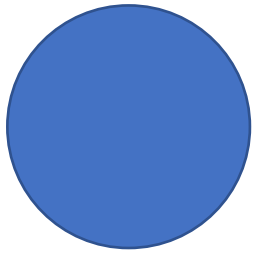
sequence modeling

midjourney bot (2023)

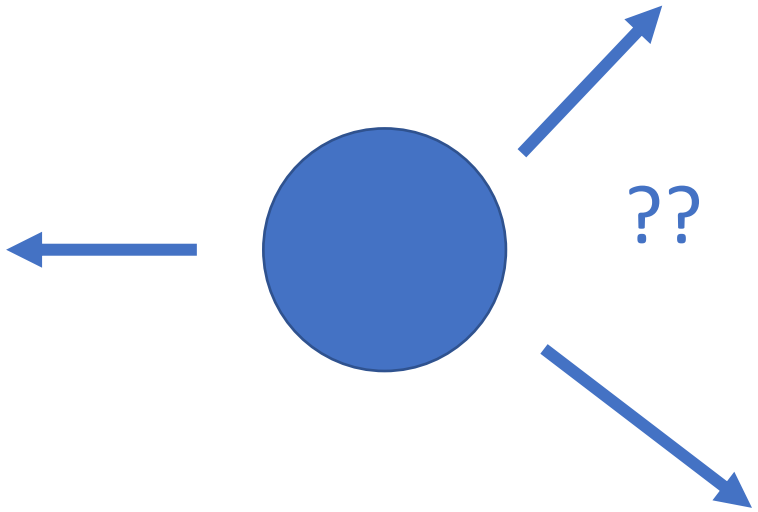
sequence modeling

Where does the ball go next?



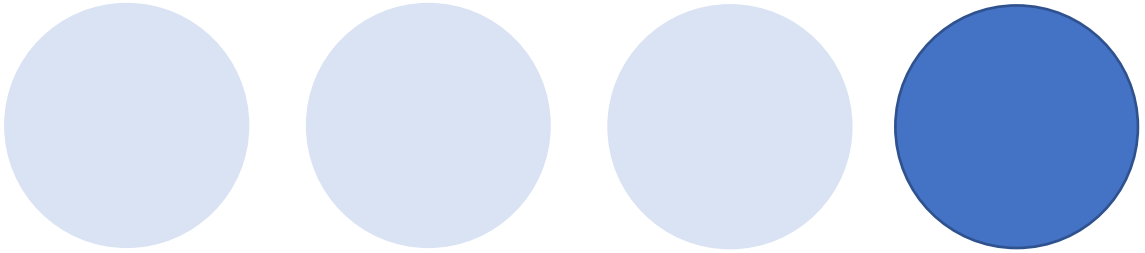
sequence modeling

Where does the ball go next?



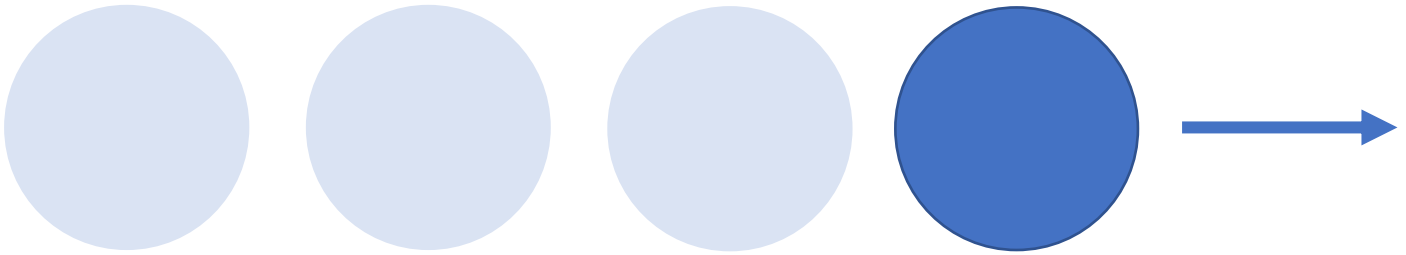
sequence modeling

Where does the ball go next?



sequence modeling

Where does the ball go next?



sequence modeling

text

video streams

audio streams

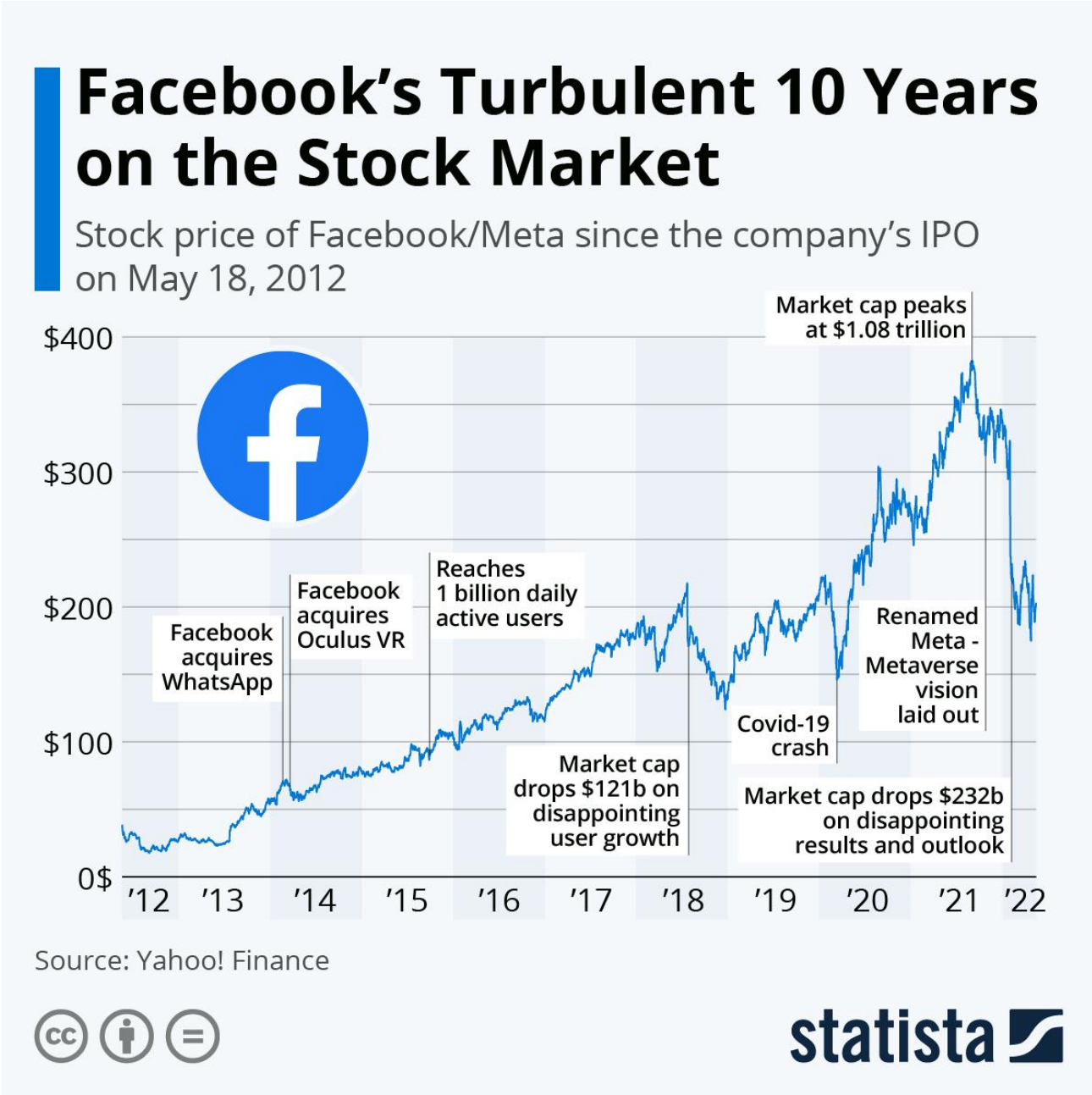
genomics, transcriptomics, proteomics

stock markets

weather, climate

electrocardiogram

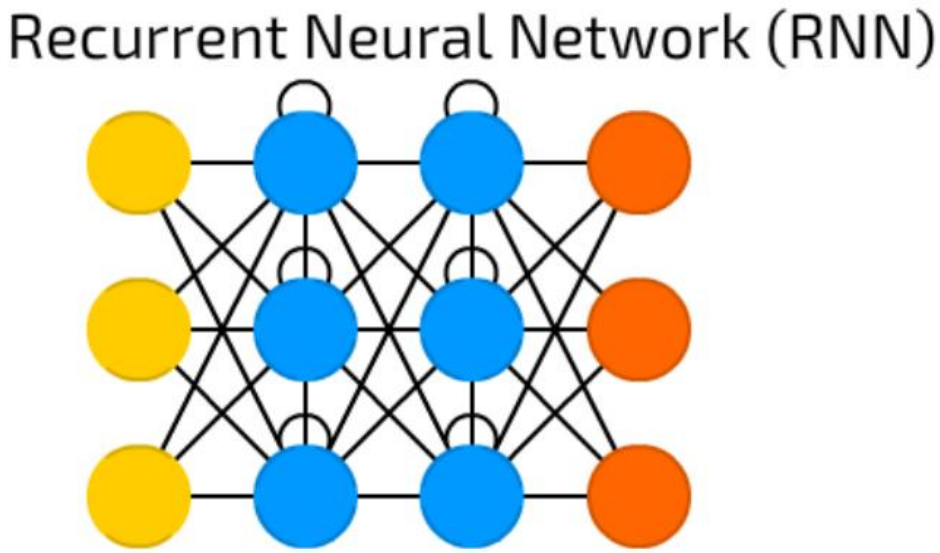
…



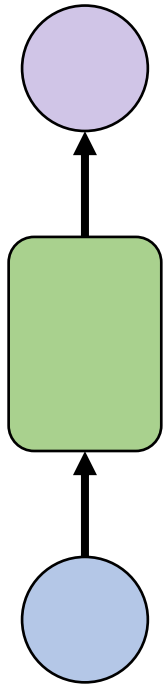
sequence modeling

Sequences can have variable length, which makes it hard to represent them as fixed length feature vectors.

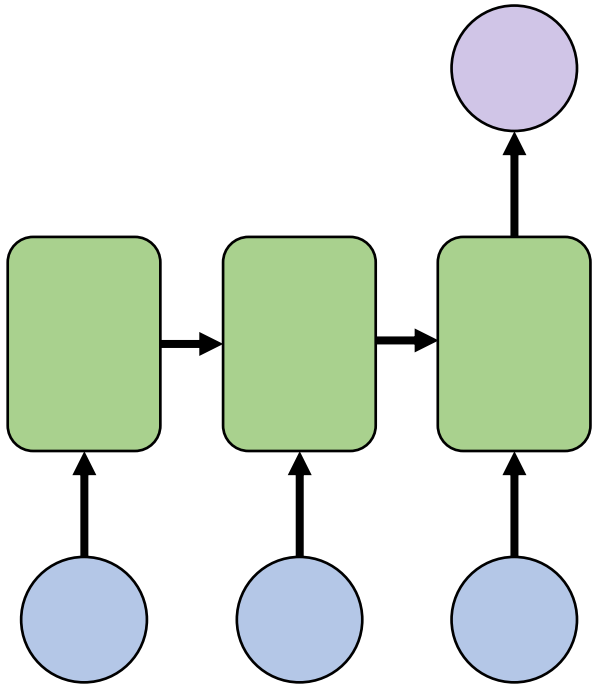
The **recurrent neural network** module is designed to tackle this issue.



sequence modeling



one to one



many to one

sentiment classification

protein function prediction

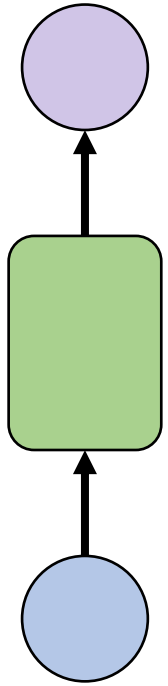
stock market price prediction

gene expression prediction

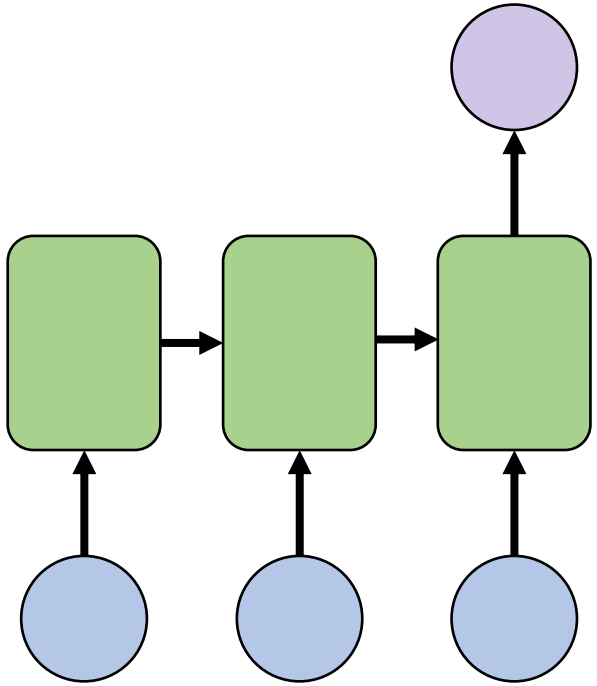
video topic classification

…

sequence modeling



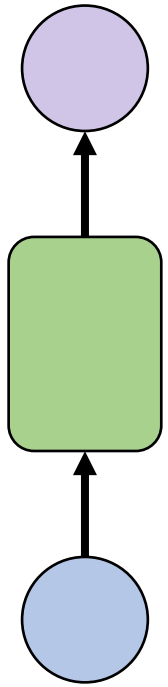
one to one



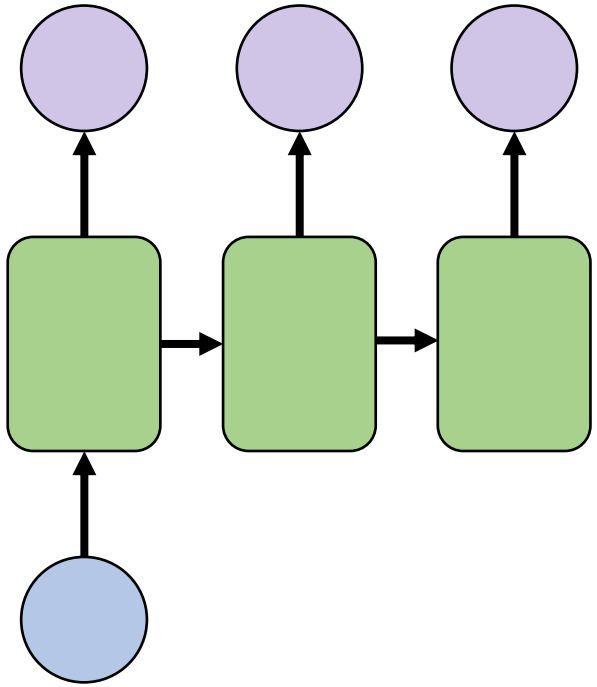
many to one

A classroom with students discussing the future of AI in healthcare, photorealistic.

sequence modeling



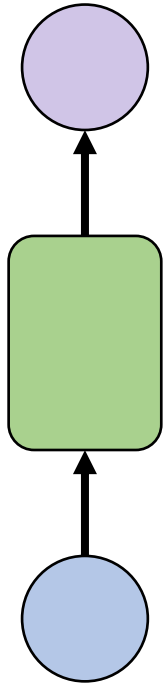
one to one



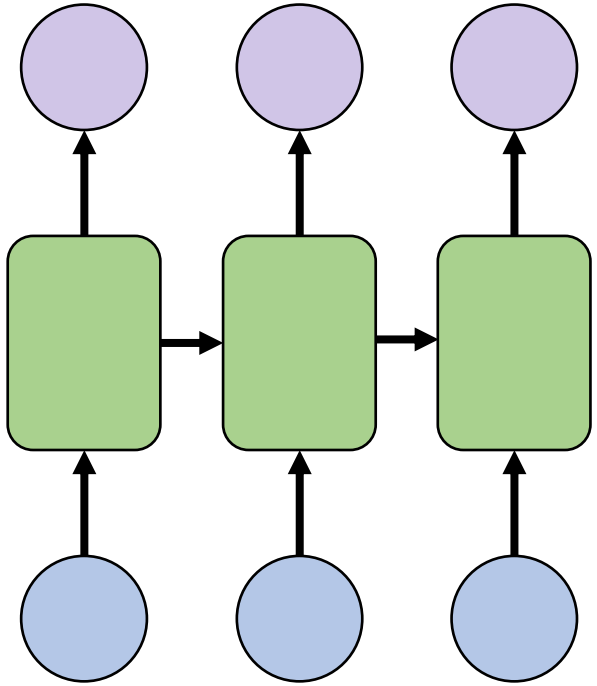
one to many

A baseball player throwing a ball.

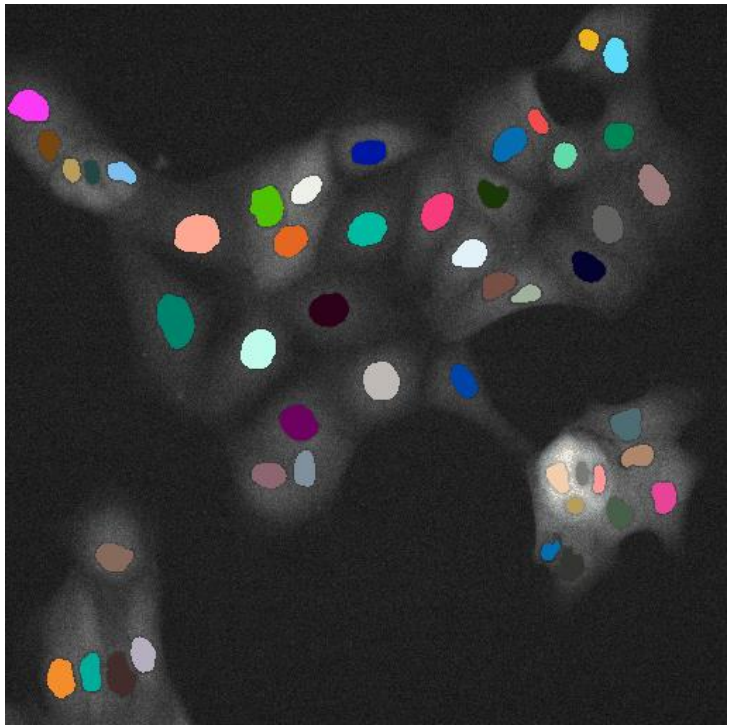
sequence modeling



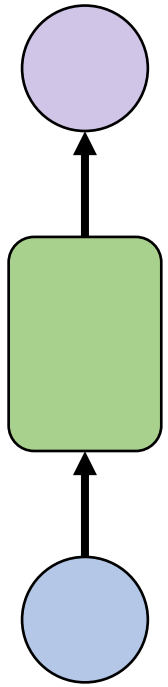
one to one



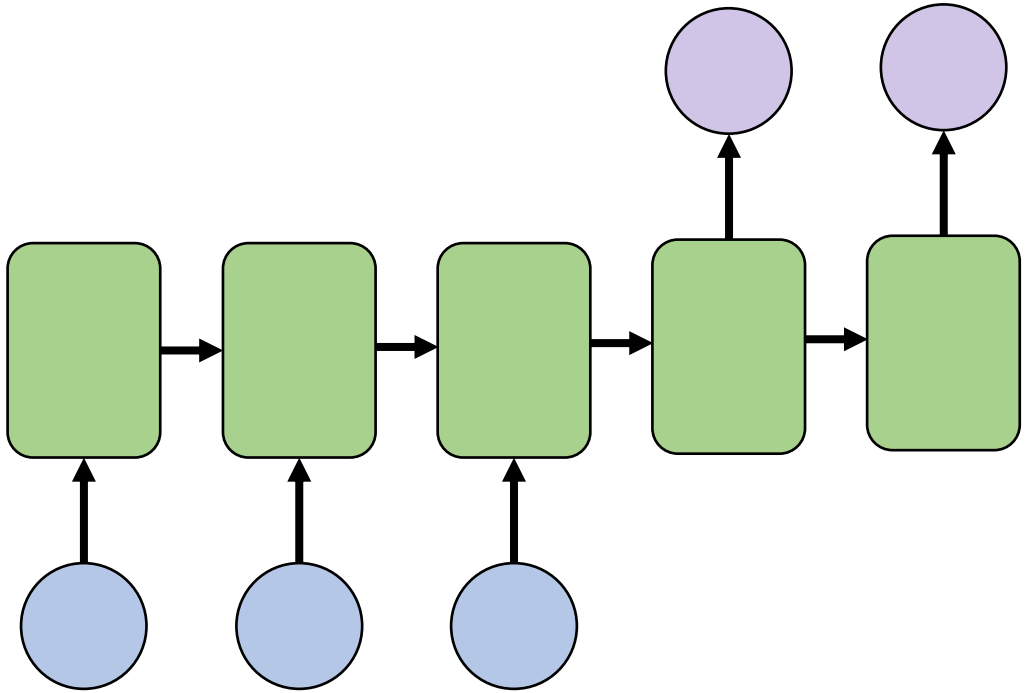
many to many



sequence modeling



one to one



many to many

text translation

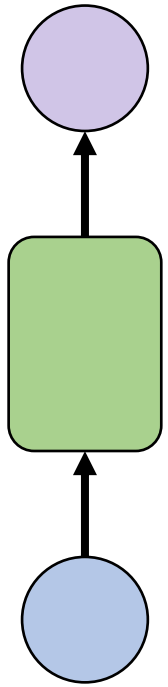
protein secondary structure prediction

MS/MS spectrum prediction

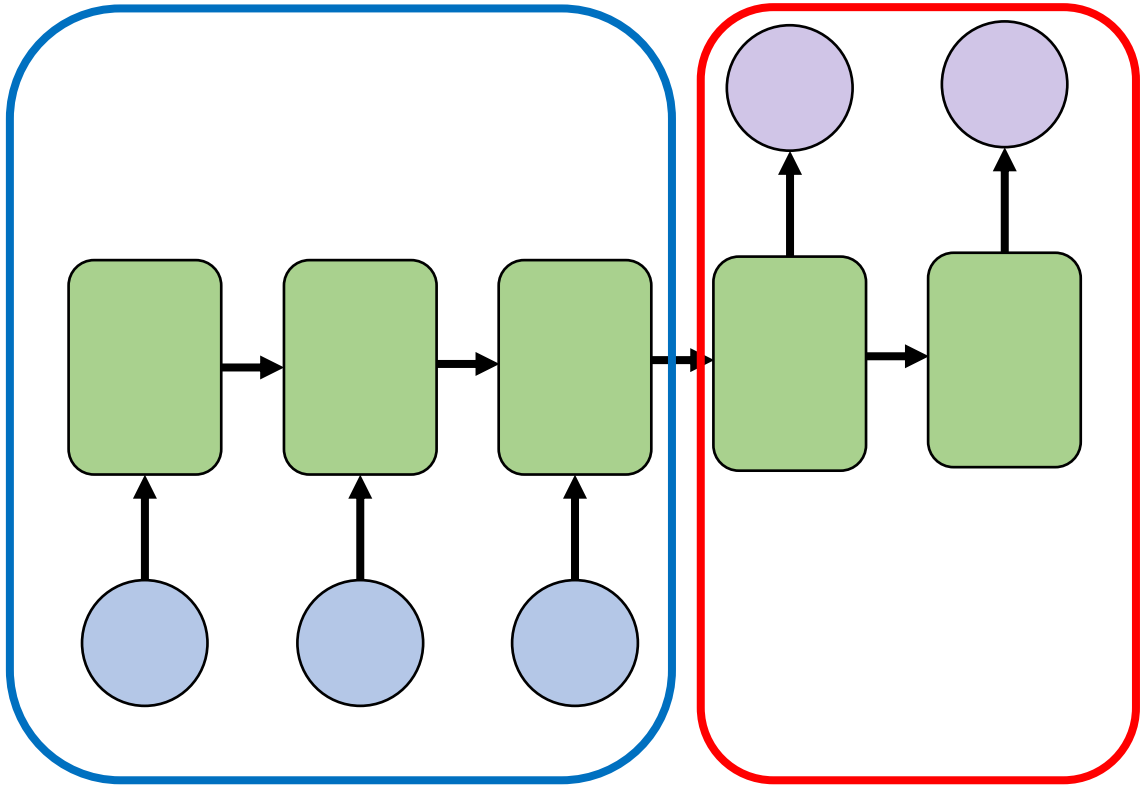
peptide identification

…

sequence modeling



one to one



encoder

many to many

decoder

text translation

protein secondary structure prediction

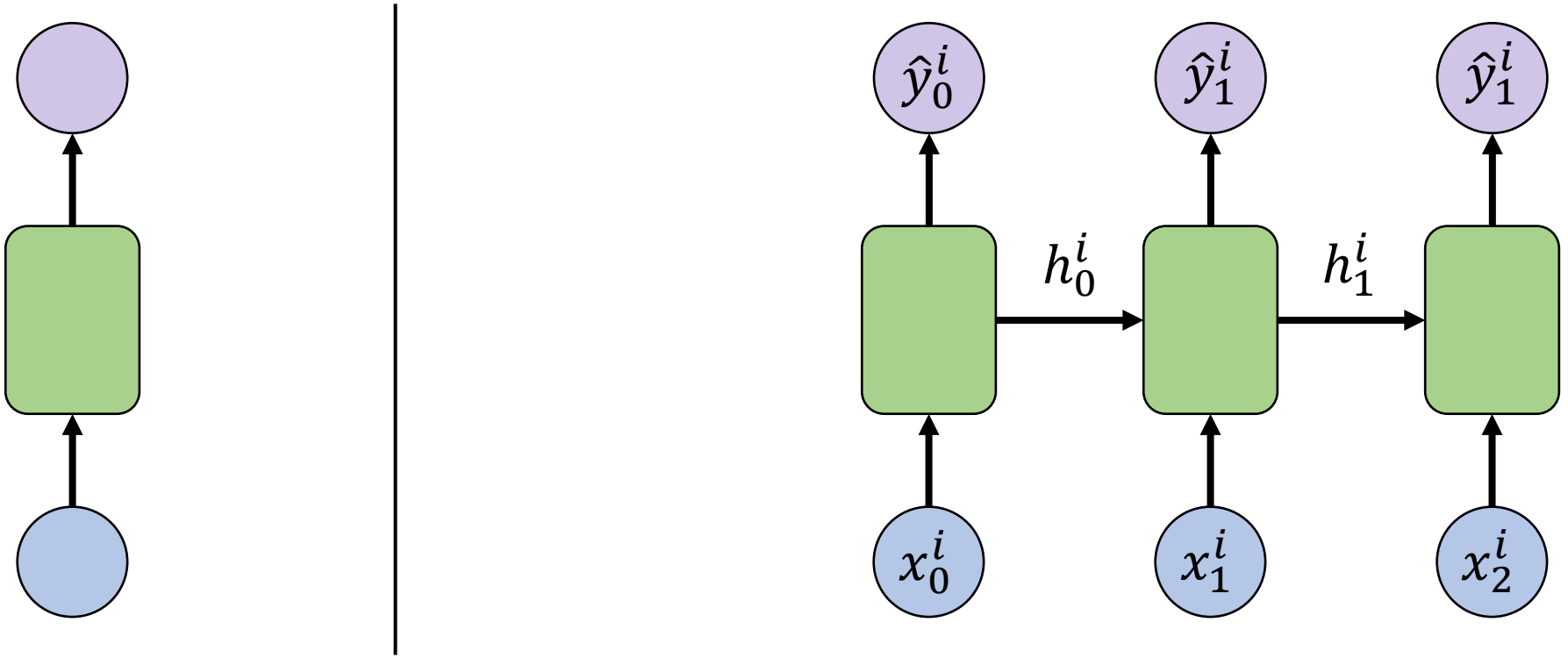
MS/MS spectrum prediction

peptide identification

…

recurrent neural network (RNN)

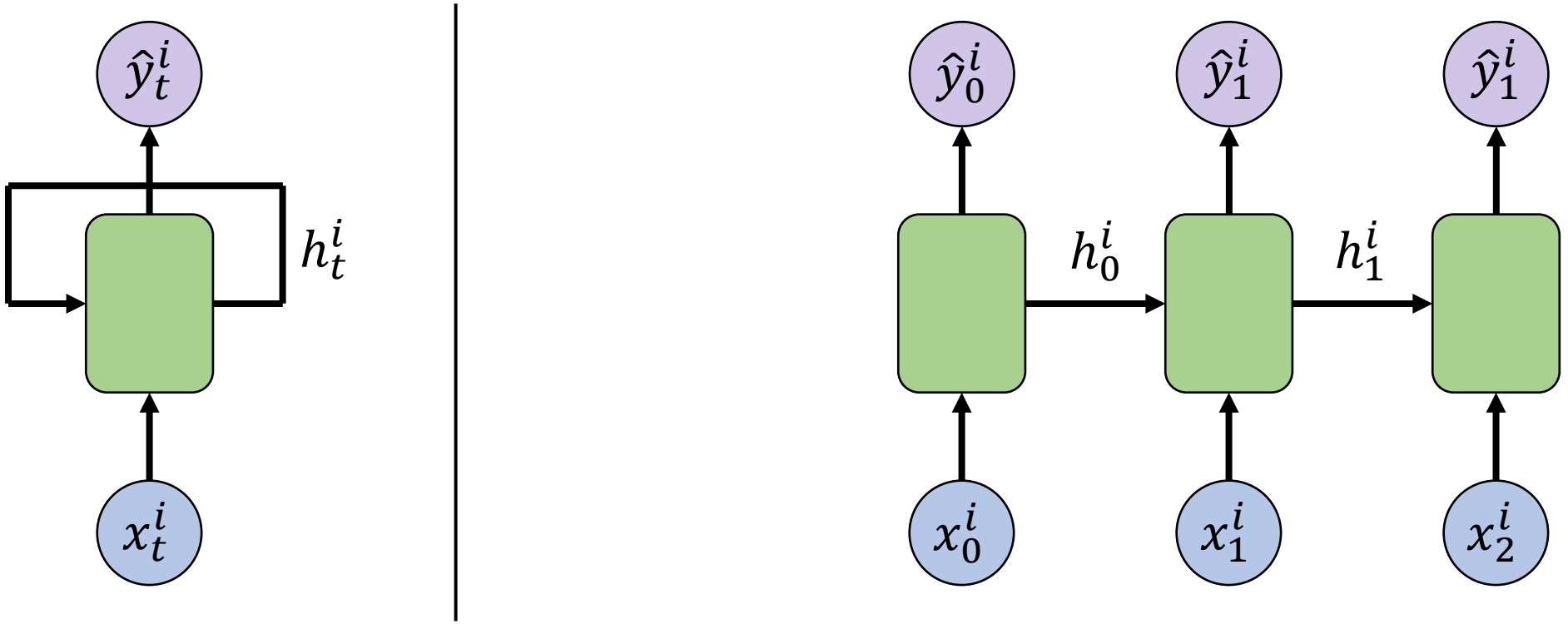
It is important to realize that the model parameters of the RNN are the same in each time-step, i.e. the green part in the diagram is always the same.



recurrent neural network (RNN)

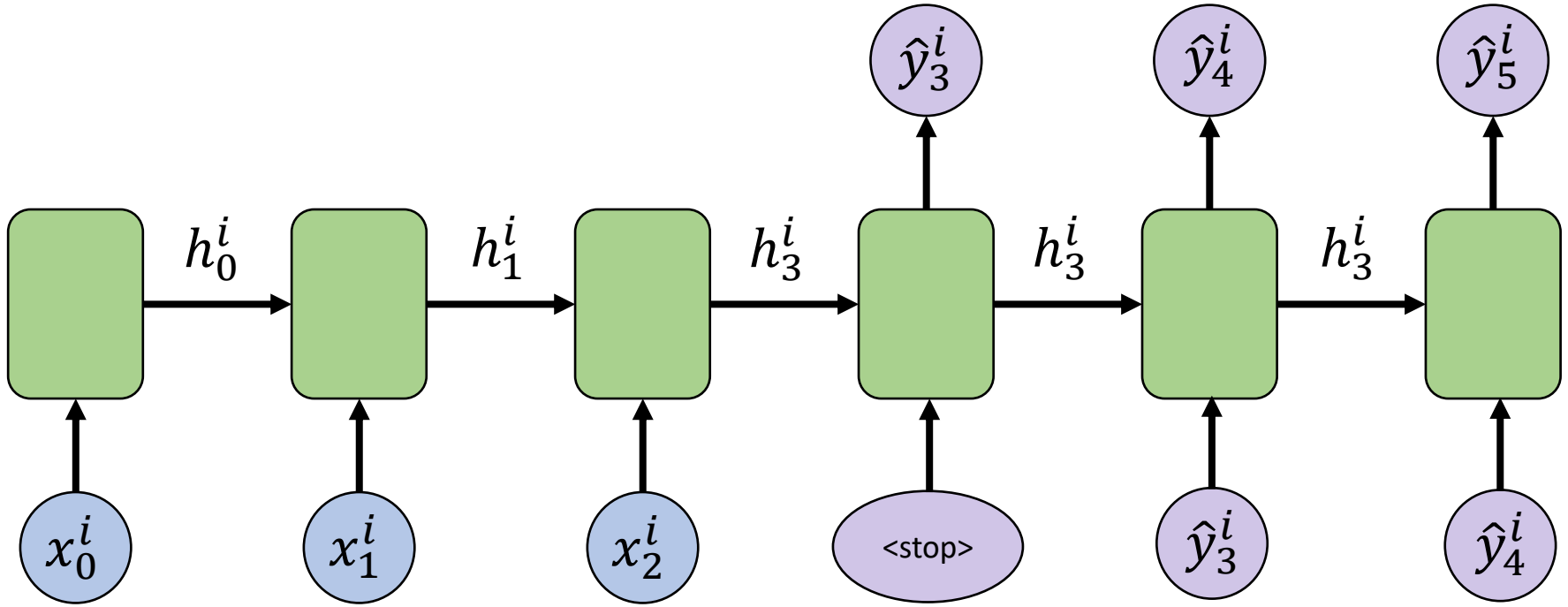
It is important to realize that the model parameters of the RNN are the same in each time-step, i.e. the green part in the diagram is always the same.

So, we can also represent the RNN as shown on the left.



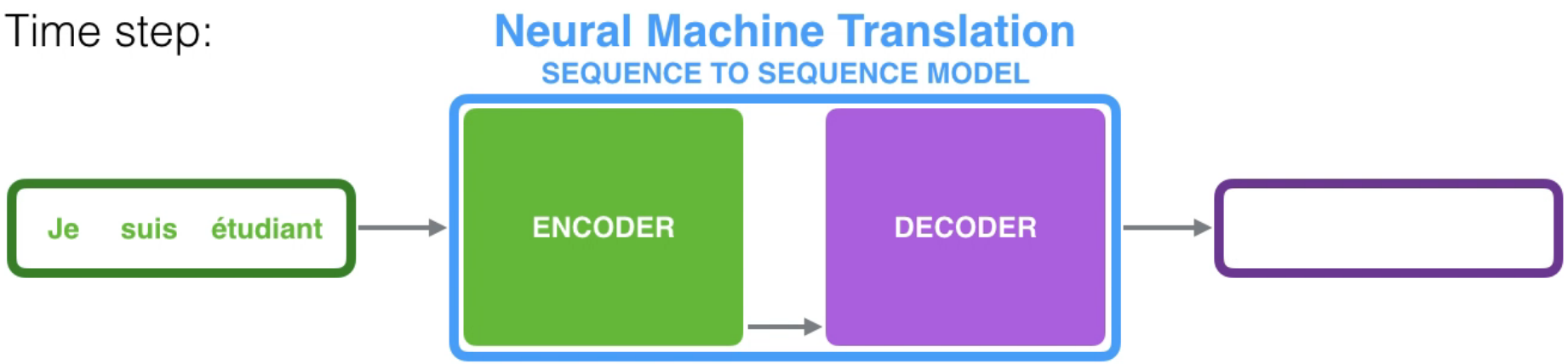
RNN encoding - decoding

A general case of an RNN is many to many with an encoder encoding the input sequence into a hidden state vector (h_3^i in the diagram below) and then applying the decoder to decode the hidden state vector into the output sequence.



RNN encoding - decoding

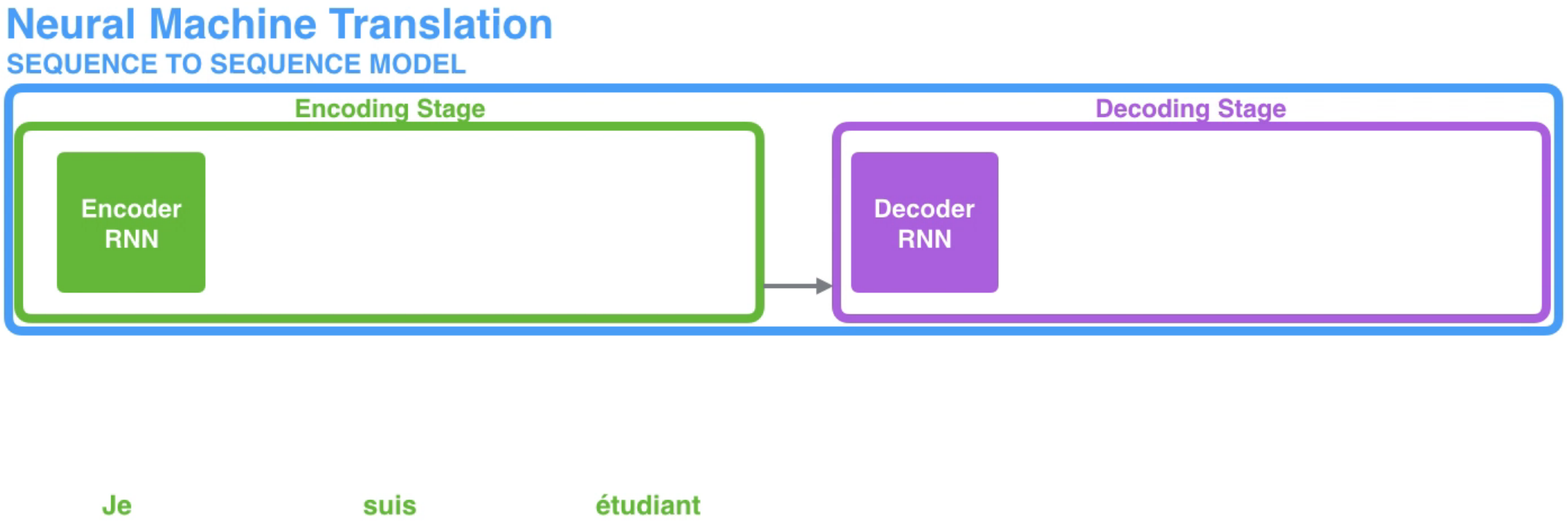
Folded this looks like this.



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

RNN encoding - decoding

Unfolded this looks like this.

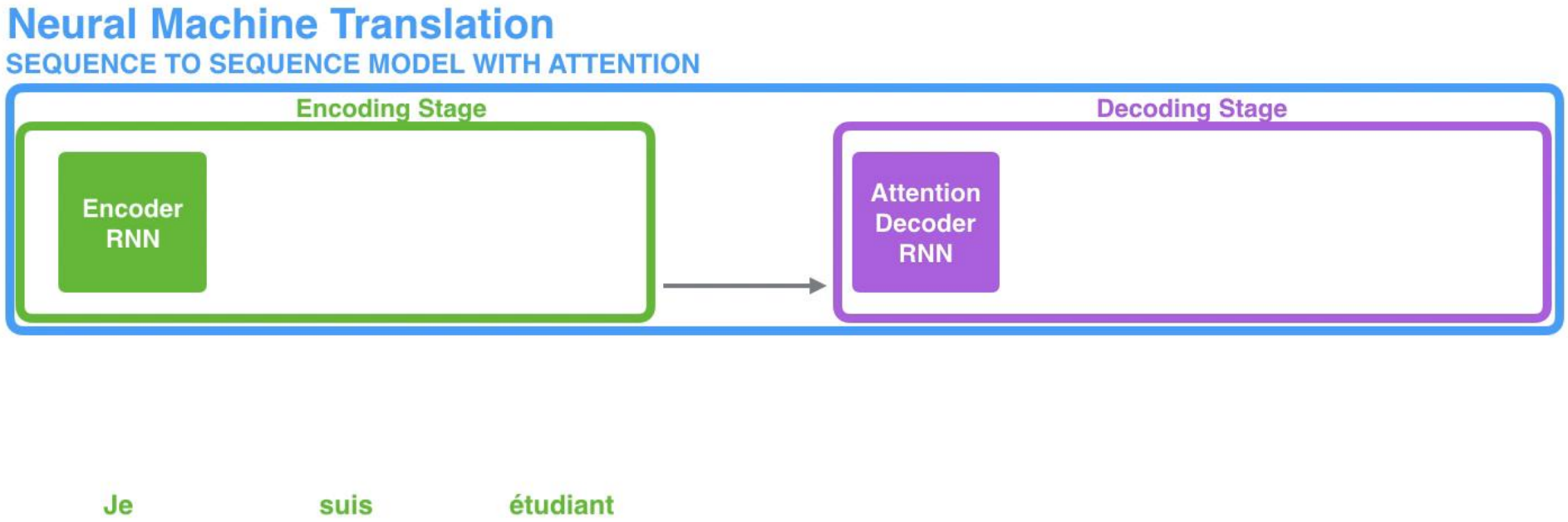


This creates a bottleneck in which all information in the input sequence needs to be encoded into the final hidden state vector.
This puts limitations to the length of the input sequence.

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

RNN encoding - decoding

One way to solve this issue is to use all hidden state vectors to decode.

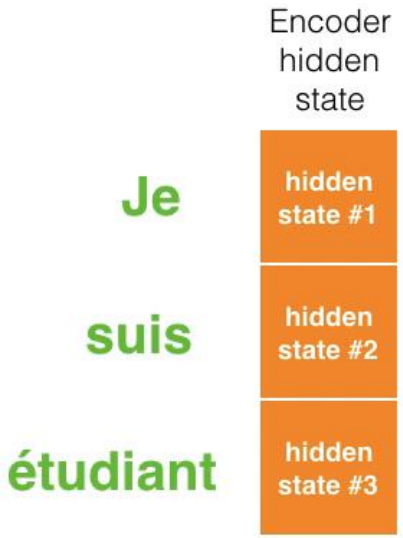


But this can create an information overflow during decoding.

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

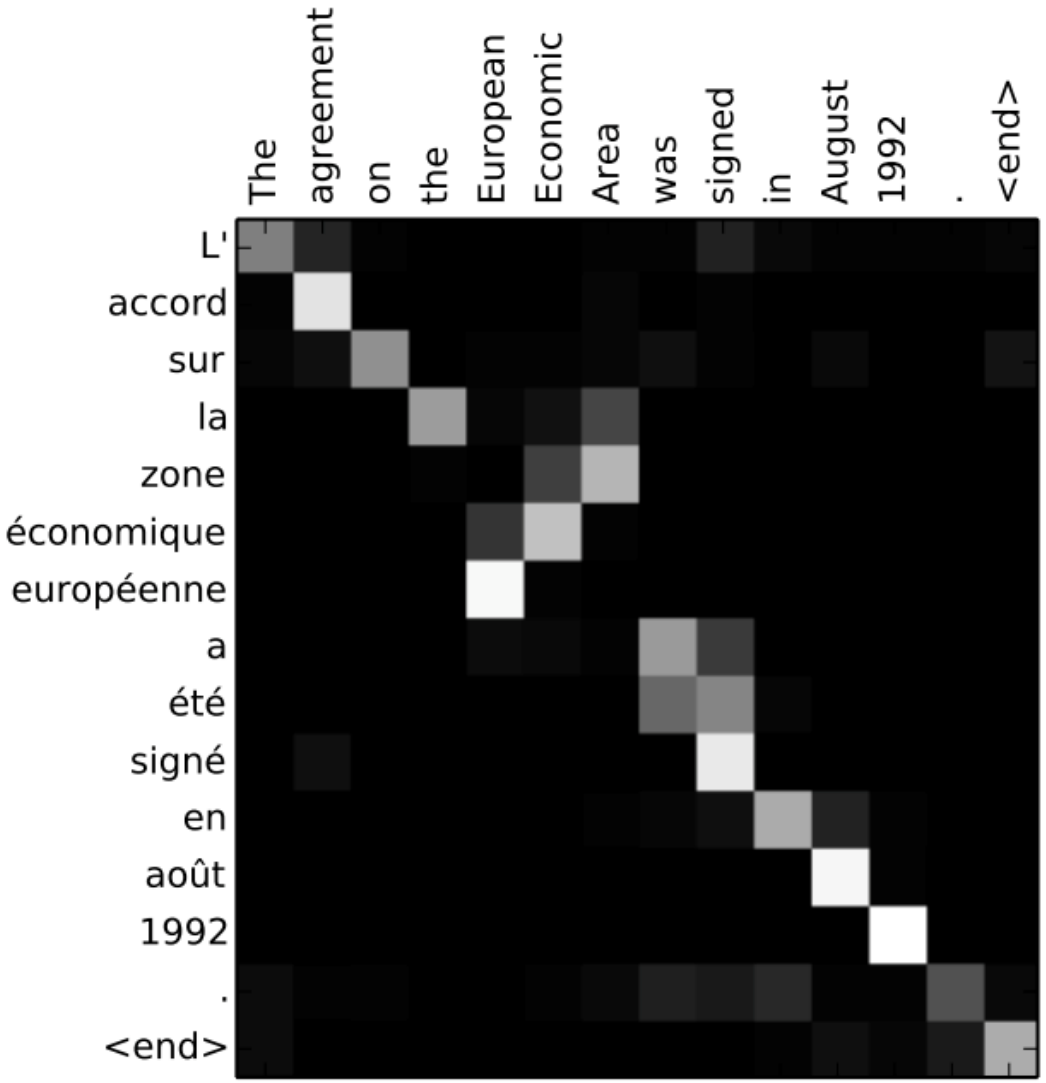
learning to pay attention

This issue can be solved by adding an **attention layer**. This is again a specialized architecture with model parameters that learns to pay more or less attention to the specific hidden state vectors during each time-step in the decoding.



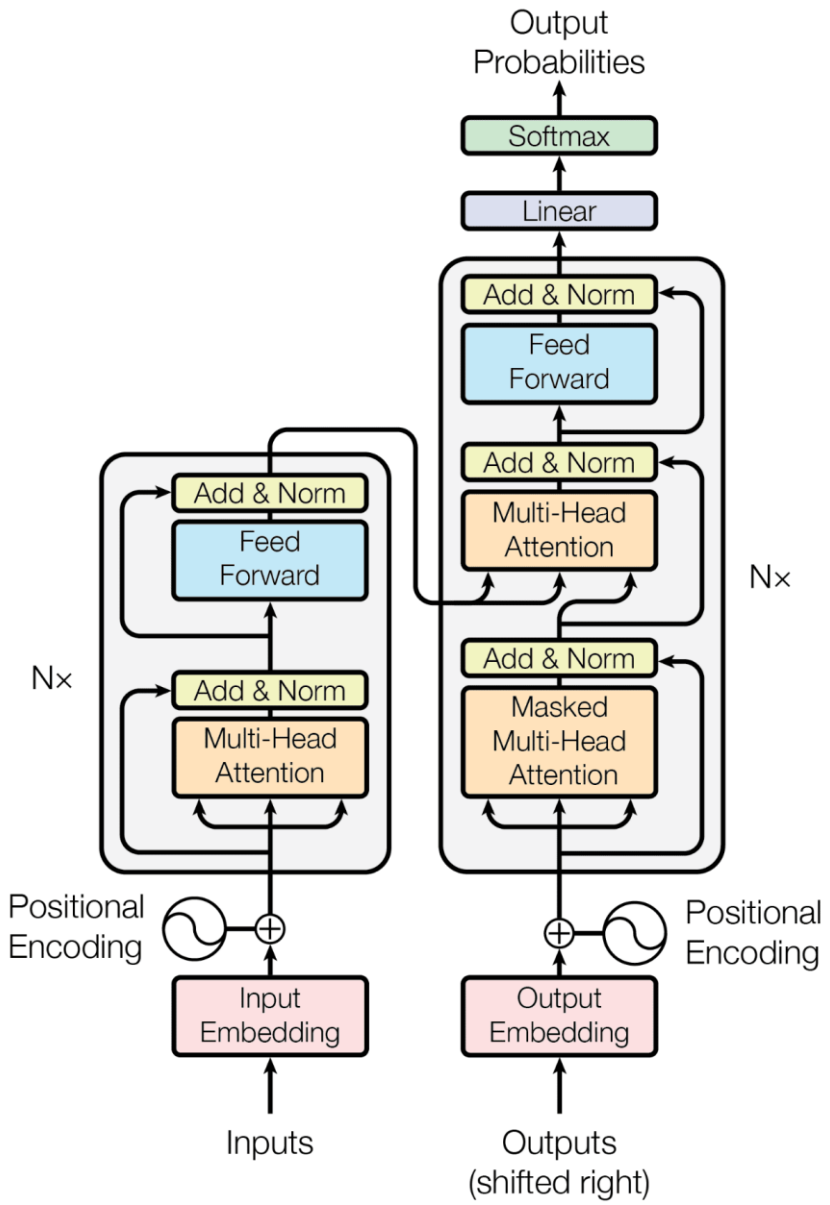
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

learning to pay attention



attention is all you need

The transformer architecture is the current state of the art (I think…).

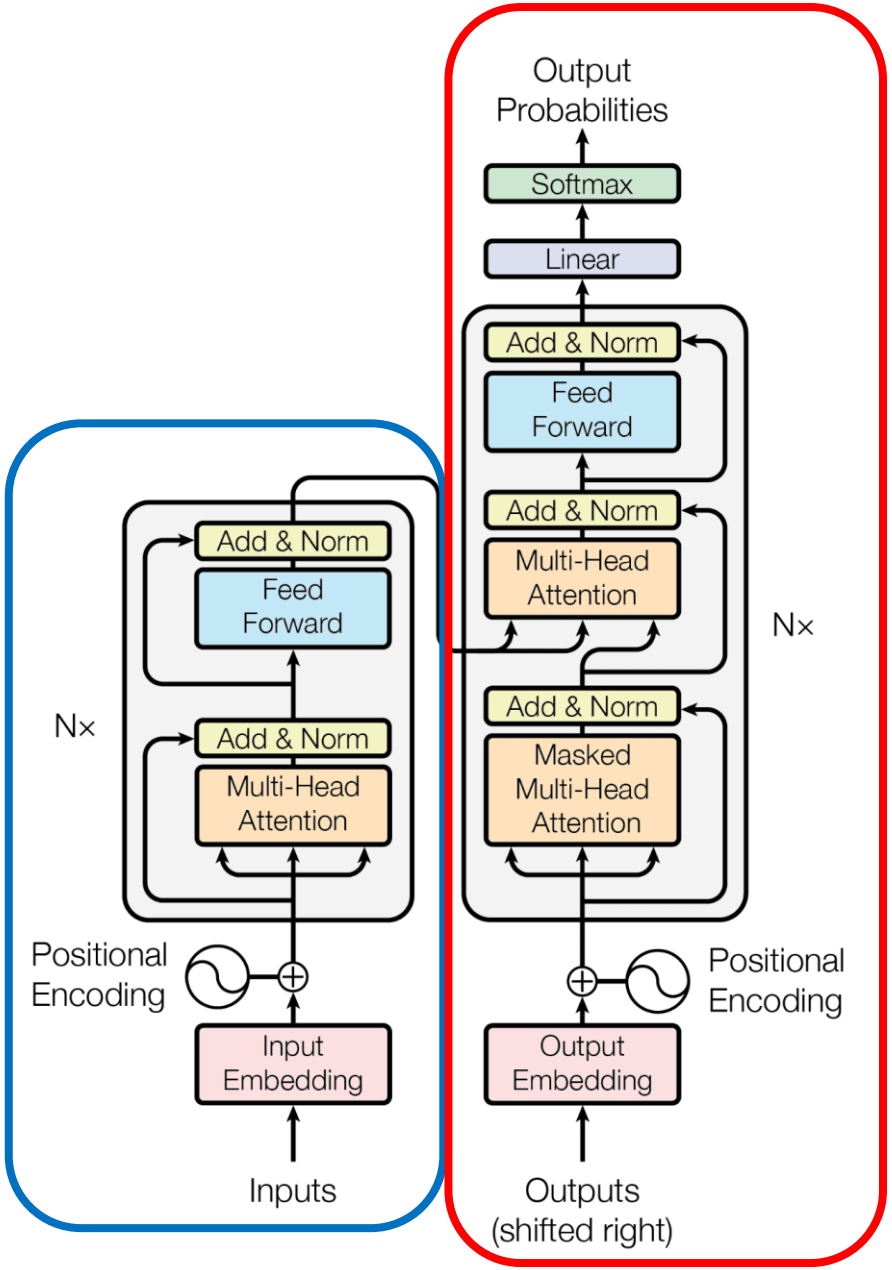


Ashish Vaswani et al., Attention is all you need. (NIPS'17)

Transformers

The transformer architecture is the current state of the art (I think…).

It consist of an encoder and a decoder.



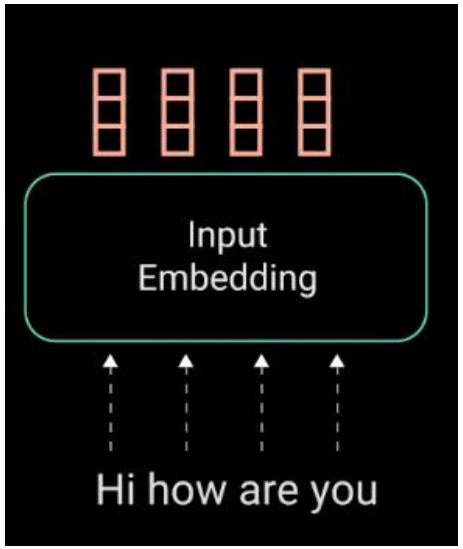
Ashish Vaswani et al., Attention is all you need. (NIPS'17)

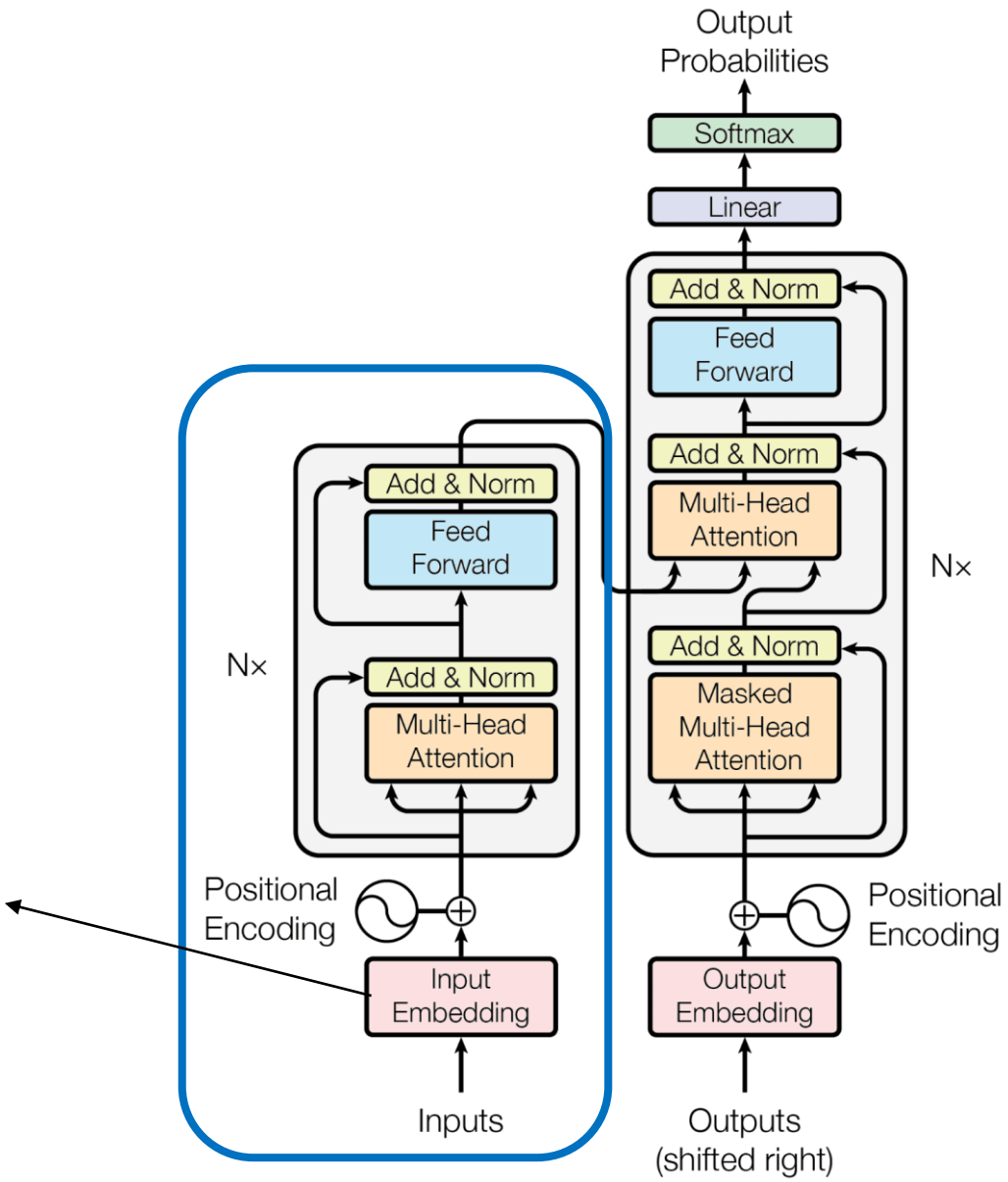
Transformers

The transformer architecture is the current state of the art (I think…).

It consist of an encoder and a decoder.

The encoder takes a sequence of tokens (each possible input represented by a different number) as input and learns an embedding for each of the tokens.





Ashish Vaswani et al., Attention is all you need. (NIPS'17)

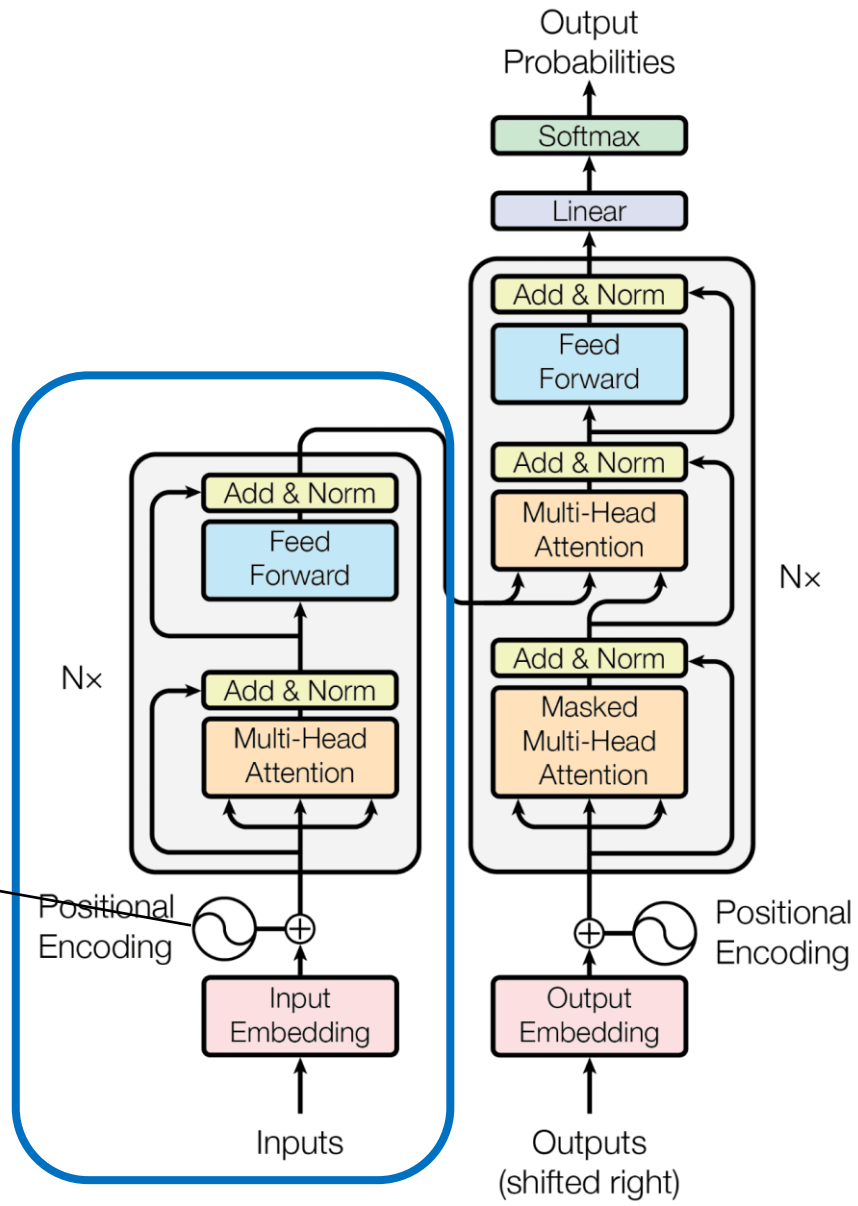
Transformers

The transformer architecture is the current state of the art (I think…).

It consist of an encoder and a decoder.

The encoder takes a sequence of tokens (each possible input represented by a different number) as input and learns an embedding for each of the tokens.

There is no recurrence, a special positional encoding is added to the token embeddings.

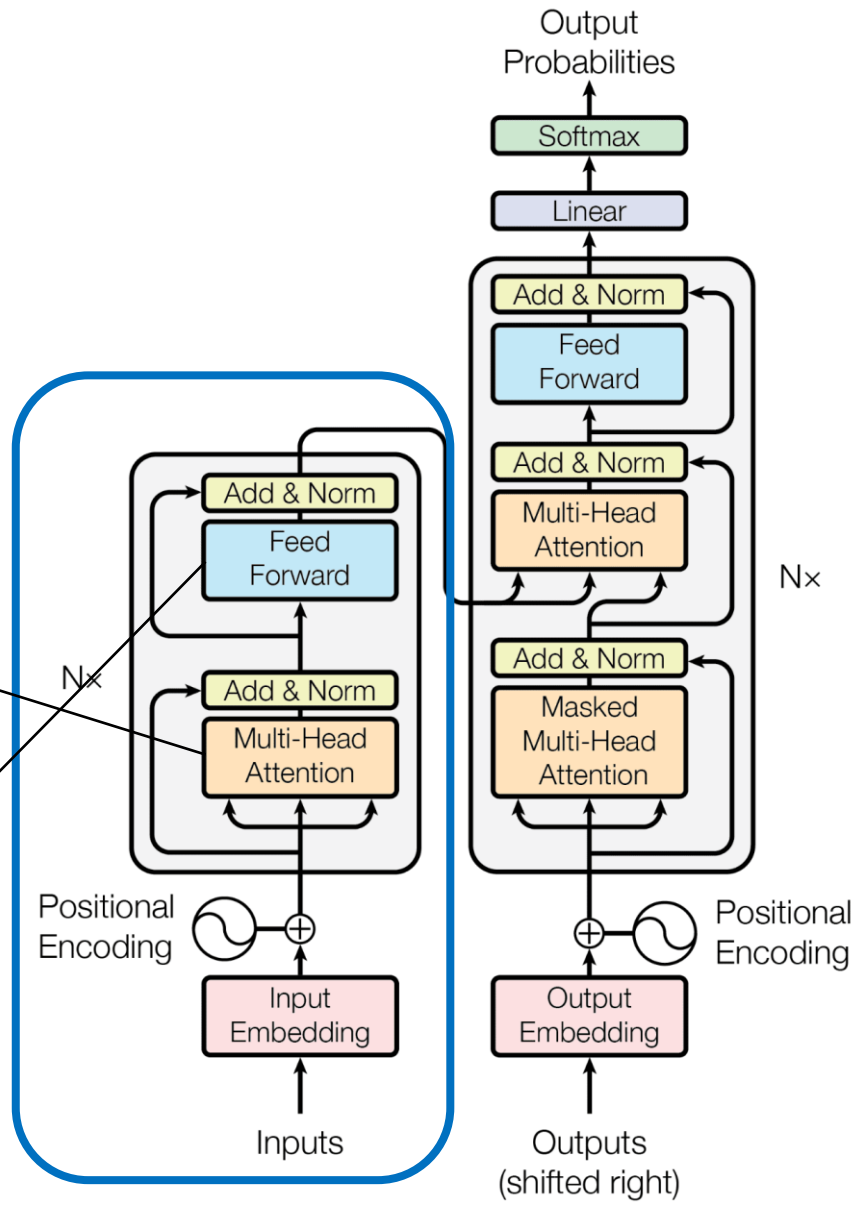


Ashish Vaswani et al., Attention is all you need. (NIPS'17)

Transformers

Multi-head (self-)attention learns to score each token embedding in order to compute the correct output (decoder).

	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.6	0.1
you	0.1	0.3	0.3	0.3



The scores and the token embeddings are then multiplied and passed trough a neural network module to add non-linearity.
This is repeated N times to produce a final embedding.

Ashish Vaswani et al., Attention is all you need. (NIPS'17)

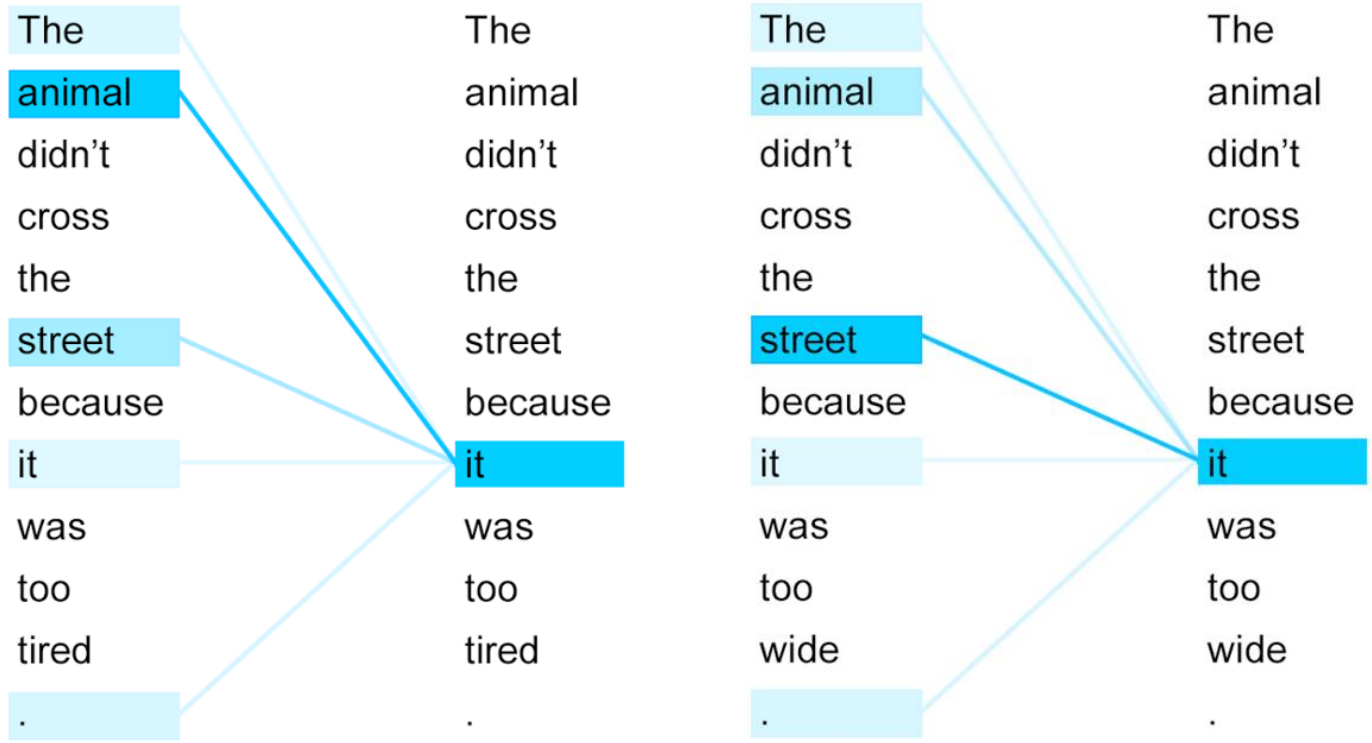
Transformers

Multi-head (self-)attention learns to score each token embedding in order to compute the correct output (decoder).

The animal didn't cross the street because it was too tired.
L'animal n'a pas traversé la rue parce qu'il était trop fatigué.

The animal didn't cross the street because it was too wide.
L'animal n'a pas traversé la rue parce qu'elle était trop large.

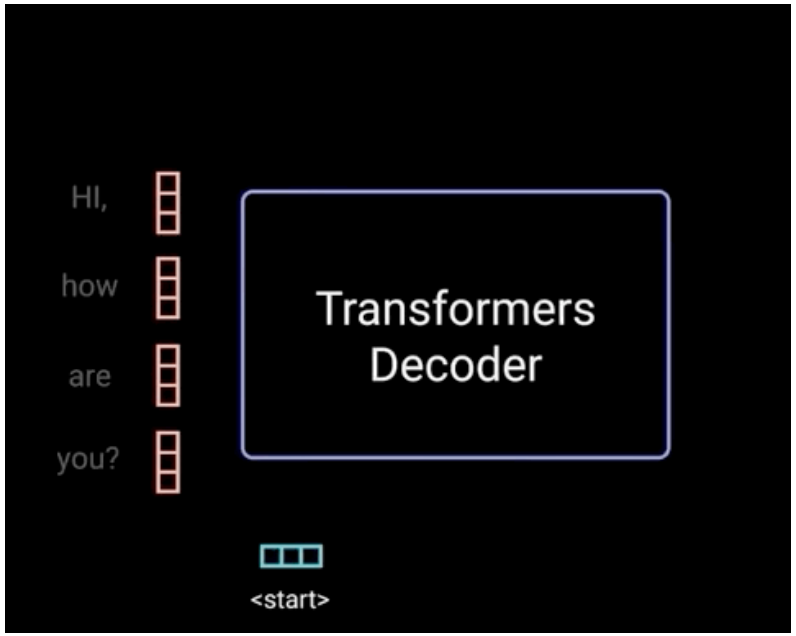
.

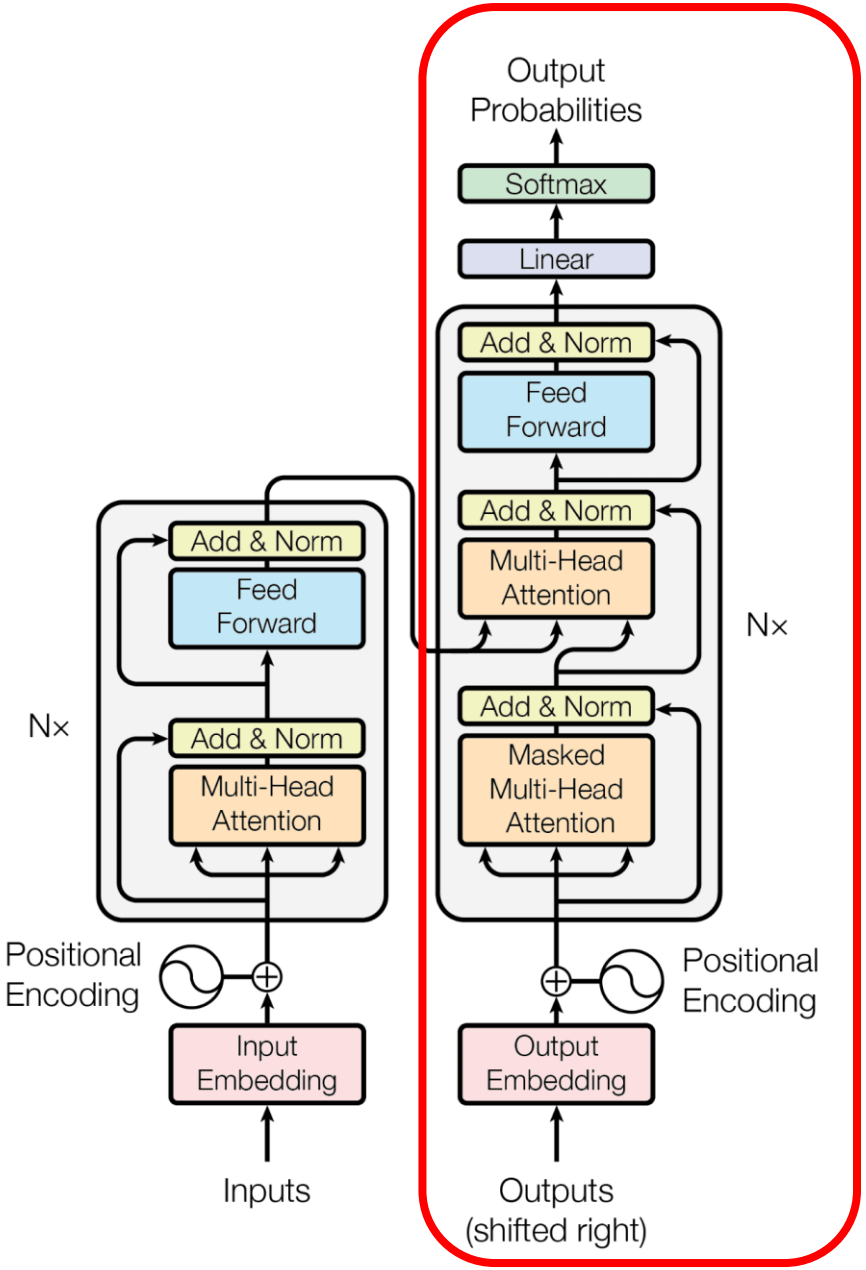


Transformers

The decoder learns to generate tokens.

It is autoregressive, it begins with a start token, and it takes in a list of previous outputs as inputs, as well as the encoder outputs that contain the attention information from the input.





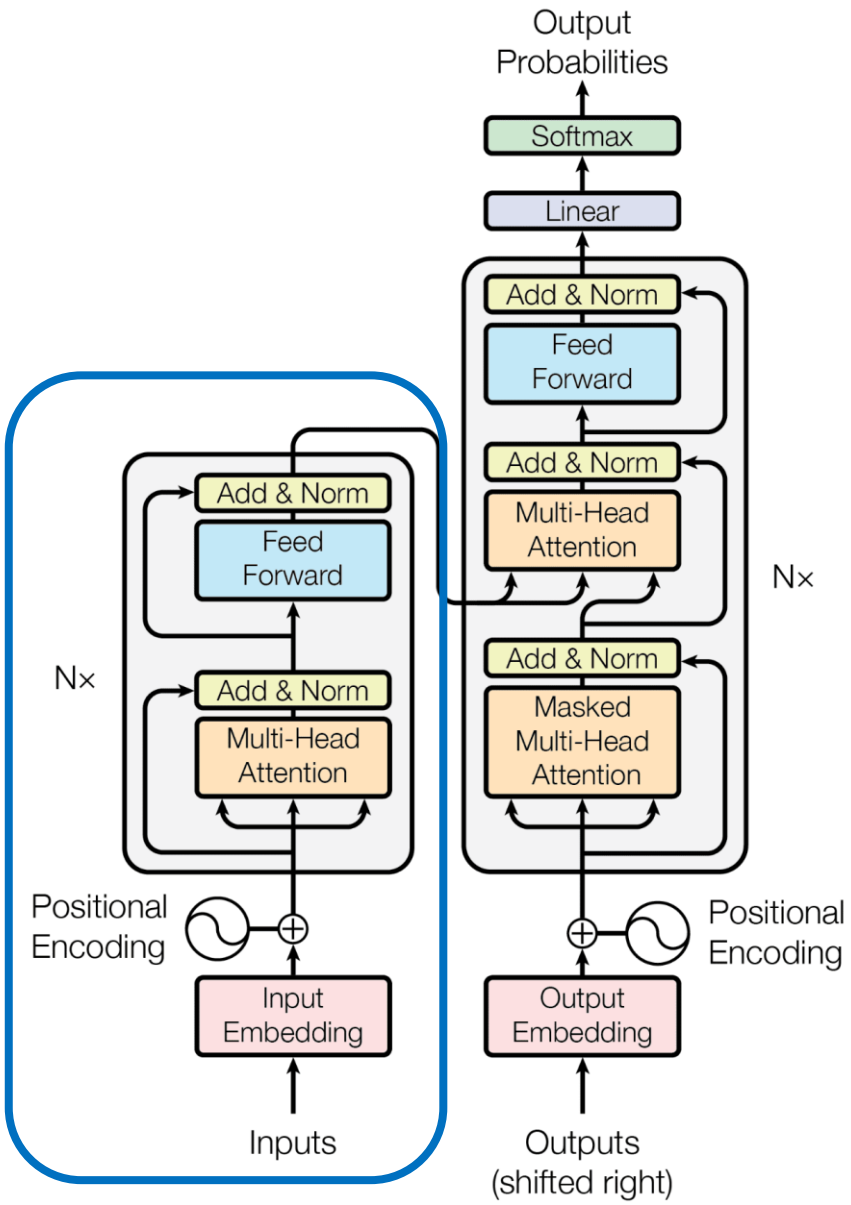
Ashish Vaswani et al., Attention is all you need. (NIPS'17)

Transformers

Some models only use the encoder (e.g. Bidirectional Encoder Representations from Transformers or BERT).

The learned sequence embedding is used as a feature vector representation for specific prediction tasks, e.g. protein function or stability prediction.

The sequence embedding is learned by self-supervision using masked tokens. For instances masked amino acids on protein sequences.



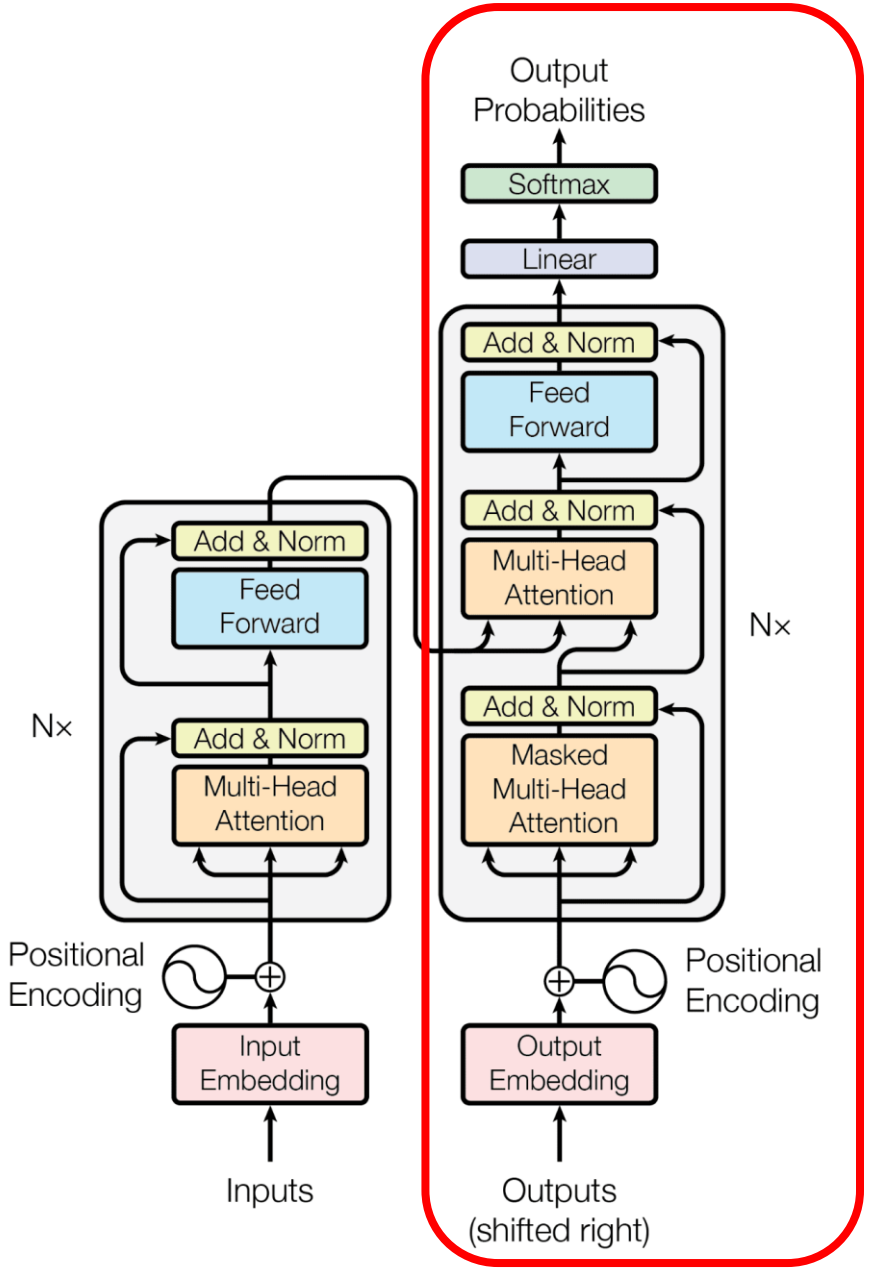
Ashish Vaswani et al., Attention is all you need. (NIPS'17)

Transformers

Other models only use the decoder.

An example is the Generative Pre-training Transformer (GPT). It is trained to predict (a probability distribution of) the next token based on the previous tokens.

By iteratively adding the predicted token to the input the model is able to generate a sequence of tokens.



Ashish Vaswani et al., Attention is all you need. (NIPS'17)

chatGPT

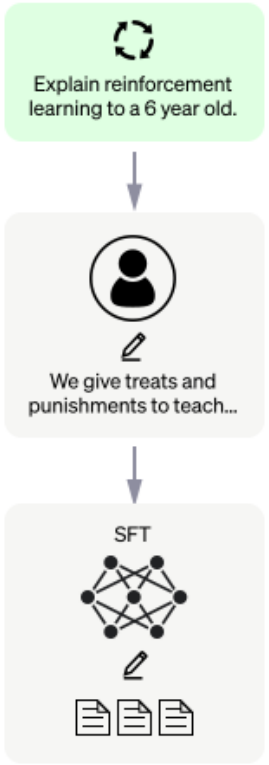
Step 1

Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



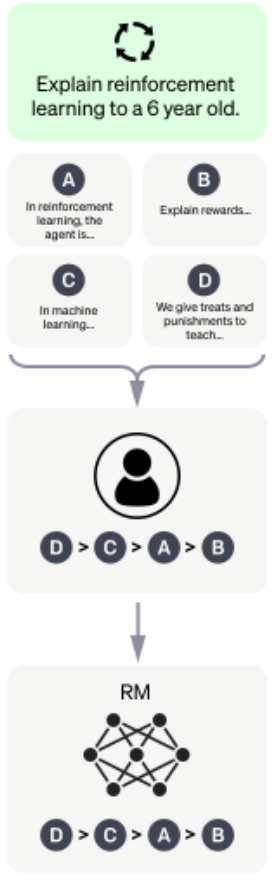
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

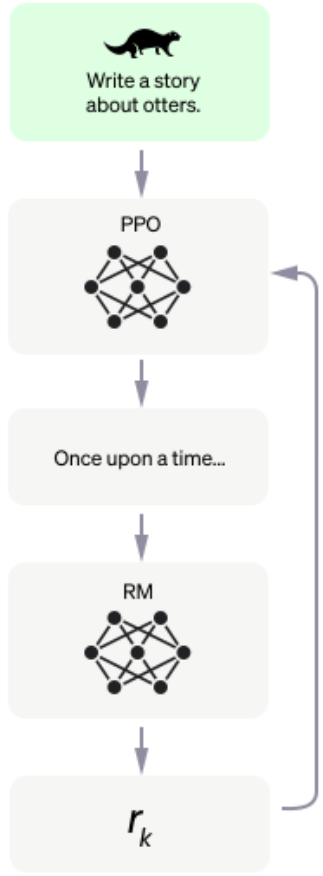
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



openAI