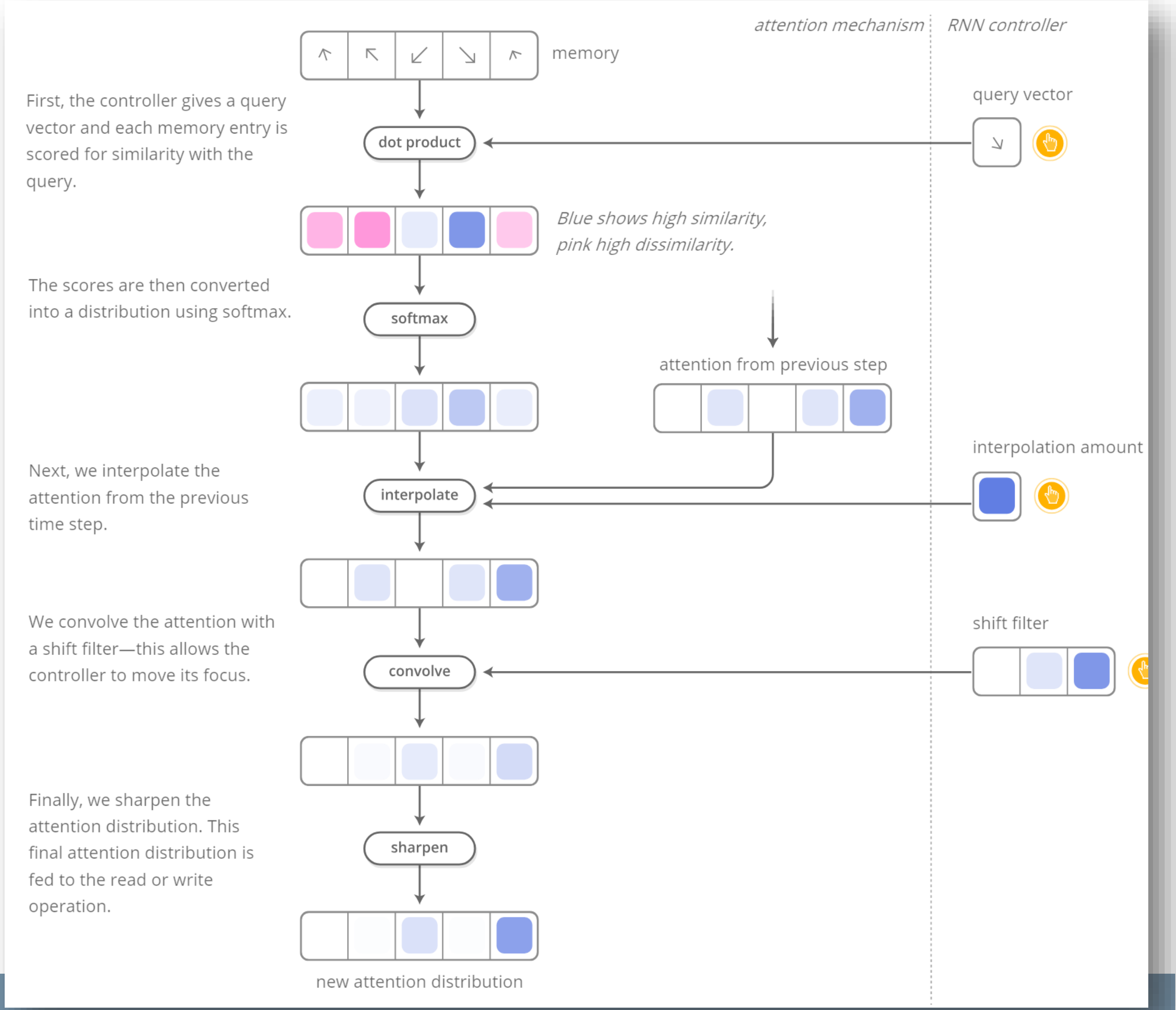
The RNN gives an attention distribution, describing how much we should change each memory position towards the write value.

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

Neural Turing Machines Attention

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

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



Neural Turing Machines Extensions

Thanks to memory, NTM can 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

Considering the sequential dataset:

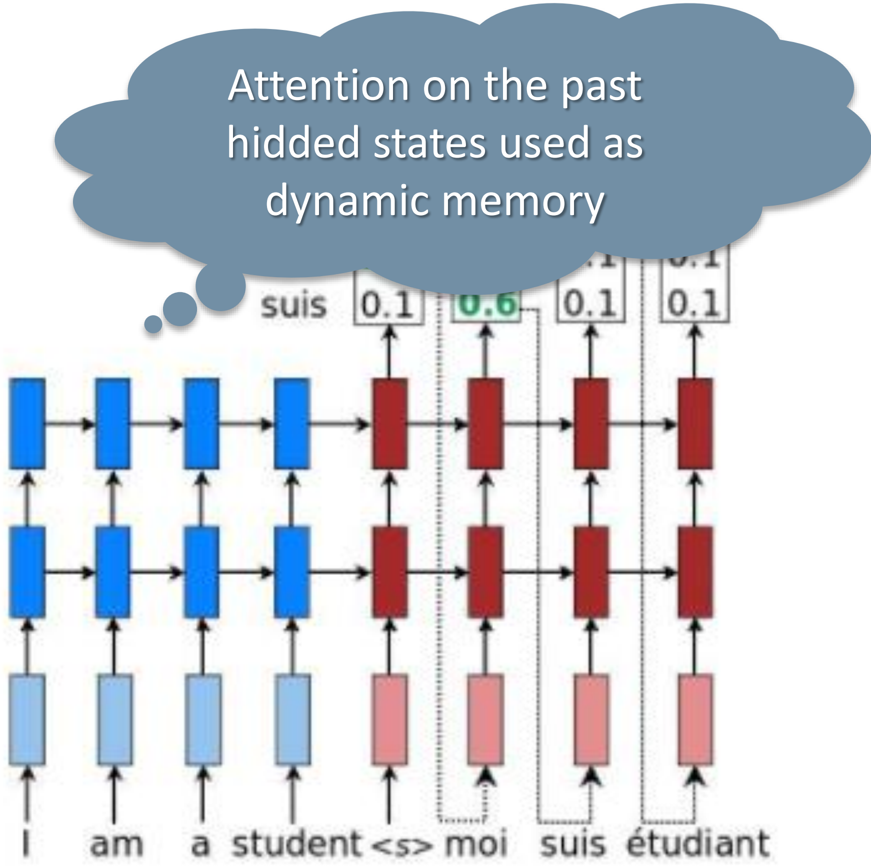
$$\{((x_1, \ldots, x_n), (y_1, \ldots, y_m))\}_{i=1}^{N}$$

The decoder role is to model the generative probability:

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

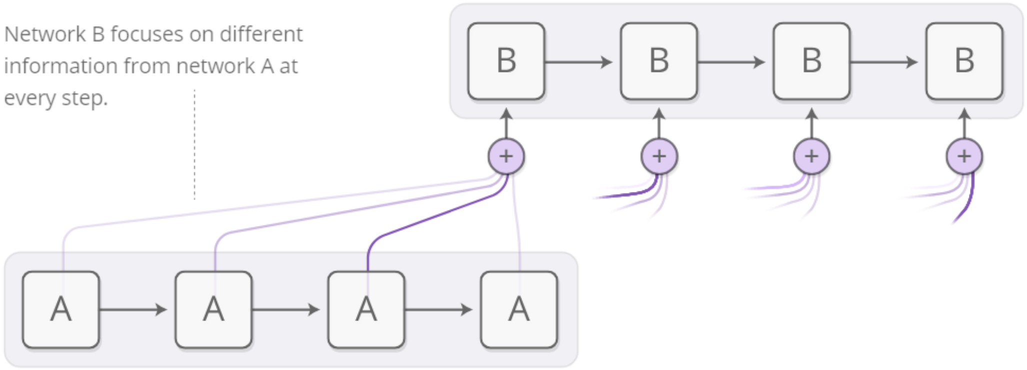
In “vanilla” seq2seq models, the decoder is conditioned initializing the initial state with last state of the encoder.

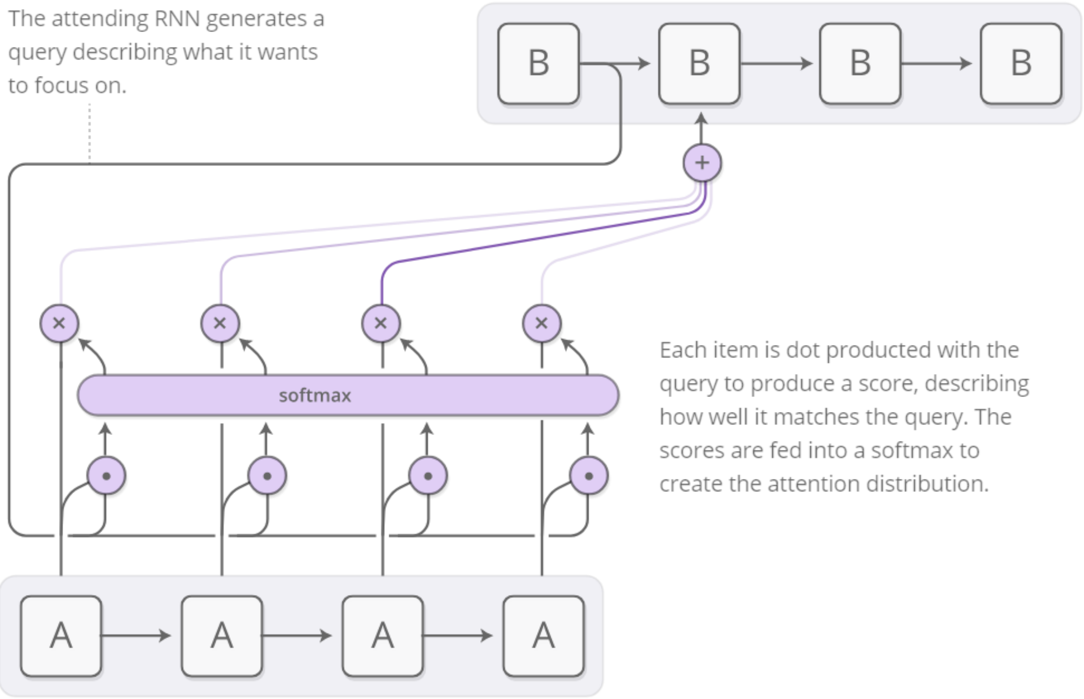
Works well for short and medium-length sentences; however, for long sentences, becomes a bottleneck



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

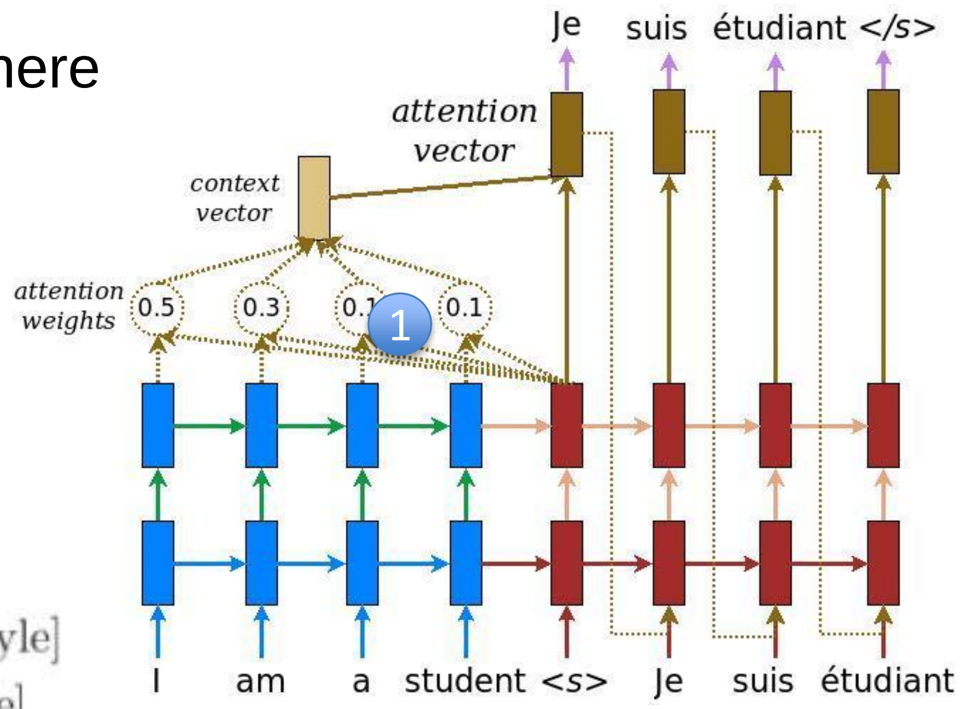
Attention Mechanism in Seq2Seq Models

The attention function as mapping a query and a 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 h_t, with all source states h_s to derive attention scores

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



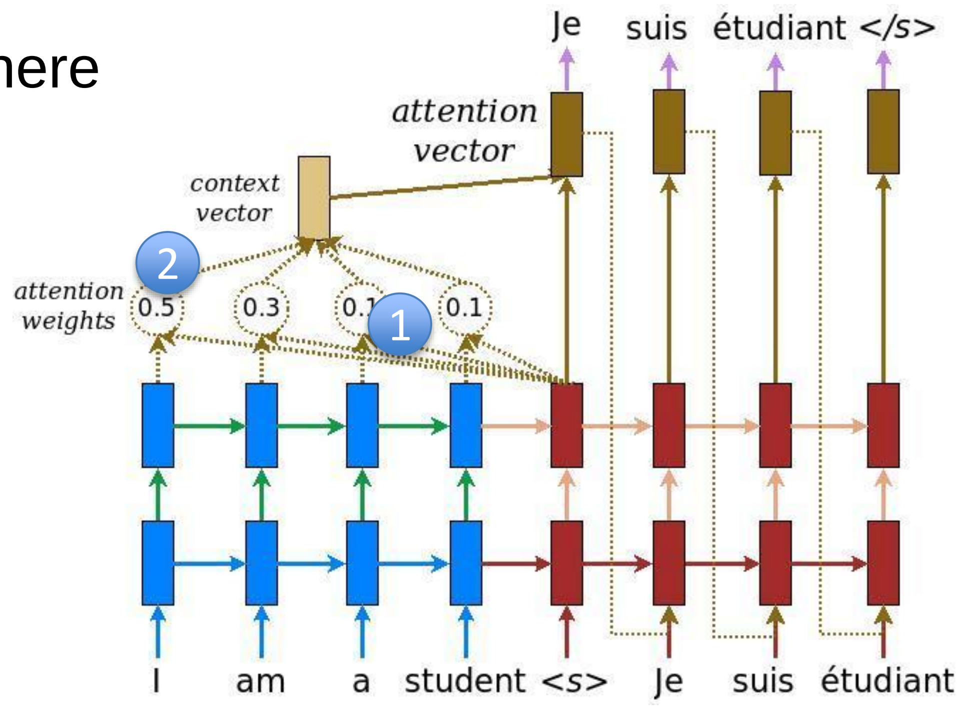
Attention Mechanism in Seq2Seq Models

The attention function as mapping a query and a 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

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



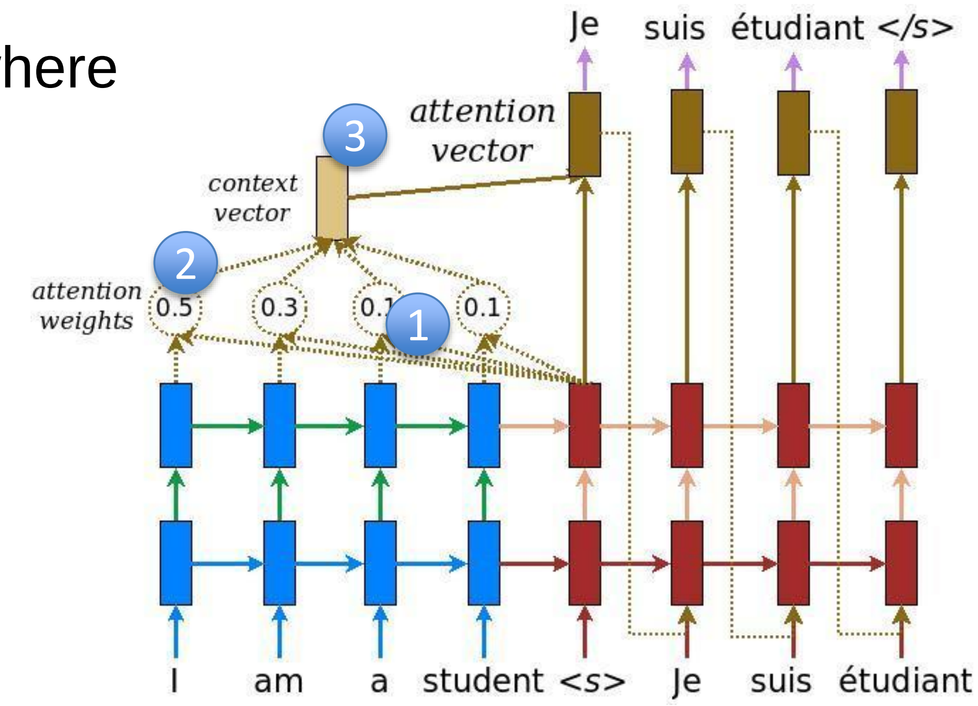
Attention Mechanism in Seq2Seq Models

The attention function as mapping a query and a 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

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



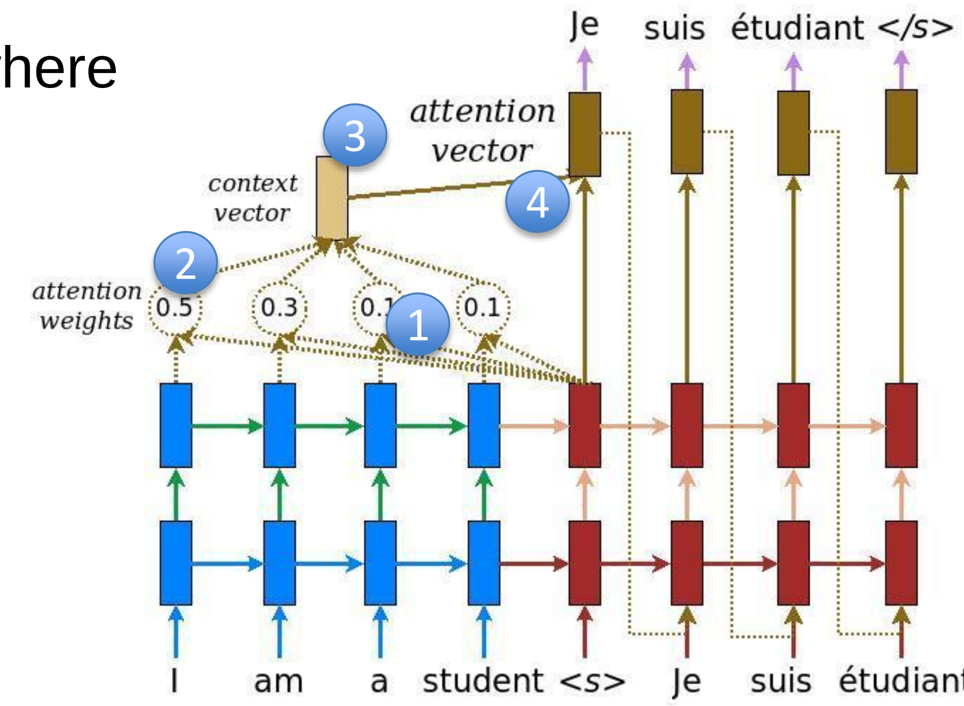
Attention Mechanism in Seq2Seq Models

The attention function as mapping a query and a 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

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

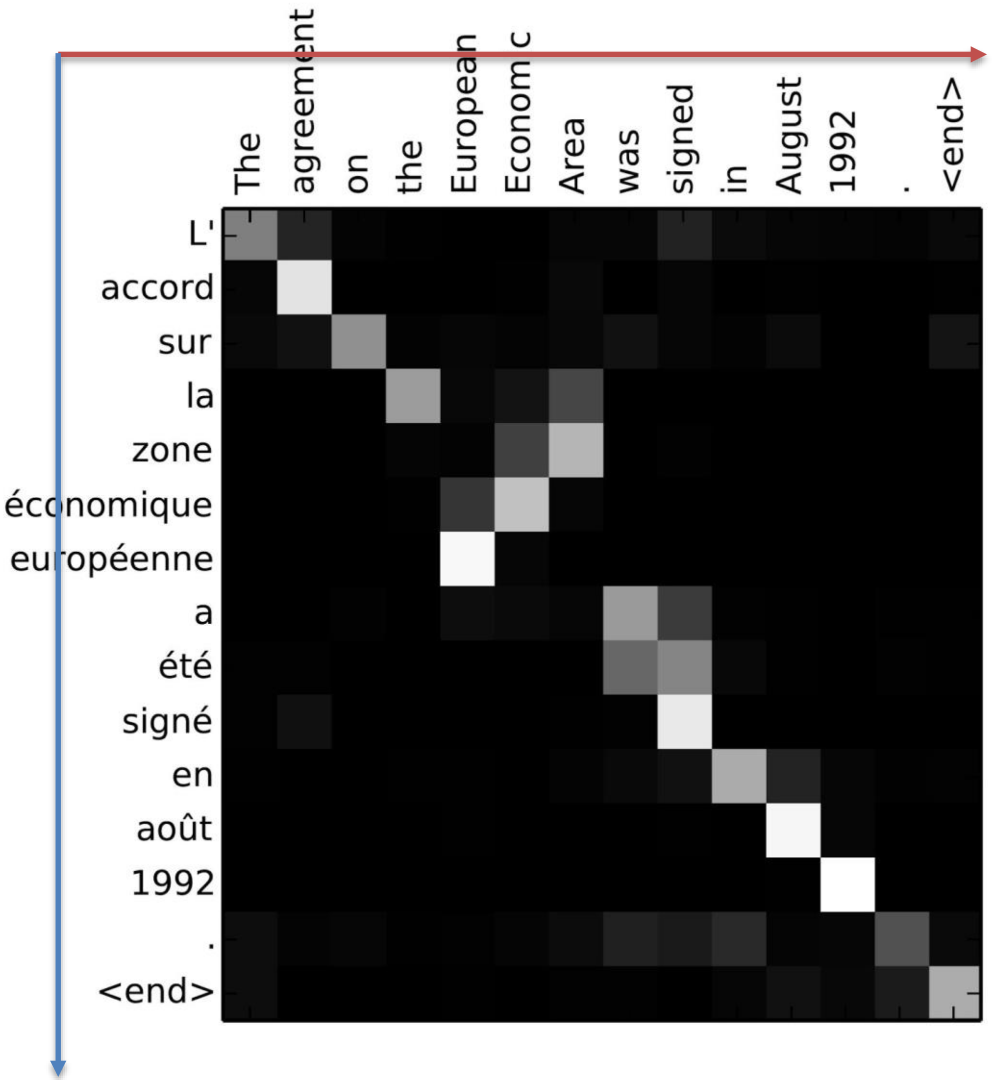


Attention Visualization

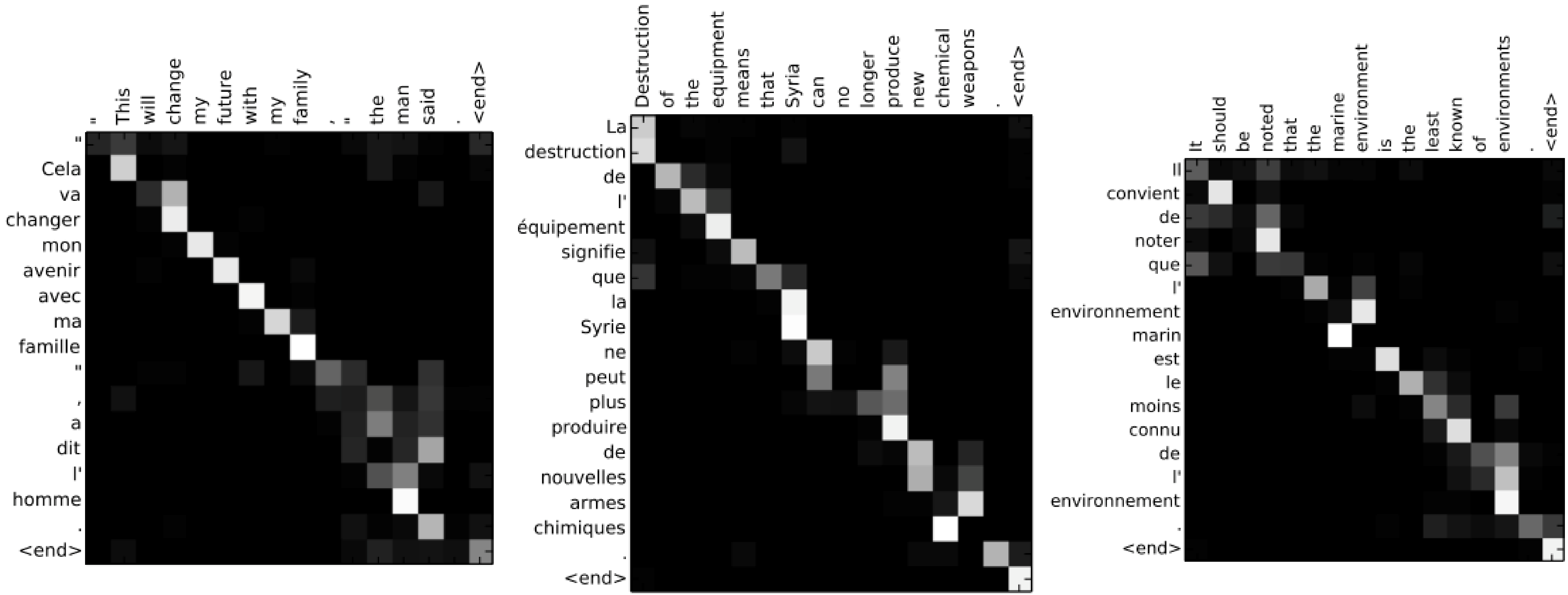
Alignment matrix is use to visualize attention weights between source and target sentences.

For each decoding step, i.e., for 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 as they become relevant.

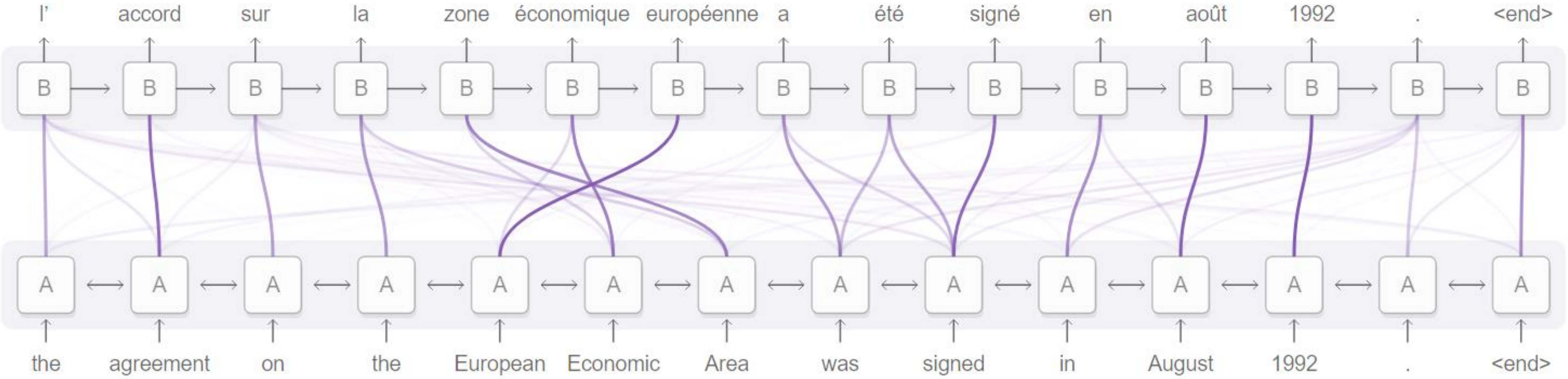


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

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.

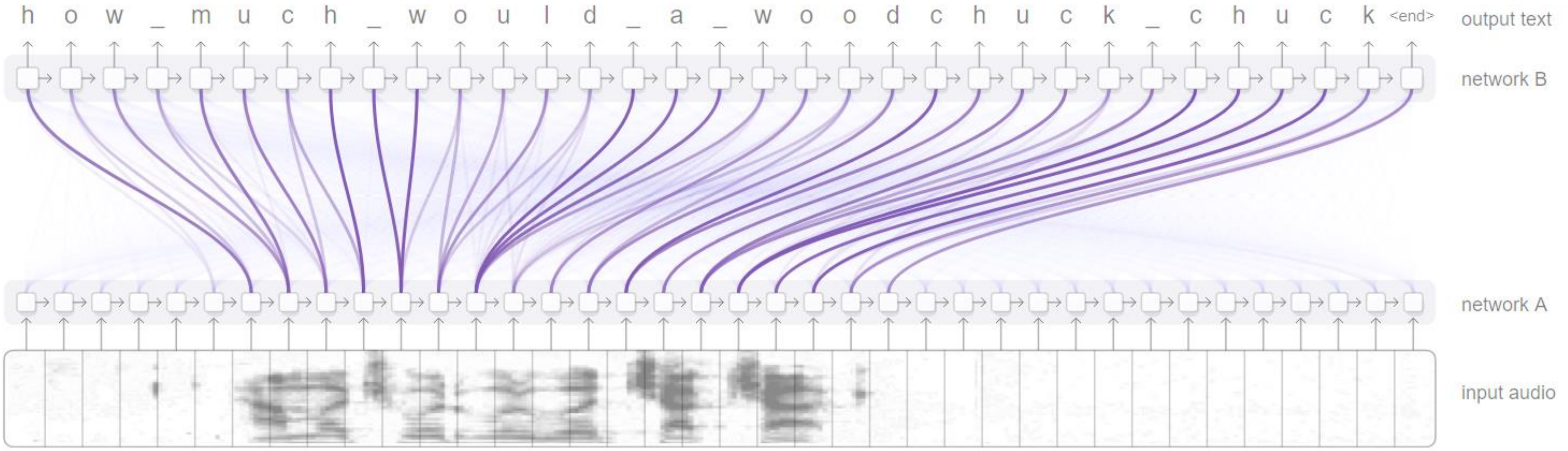


Figure derived from Chan, *et al.* 2015

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.

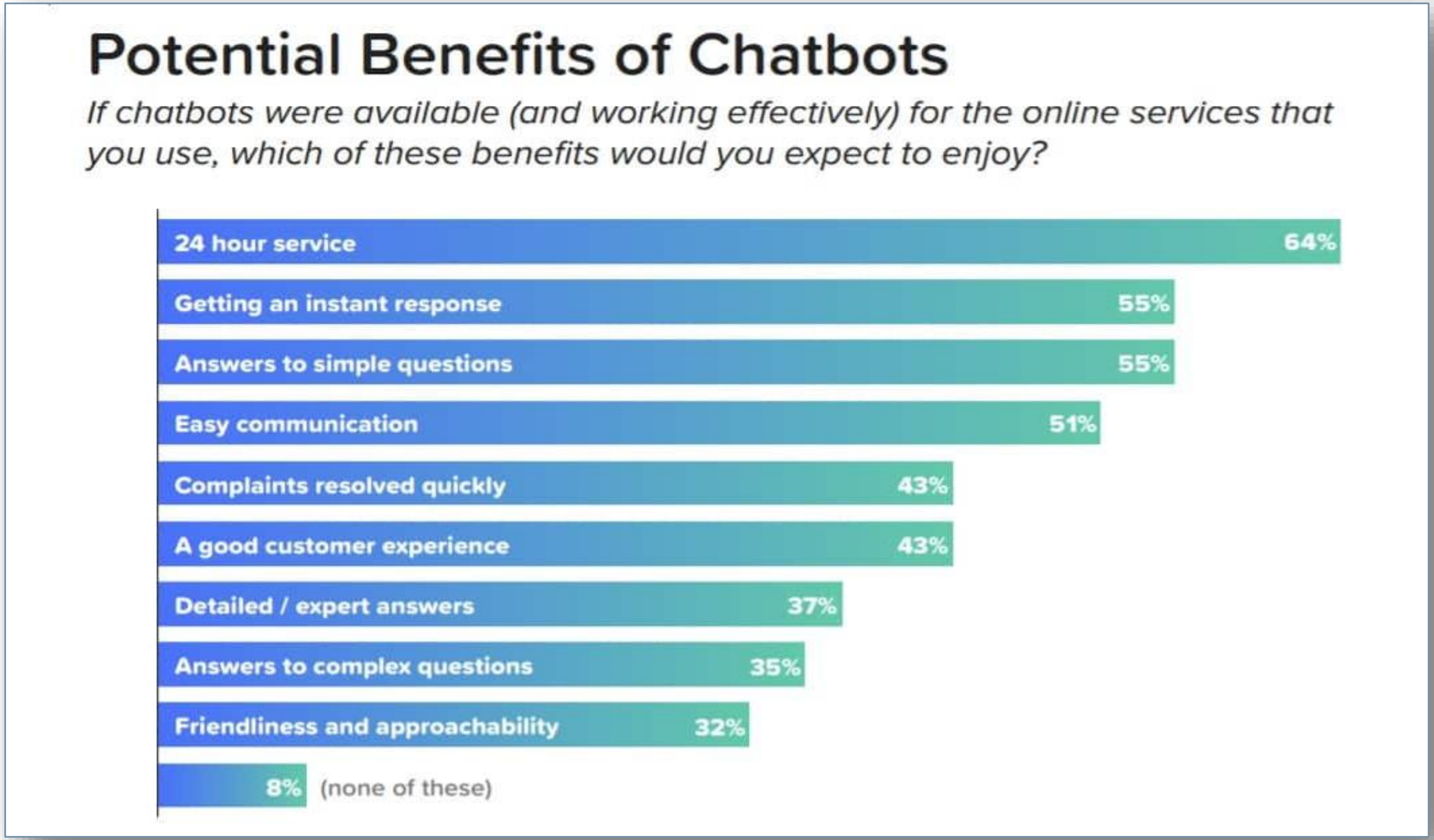
As it generates each word in the description, the RNN focuses on the CNN interpretation of the relevant parts of the image.

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.

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

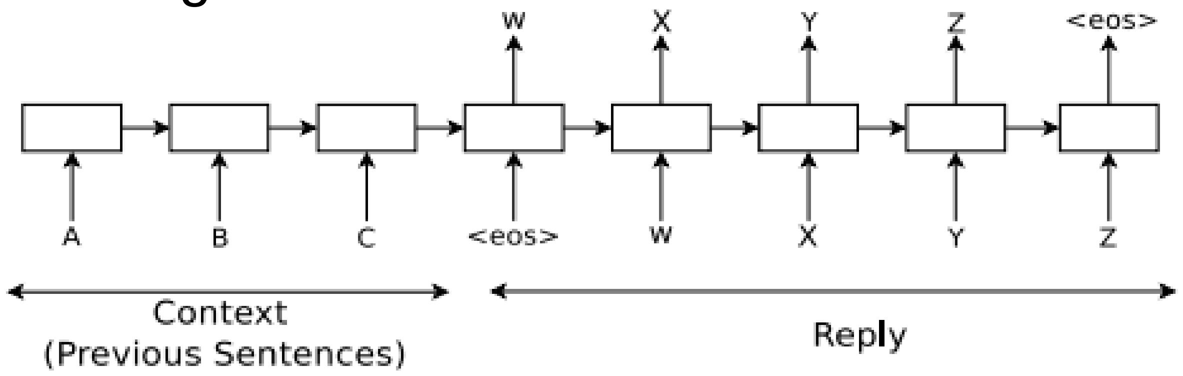
- Multi-turn: the input vector is built by considering a multi-turn conversational context, containing also incoming question.

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

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”



The idea of generative chatbots is to 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

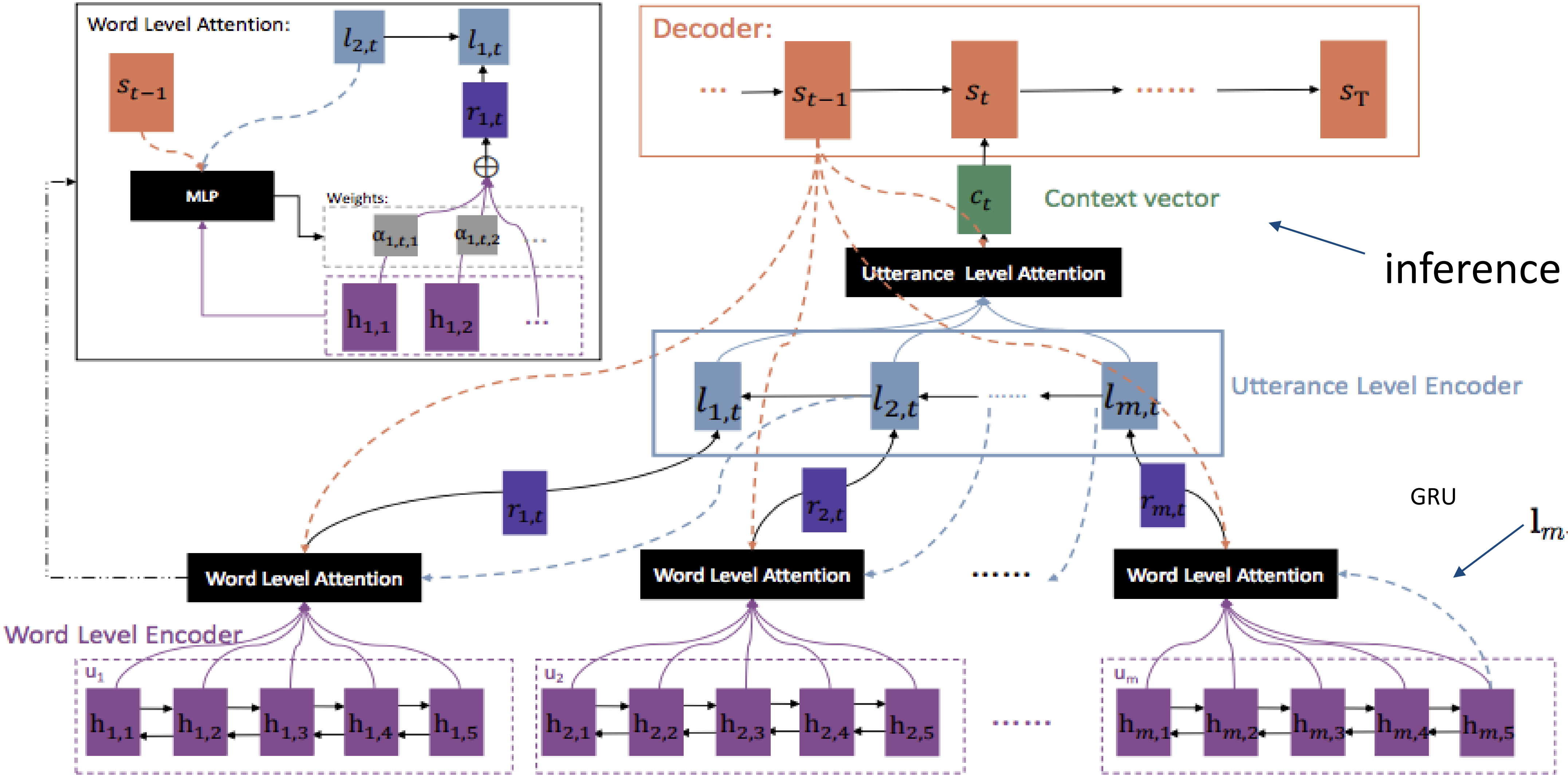
The first idea could be **concatenating multiple turns** into a **single long input sequence**, but this would probably result in poor performances.

- 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 the attention mechanism from single-turn response generation to a hierarchical attention mechanism for multi-turn response generation

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

Hierarchical Generative Multi-turn Chatbots



inference !

GRU

$l_{m+1,t}$

Hierarchical Generative Multi-turn Chatbots

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

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

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

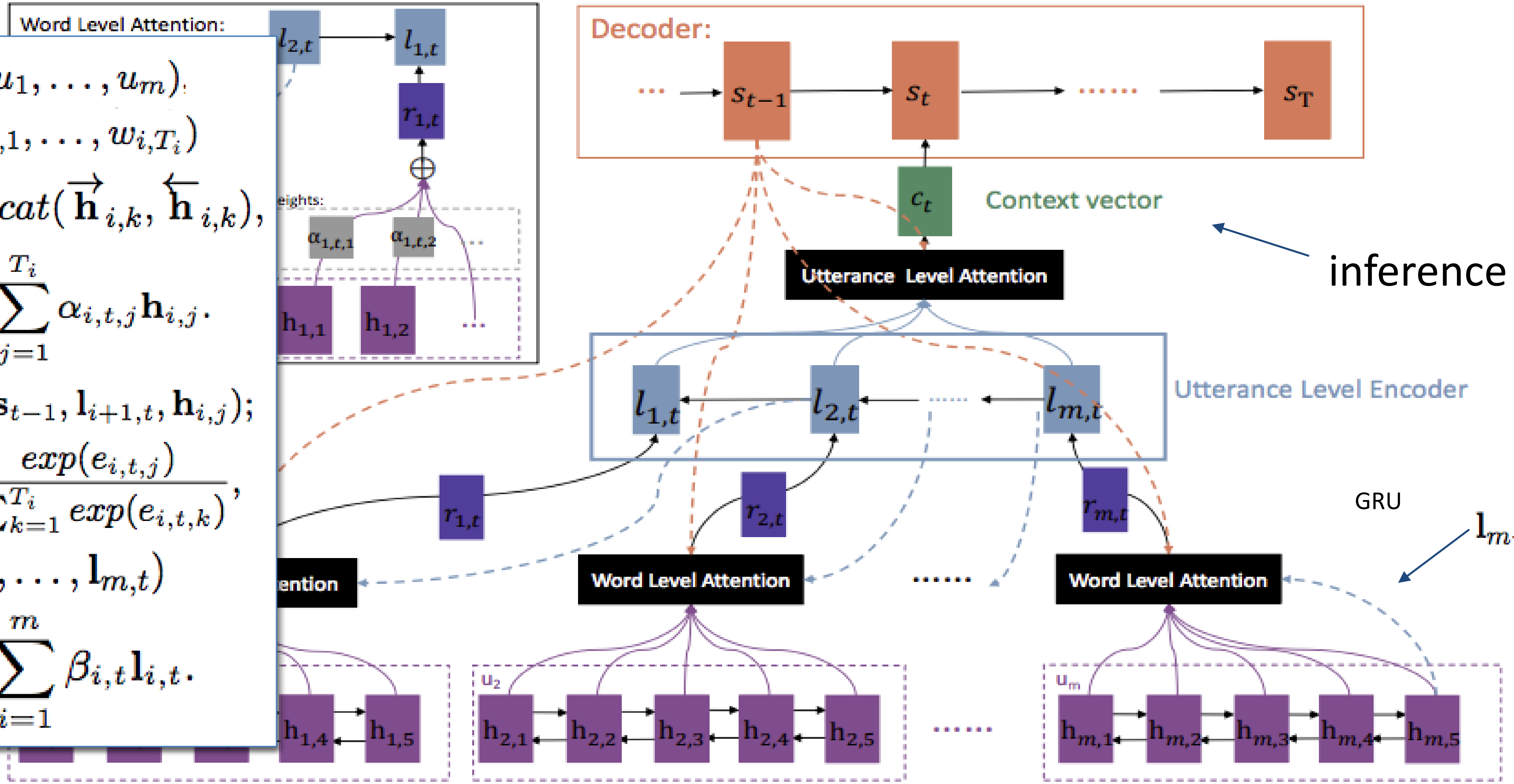
$$\mathbf{r}_{i,t} = \sum_{j=1}^{T_i} \alpha_{i,t,j} \mathbf{h}_{i,j}.$$

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

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

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

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



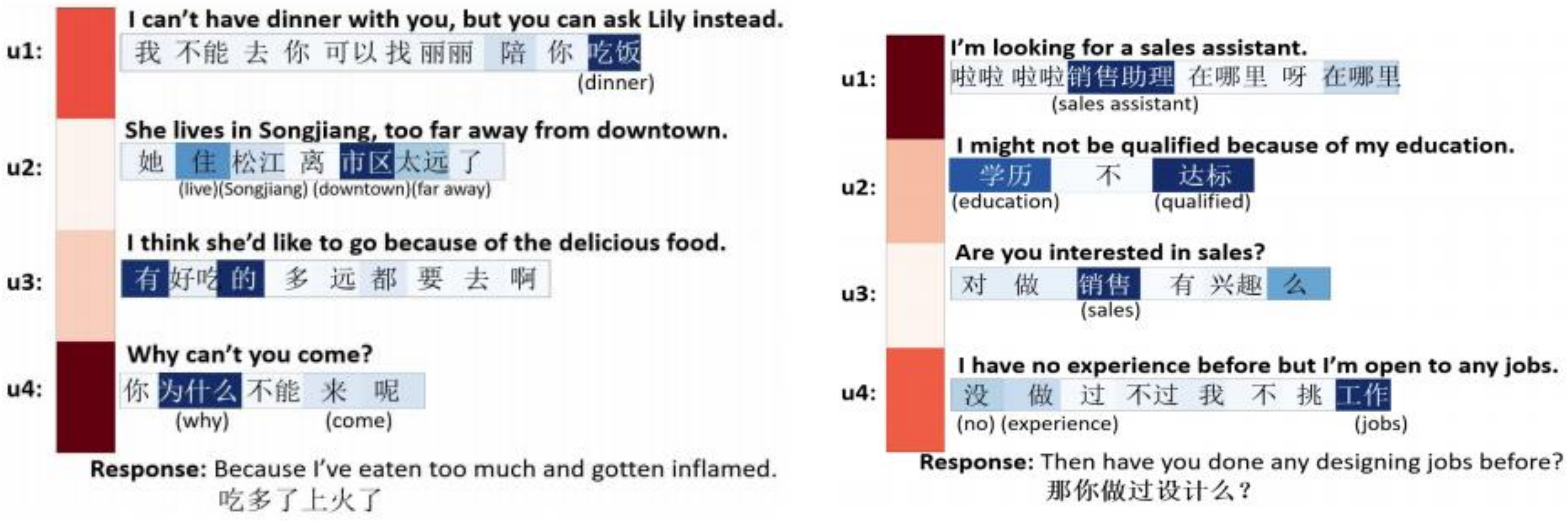
inference !

GRU

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

Hierarchical Generative Multi-turn Chatbots

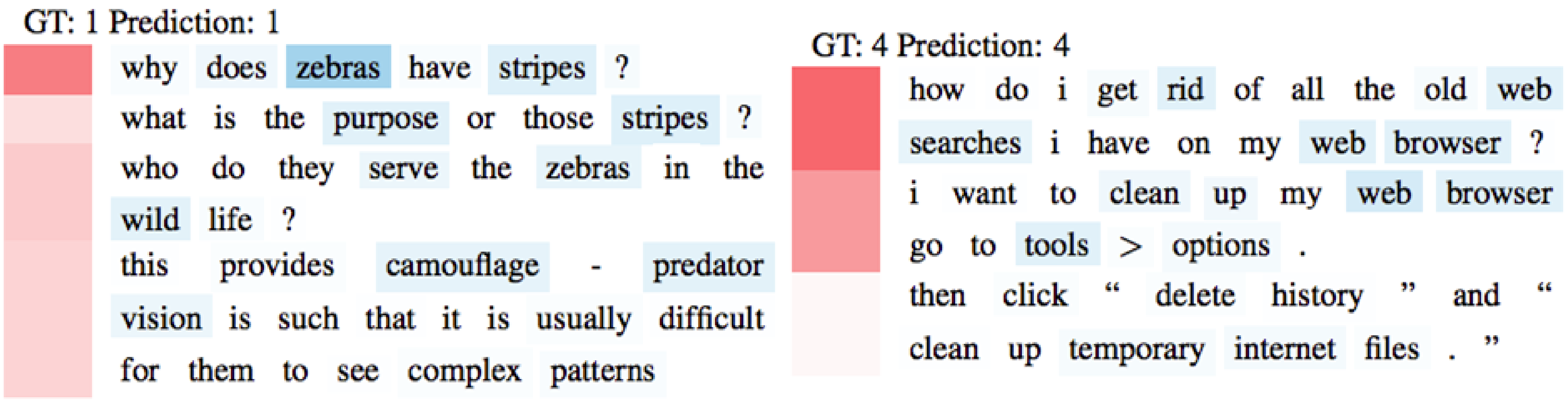
We can visualize hierarchical attention weights, darker color means more important words or utterances.



Hierarchical Document Classification

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

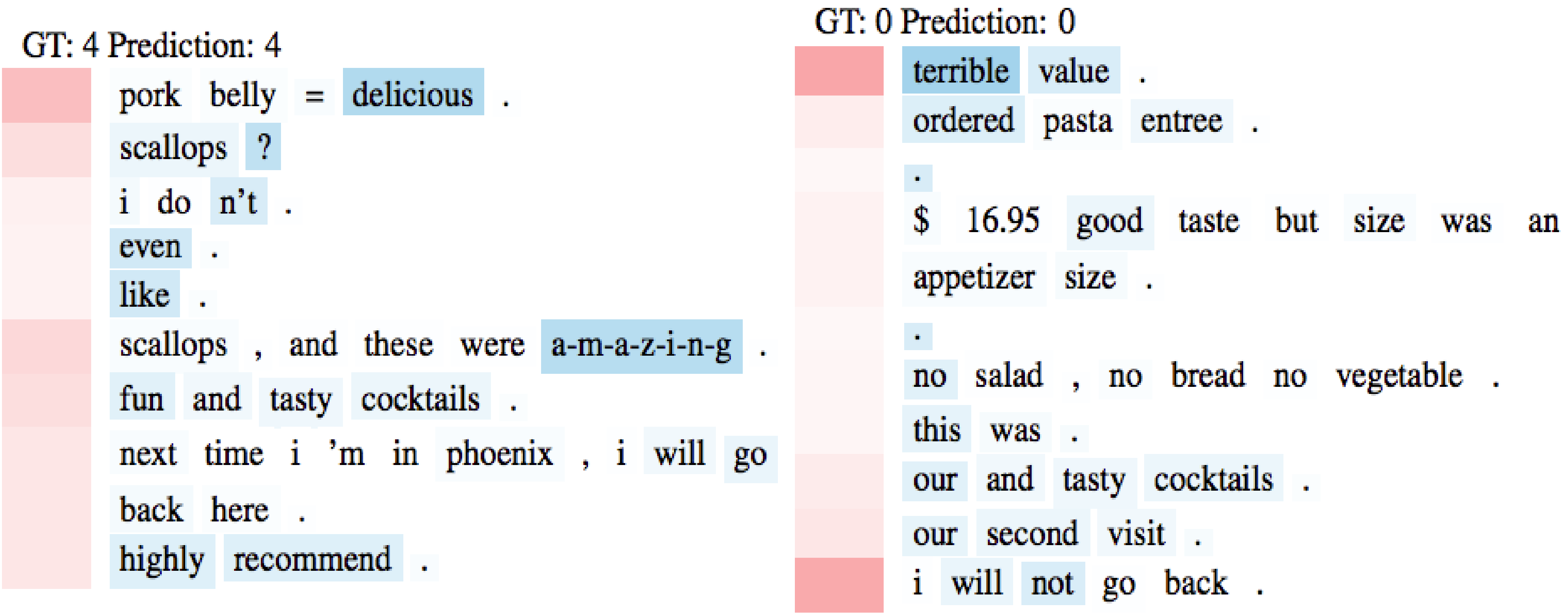
- Left document denotes Science and Mathematics; model accurately localizes the words ***zebra***, ***stripes***, ***camouflage***, ***predator*** and corresponding sentences.
- Right document denotes Computers and Internet; the model focuses on ***web***, ***searches***, ***browsers*** and their corresponding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



Hierarchical Document Classification

In Sentiment Analysis, the model can select words carrying strong sentiment like ***delicious***, ***amazing***, ***terrible*** and their 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!

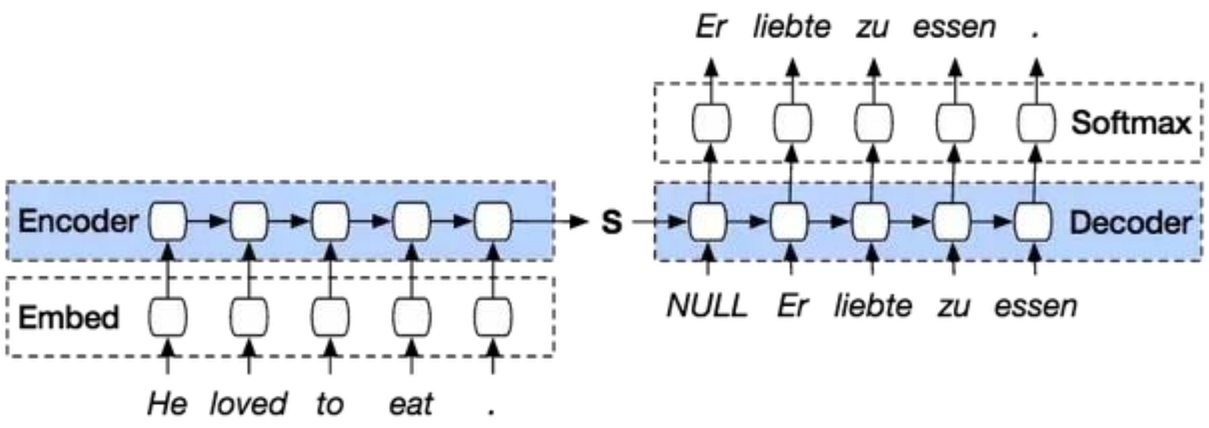
Once you have seen attention is what makes things working you start wondering:

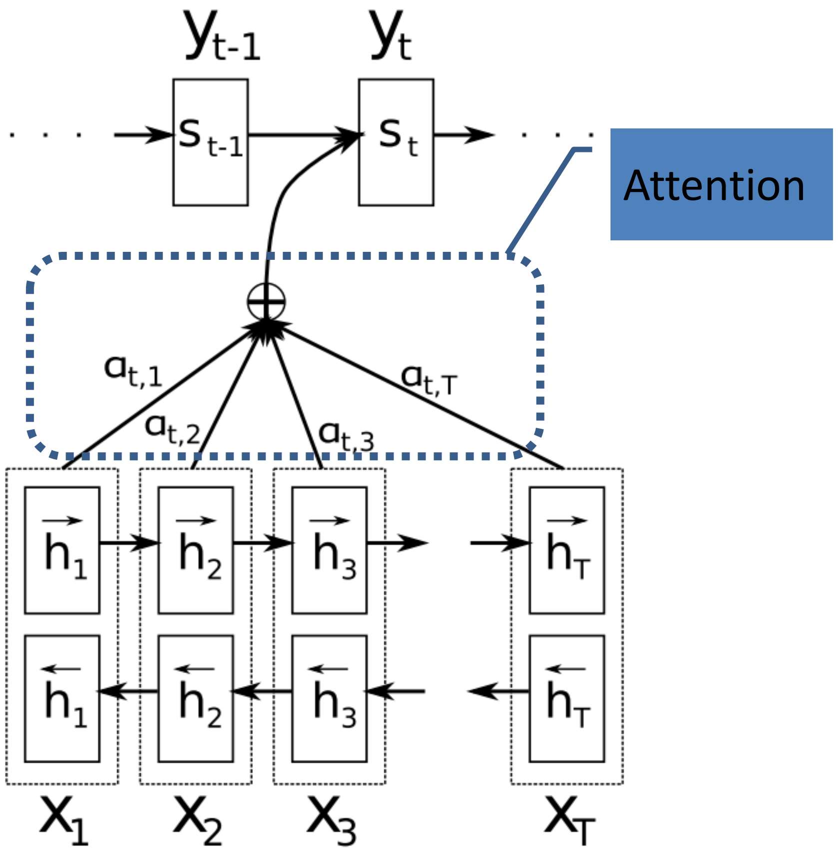
- Sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples.
- Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks. 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.

Example: Neural Machine Translation





- Observation: attention is built upon RNN
- The Transformer breaks this observation!

Transformer

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

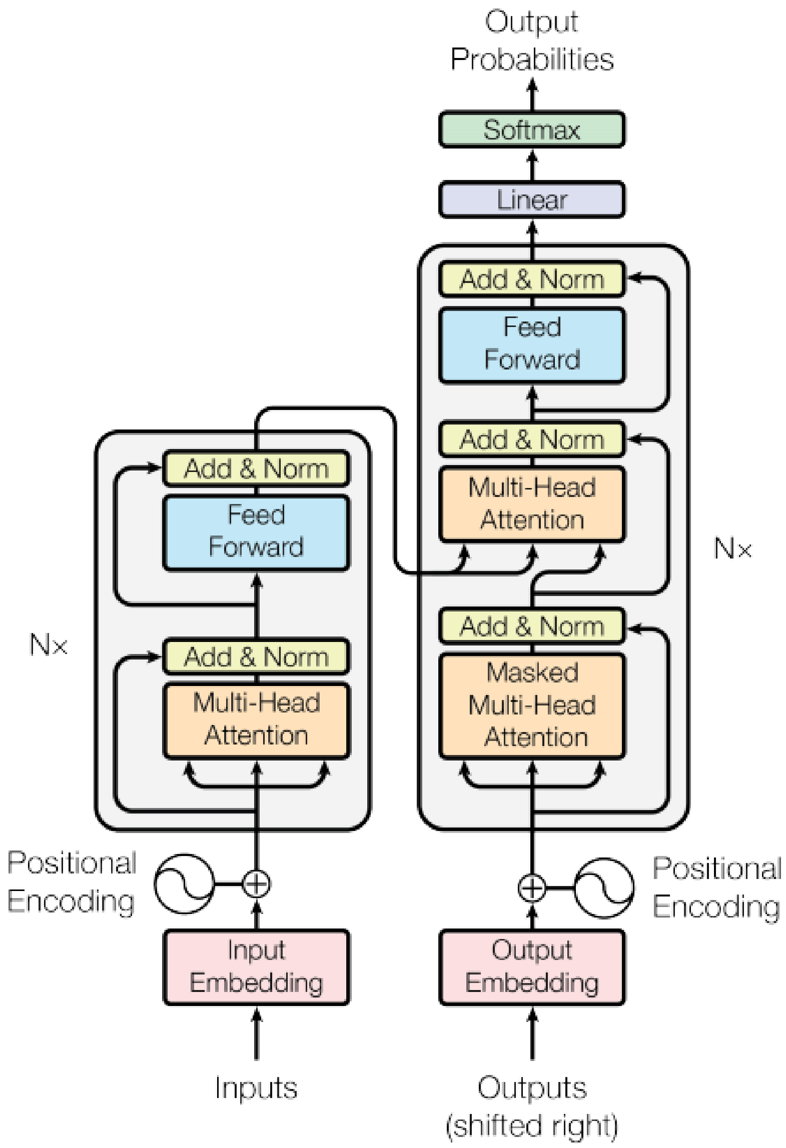


Figure 1: The Transformer - model architecture.

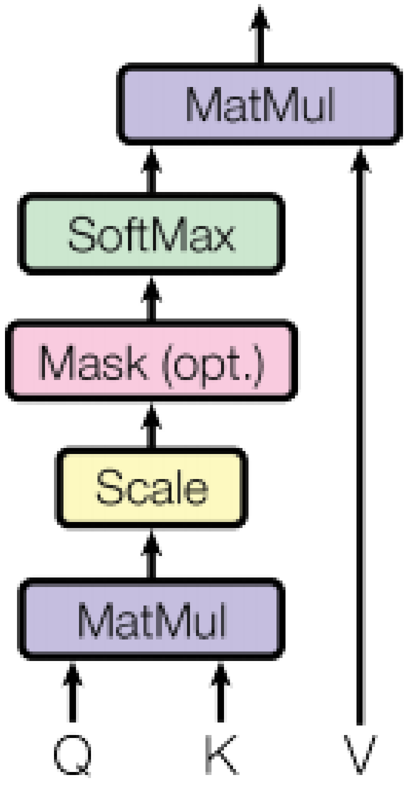
Transformer

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

Q: queries, K: keys, V: values

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

Scaled Dot-Product Attention

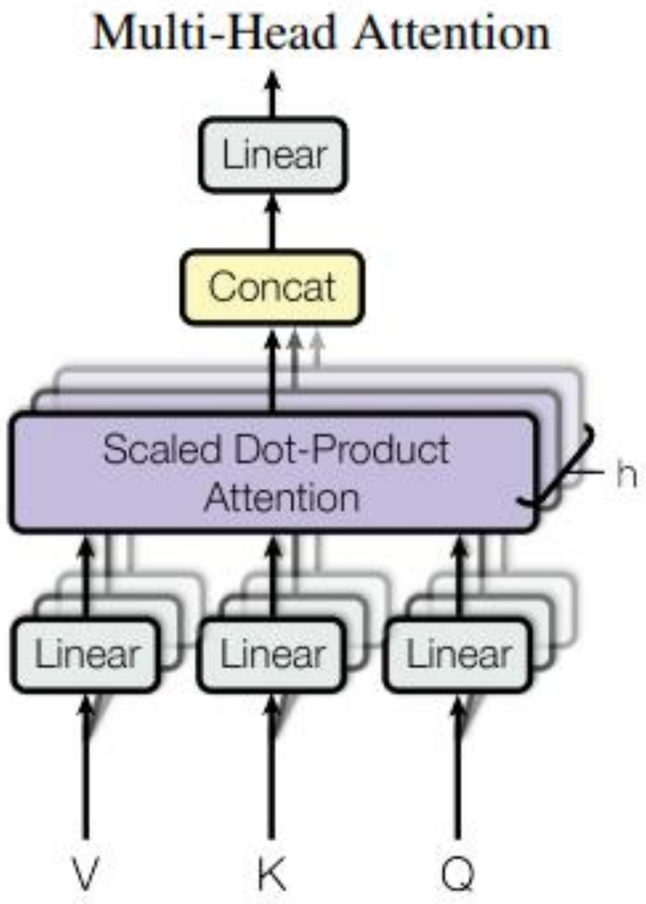


Transformer

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

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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 $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.



- The total computational cost is similar to that of single-head attention with full dimensionality.

Transformer

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

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

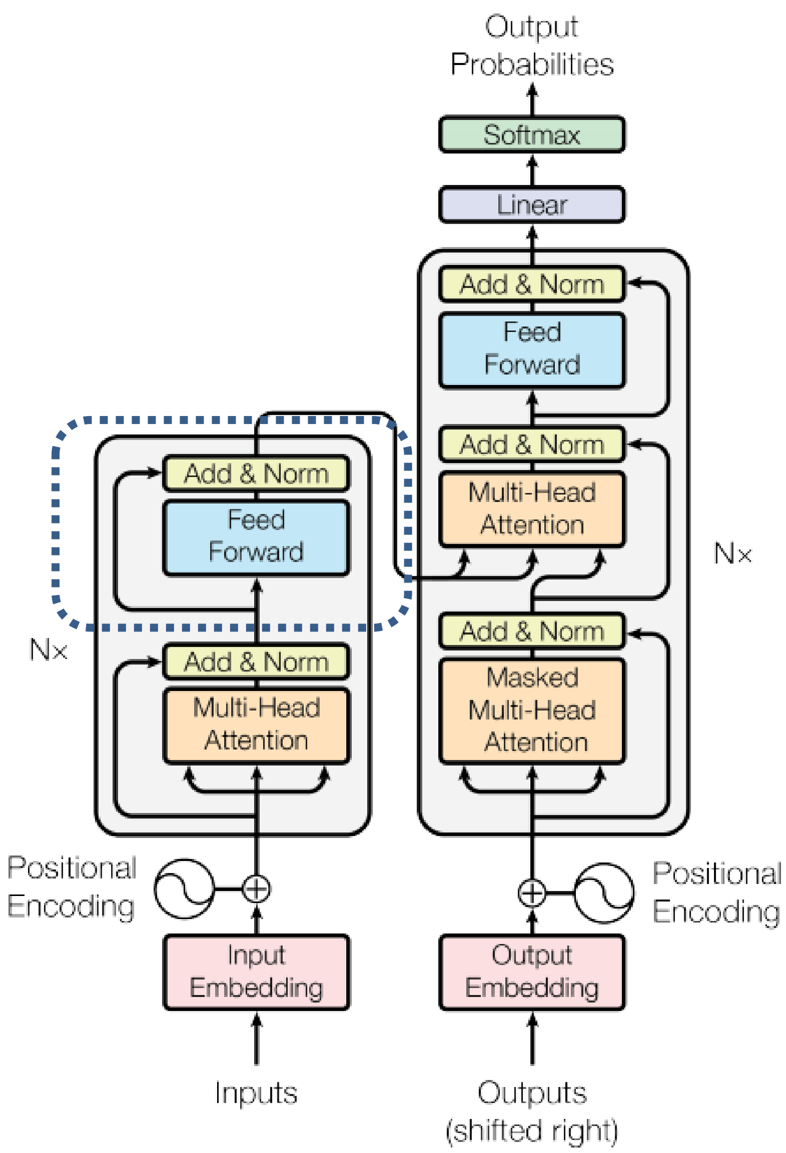


Figure 1: The Transformer - model architecture.

Transformer

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

Share embedding weights and the pre-softmax linear transformation (refer to arXiv:1608.05859)

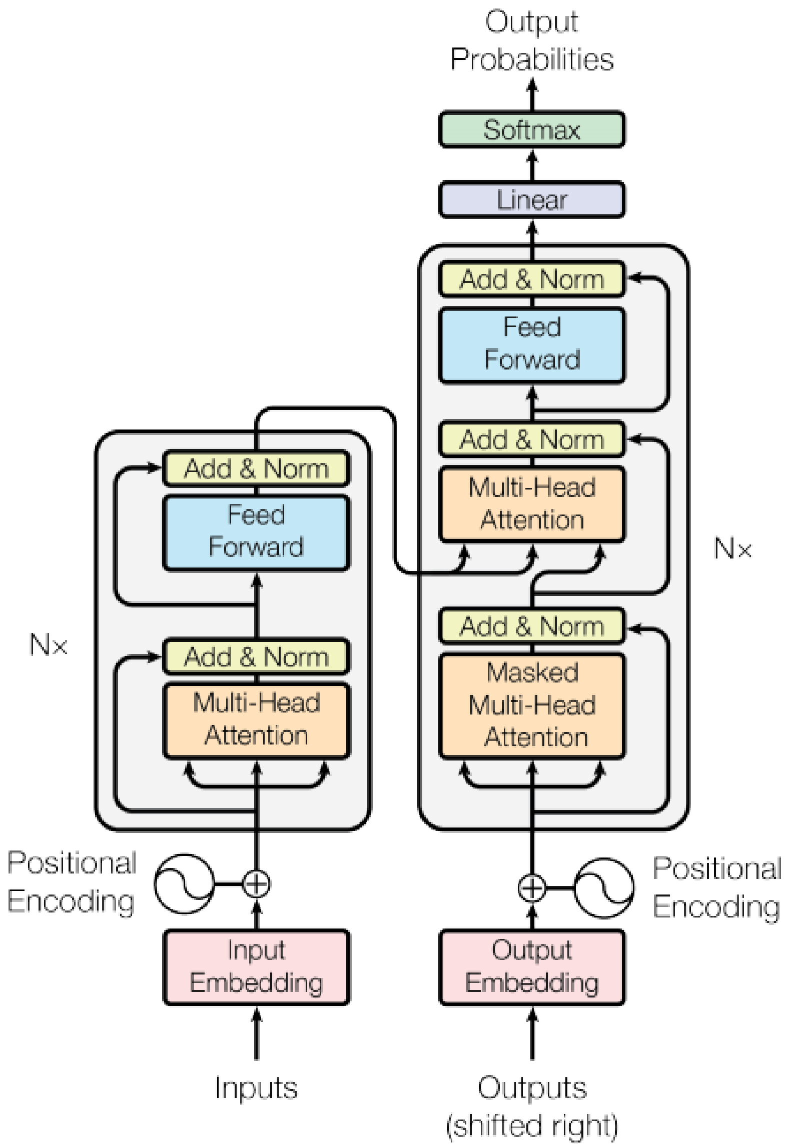


Figure 1: The Transformer - model architecture.

Transformer

- ➢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:

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

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

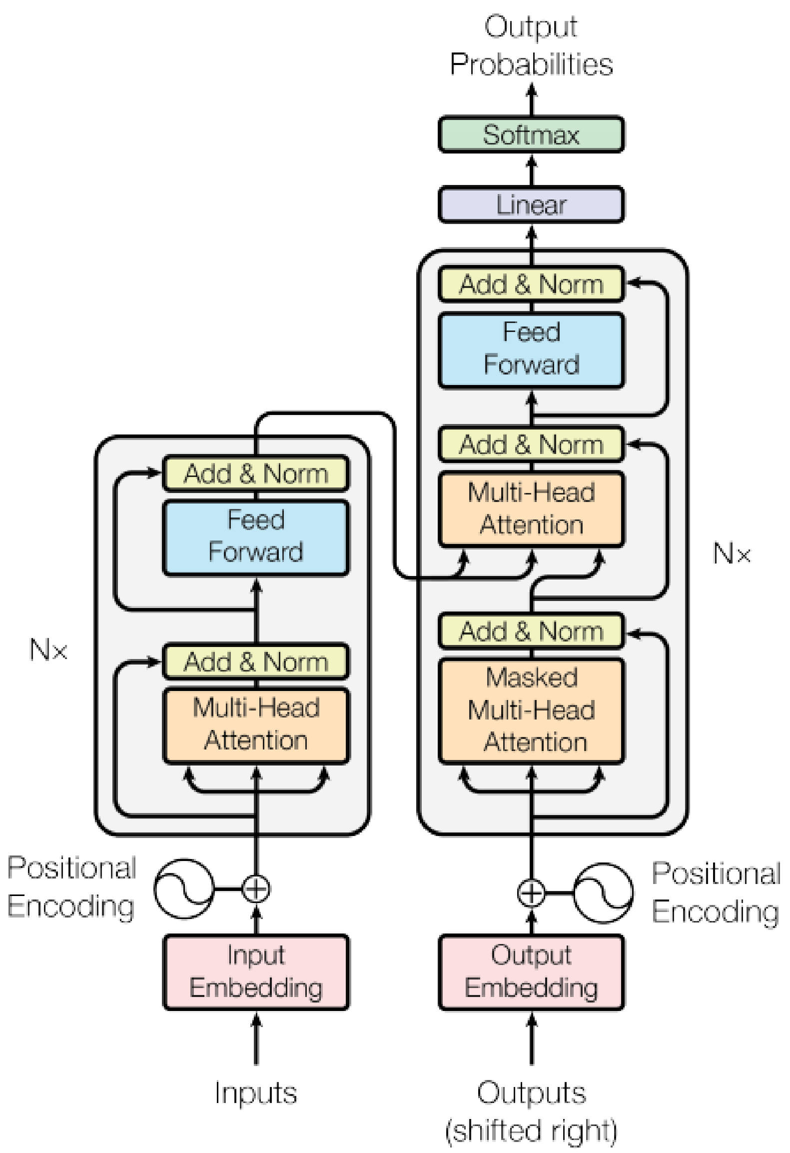


Figure 1: The Transformer - model architecture.

Transformer (Self Attention)

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

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Leng
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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Observation:

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

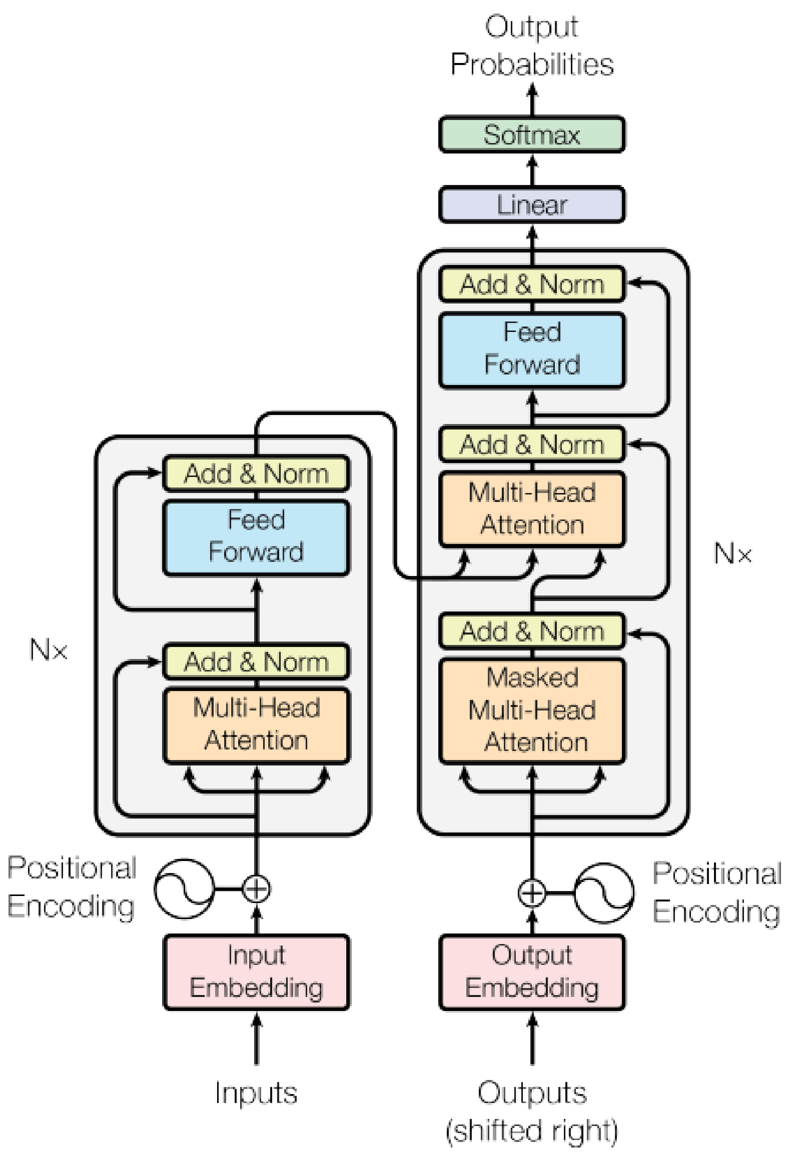


Figure 1: The Transformer - model architecture.

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 (increase over 2 BLEU)
- Eng-to-Fr: new single-model state-of-the-art (BLEU of 41.0)
- Less training cost

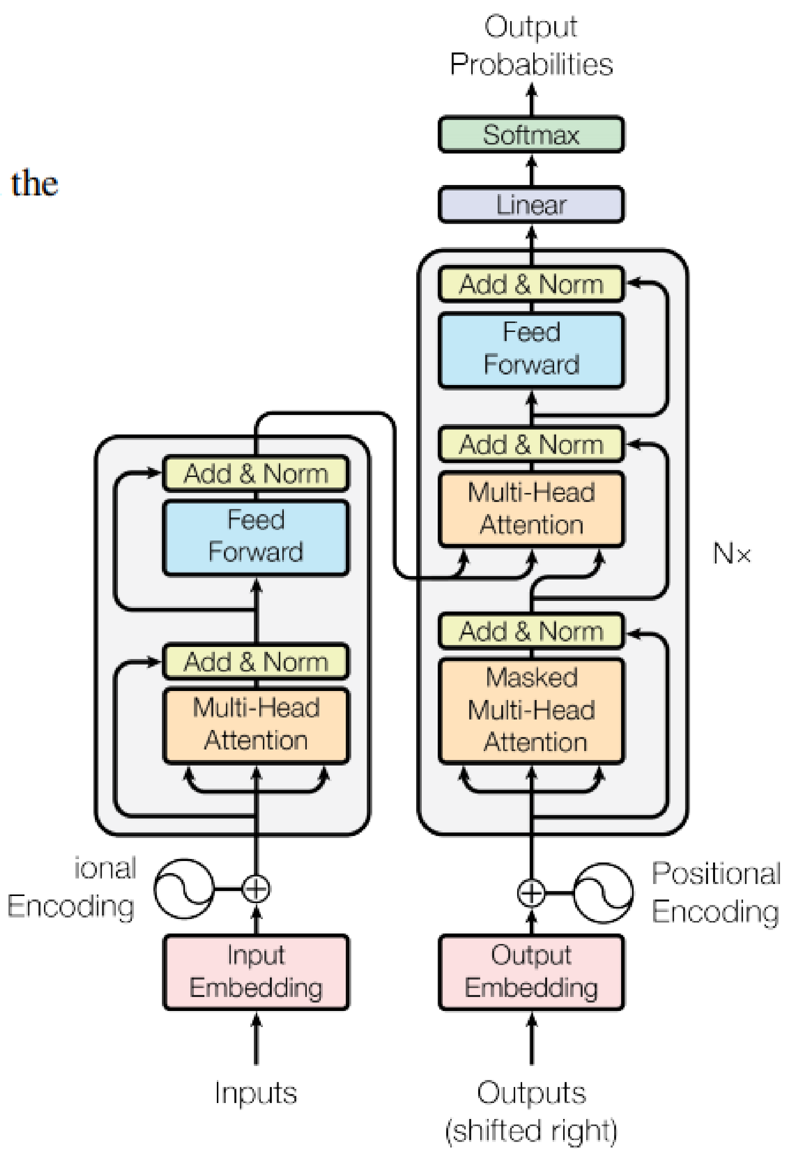


Figure 1: The Transformer - model architecture.

Transformer Performance

Some results:

- 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

- source: Wie würde eine solche Zukunft aussehen
- expected: What would such a future look like
- got: What would a future like this

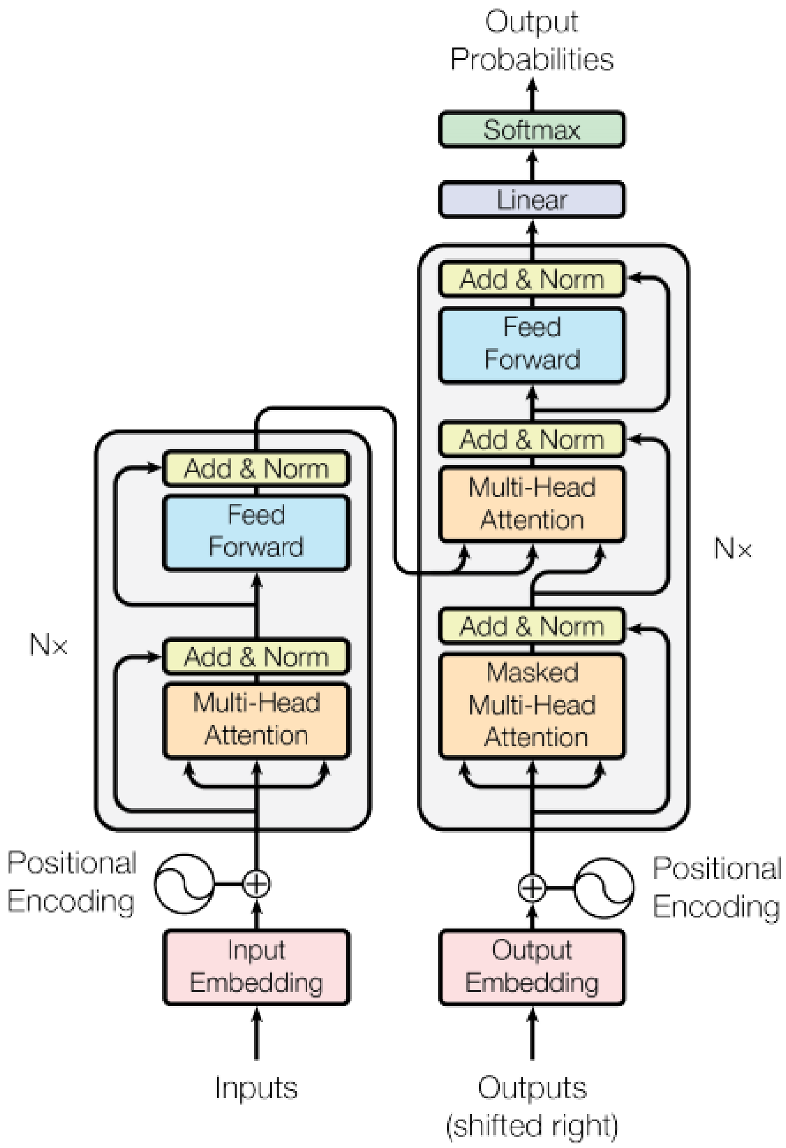


Figure 1: The Transformer - model architecture.

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/

The amazing images, and part of the content, about attention mechanisms are from

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