Sequence data

TO COMPLETE YOUR REGISTRATION, PLEASE TELL US WHETHER OR NOT THIS IMAGE CONTAINS A STOP SIGN:
STOP
NO
YES
ANSWER QUICKLY—OUR SELF-DRIVING CAR IS ALMOST AT THE INTERSECTION.
SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

Xkdc

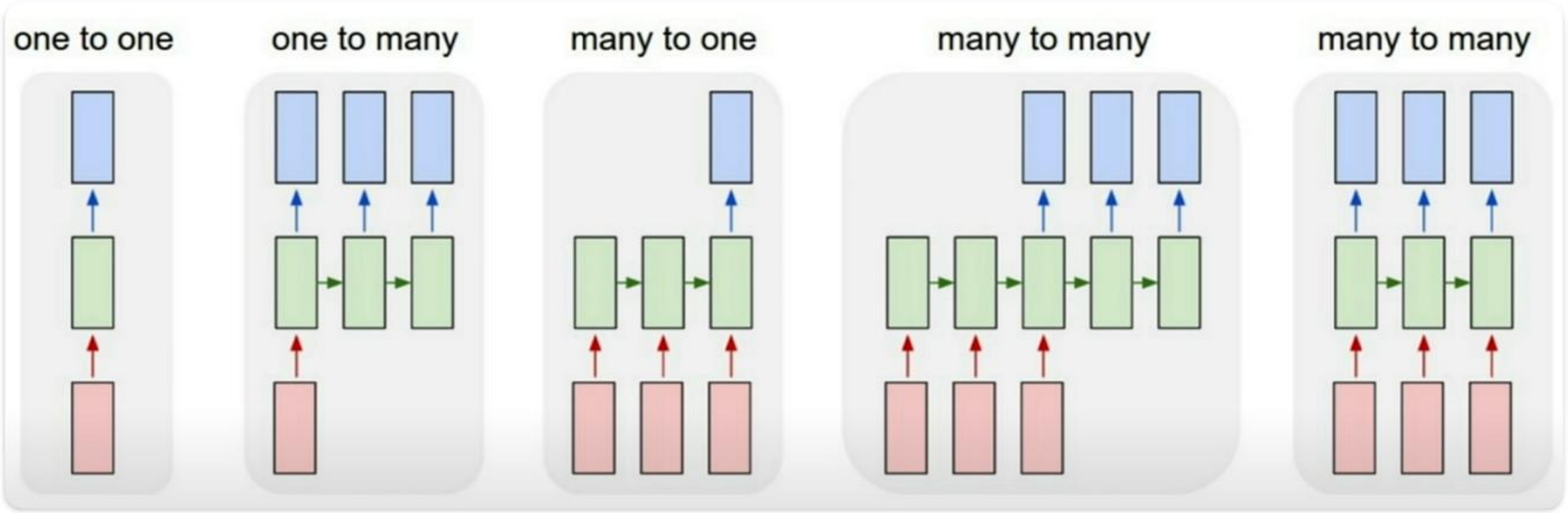
Sequence modeling

Sequence modeling is a deep learning approach that deals with **ordered data**, where the position of elements matters (e.g., text, speech, time series, DNA).

Sequence models learn patterns and dependencies within sequences to make predictions or generate new sequences.

Unlike traditional models, sequence models must **capture long-range dependencies** and **handle variable-length inputs** while maintaining meaningful order information.

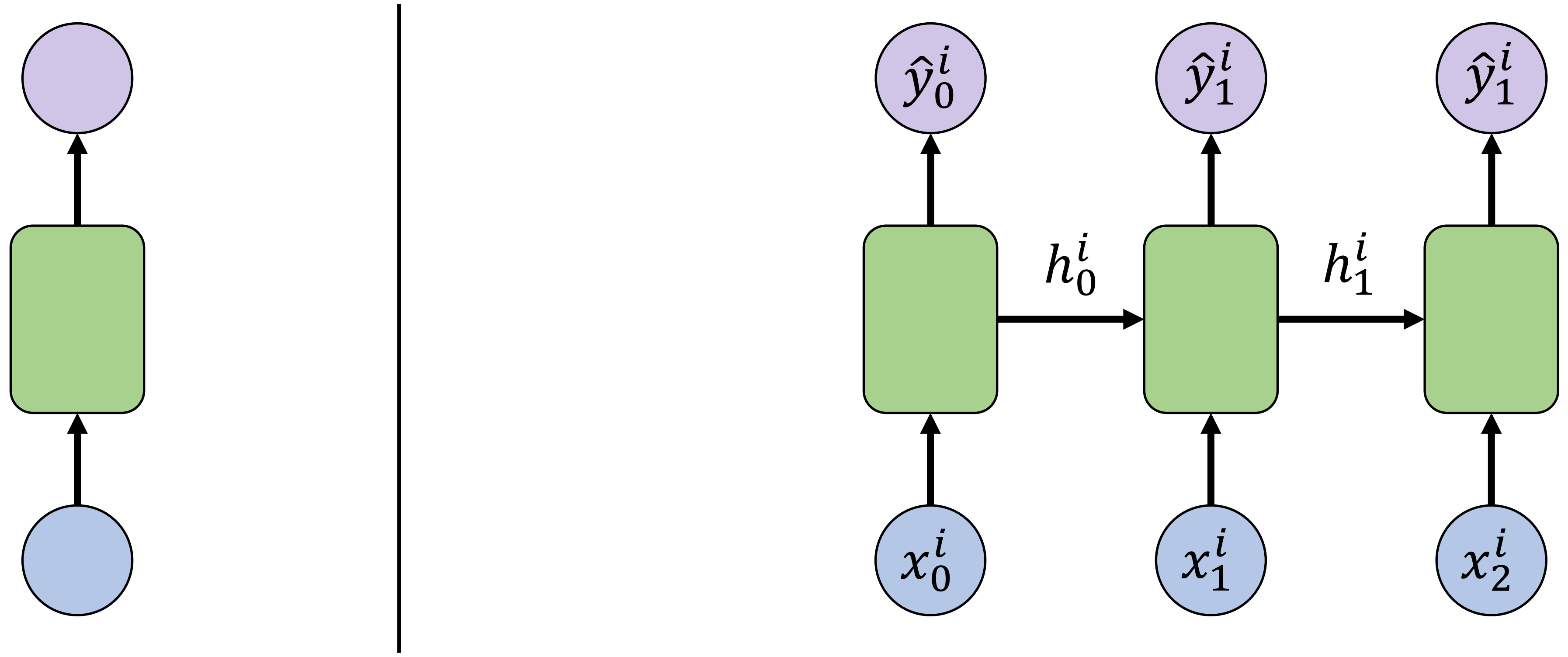
Sequence modeling



Vanilla Neural Networks e.g. **Image Caption** e.g. **Sentiment Classification** e.g. **Machine Translation** e.g. **Video classification on frame level**

Recurrent neural network (RNN)

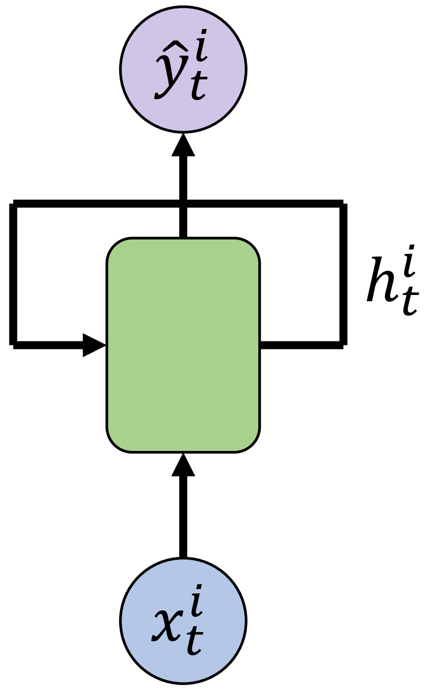
It is important to realize that the model parameters of the RNN are **the same in each time-step**, i.e. the green part in the diagram is always the same.

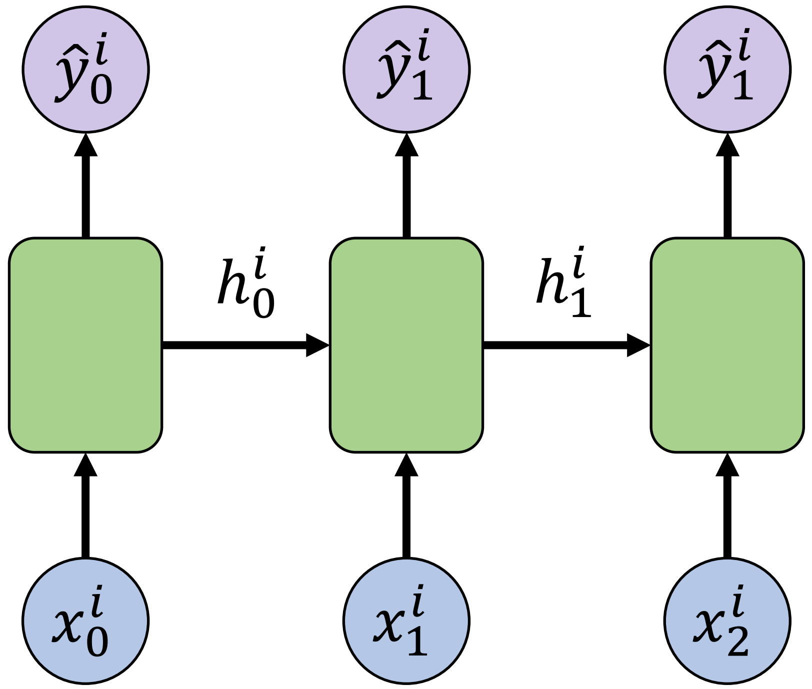


Recurrent neural network (RNN)

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.

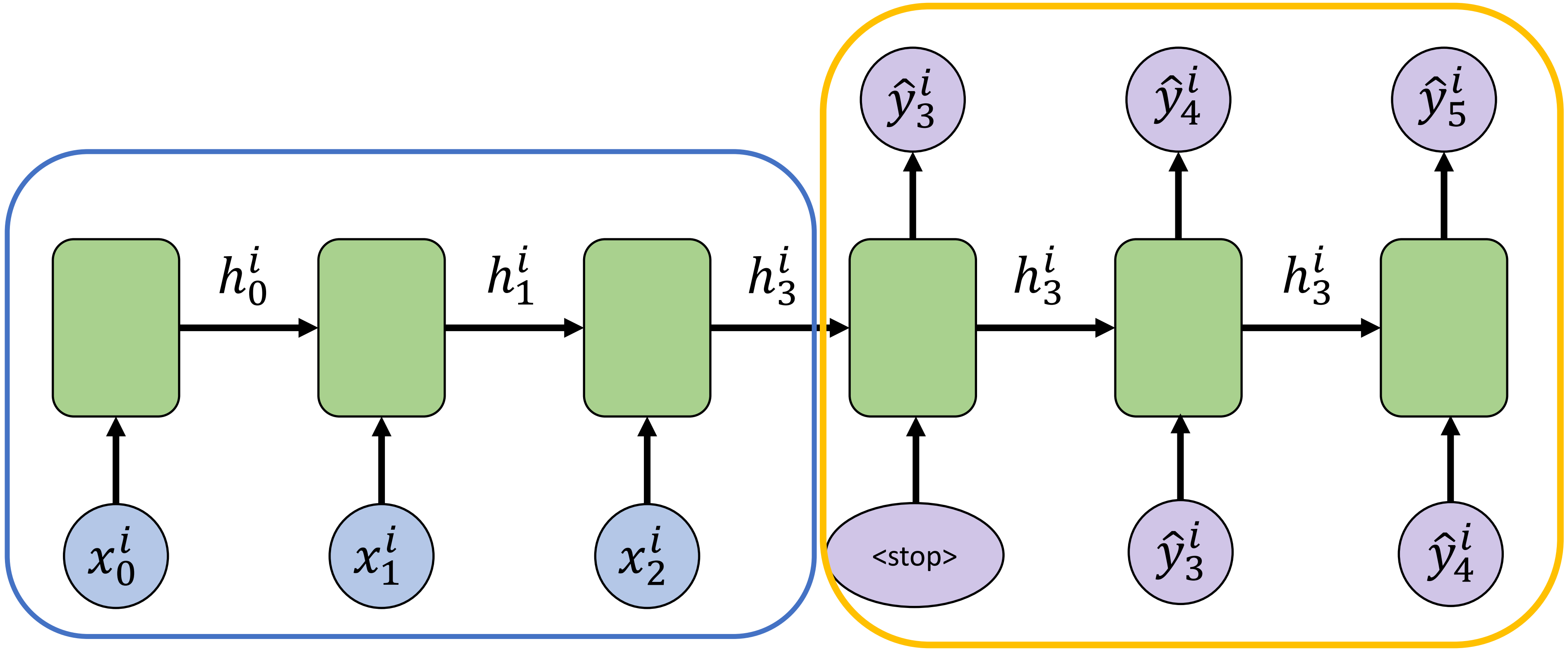
So, we can also represent the RNN as shown on the left.





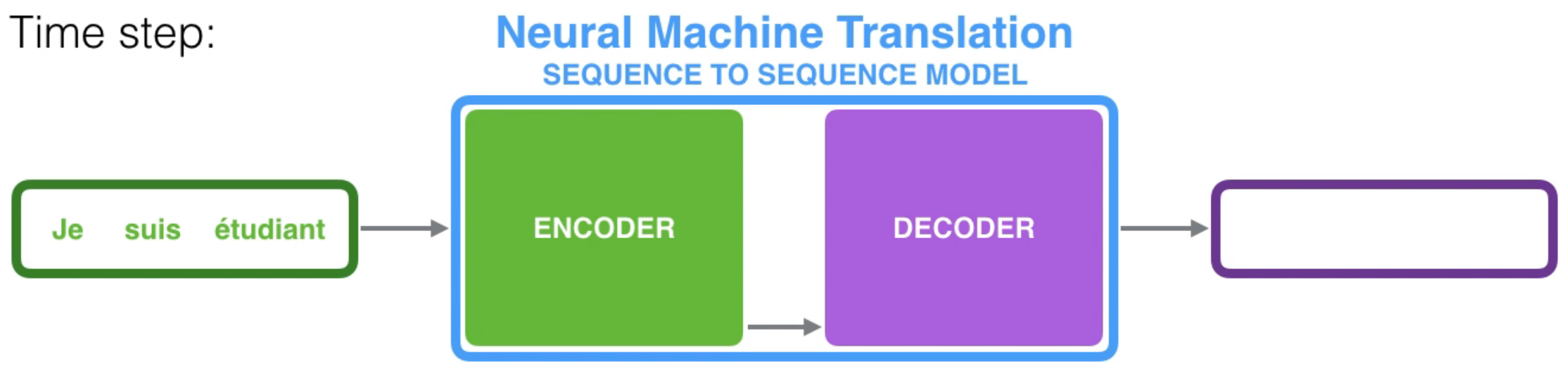
RNN encoding - decoding

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

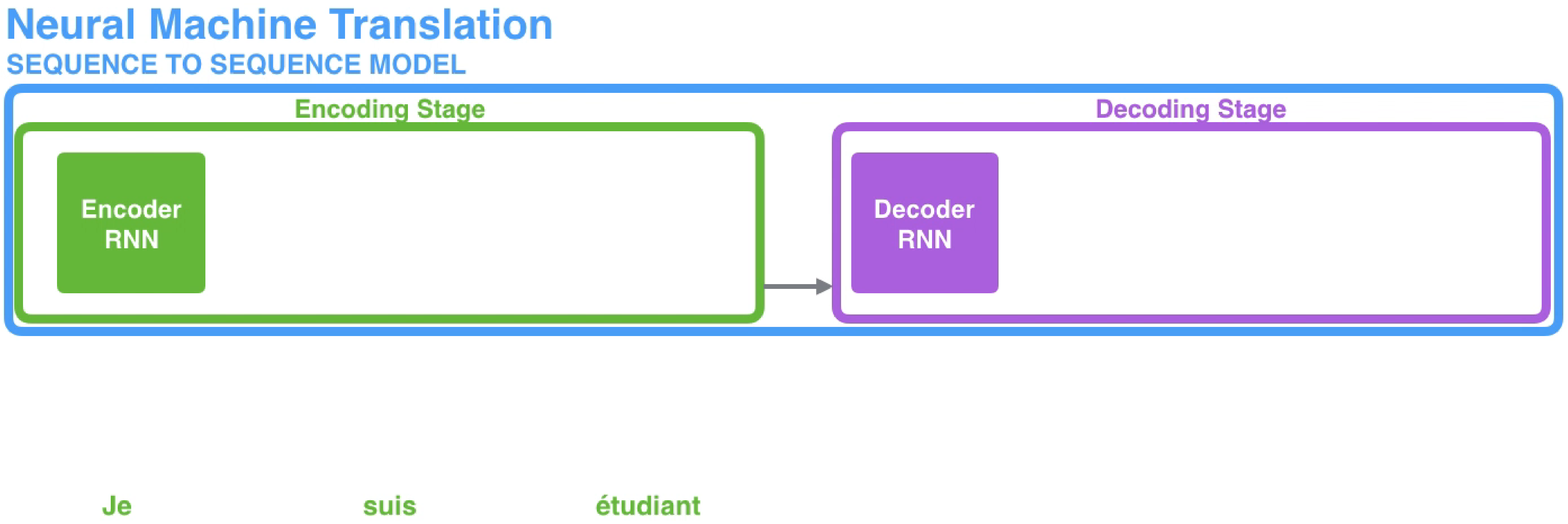
Folded this looks like this:



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

RNN encoding - decoding

Unfolded this looks like this.

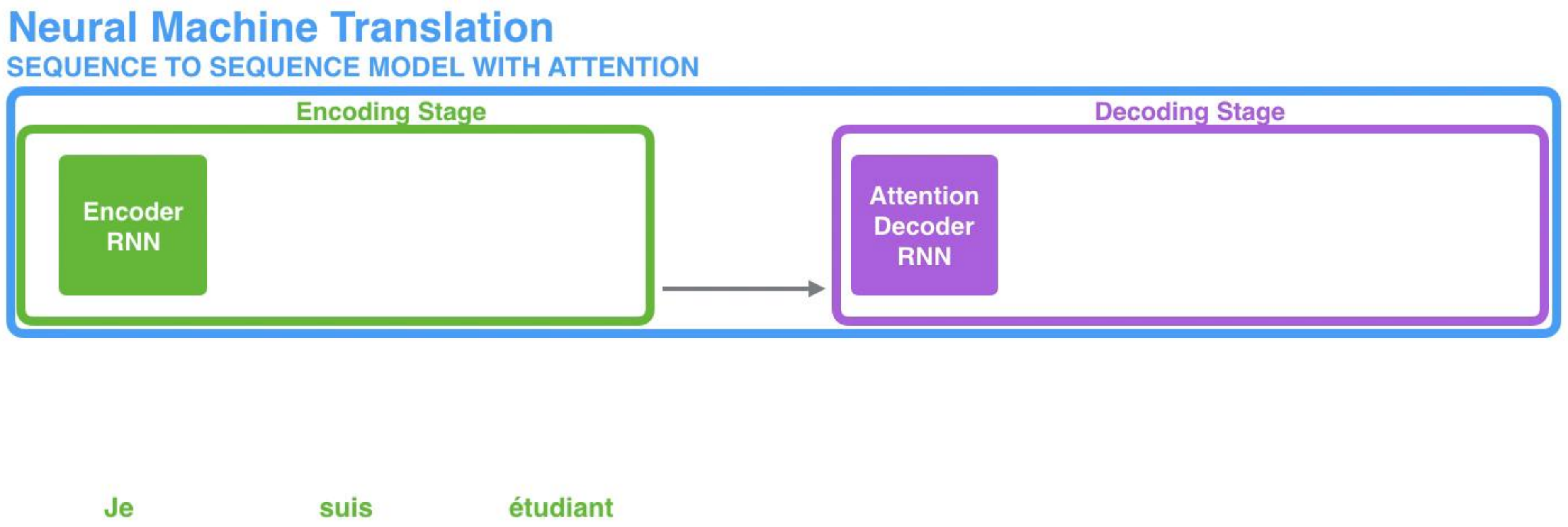


This creates a **bottleneck** in which all information in the input sequence needs to be encoded into the final hidden state vector.

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

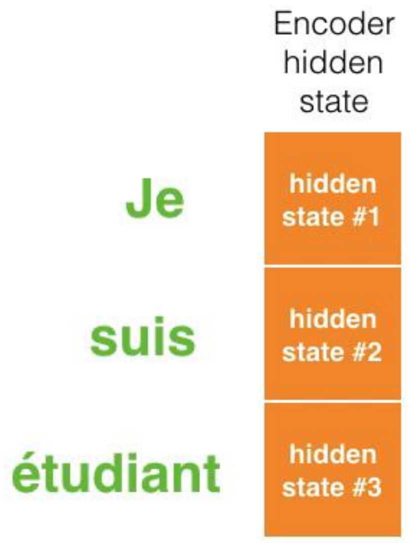
This issue can be solved by adding an **attention layer**. This is again a specialized architecture with model parameters that learns to pay more or less attention to the specific hidden state vectors during each time-step in the decoding.

Each input token is assigned a **weight (attention score)** indicating its importance for generating the current output token.

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.

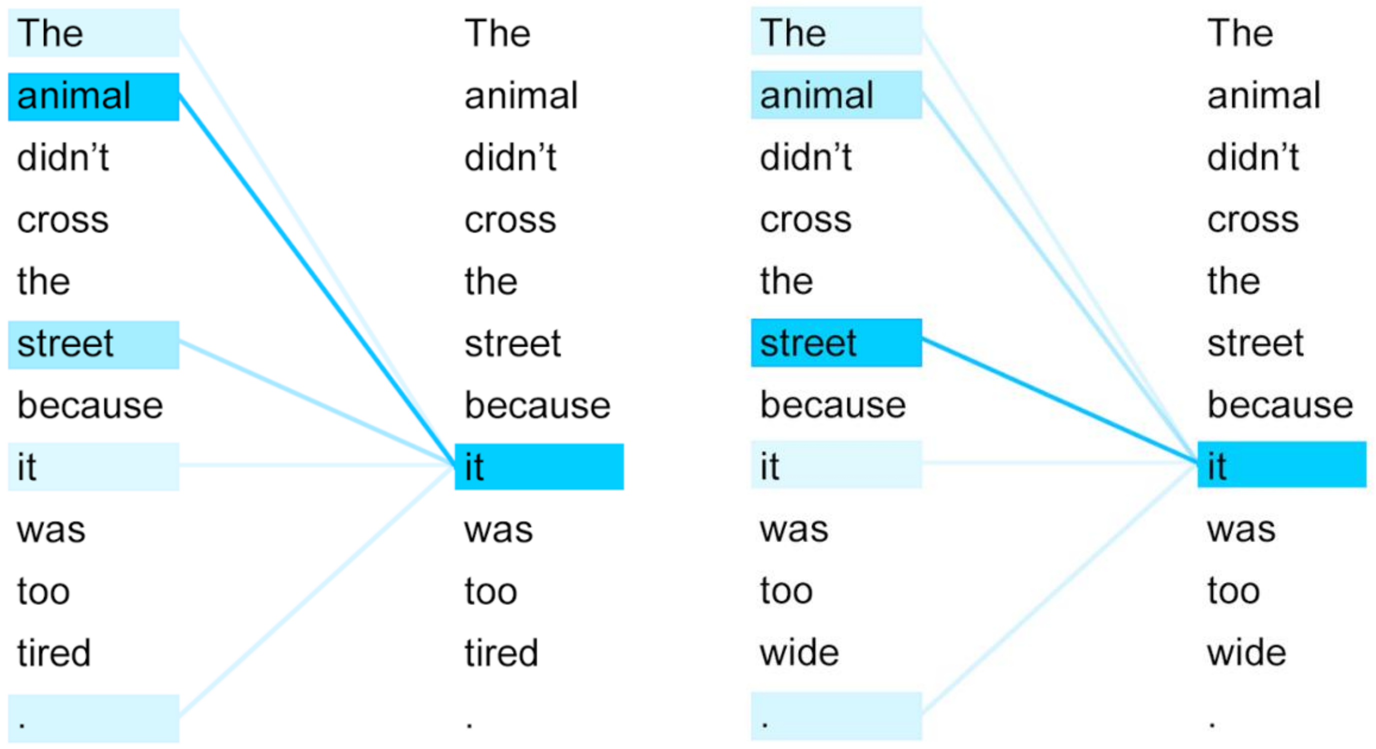


https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

(Self-)attention

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.

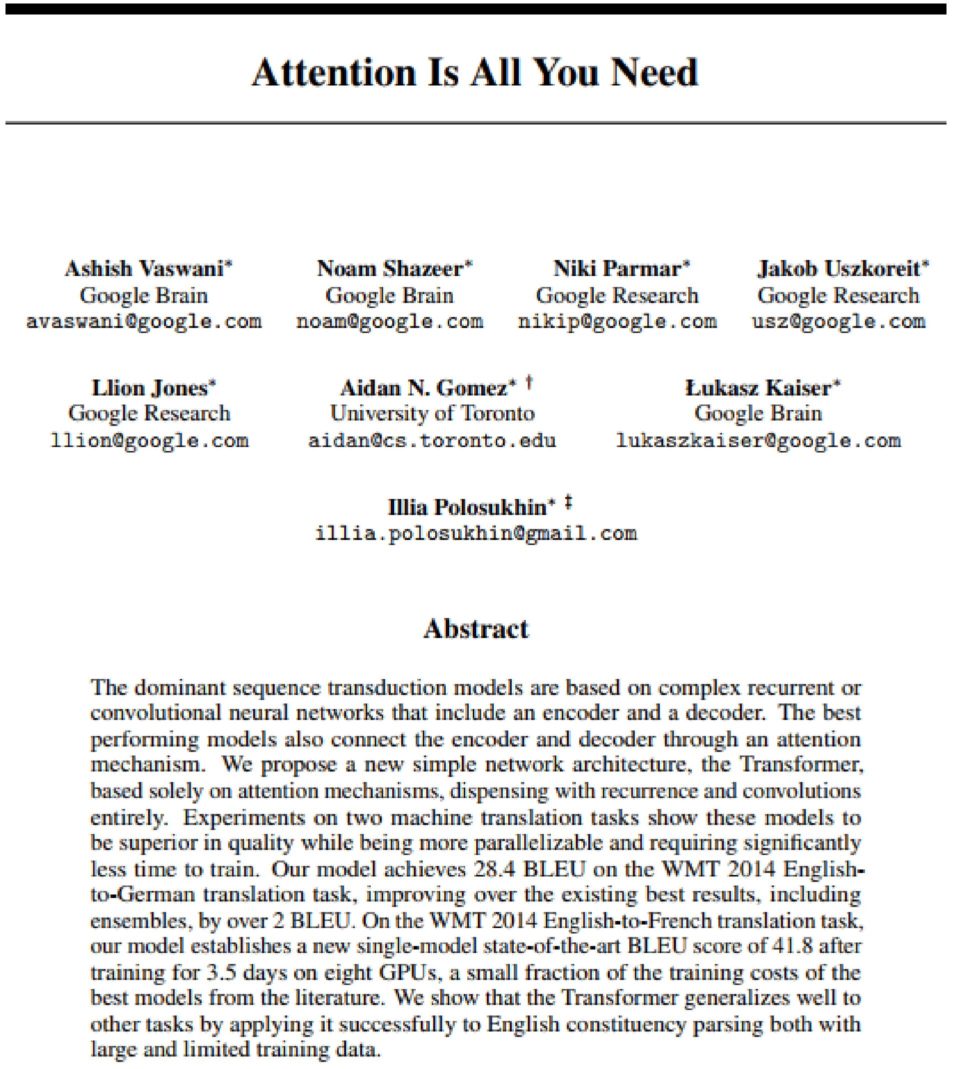


Transformer architecture

Self-attention mechanism: The Transformer model uses multi-head self-attention to capture long-range dependencies in text, allowing it to weigh the importance of different words regardless of their position.

Layered encoder-decoder structure: Originally designed for sequence-to-sequence tasks, the Transformer consists of stacked encoder and decoder layers, each containing self-attention and feedforward networks with normalization and residual connections.

Parallelization & efficiency: Unlike RNNs, Transformers process **entire sequences simultaneously**, making them highly parallelizable and enabling faster training on large datasets.



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

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

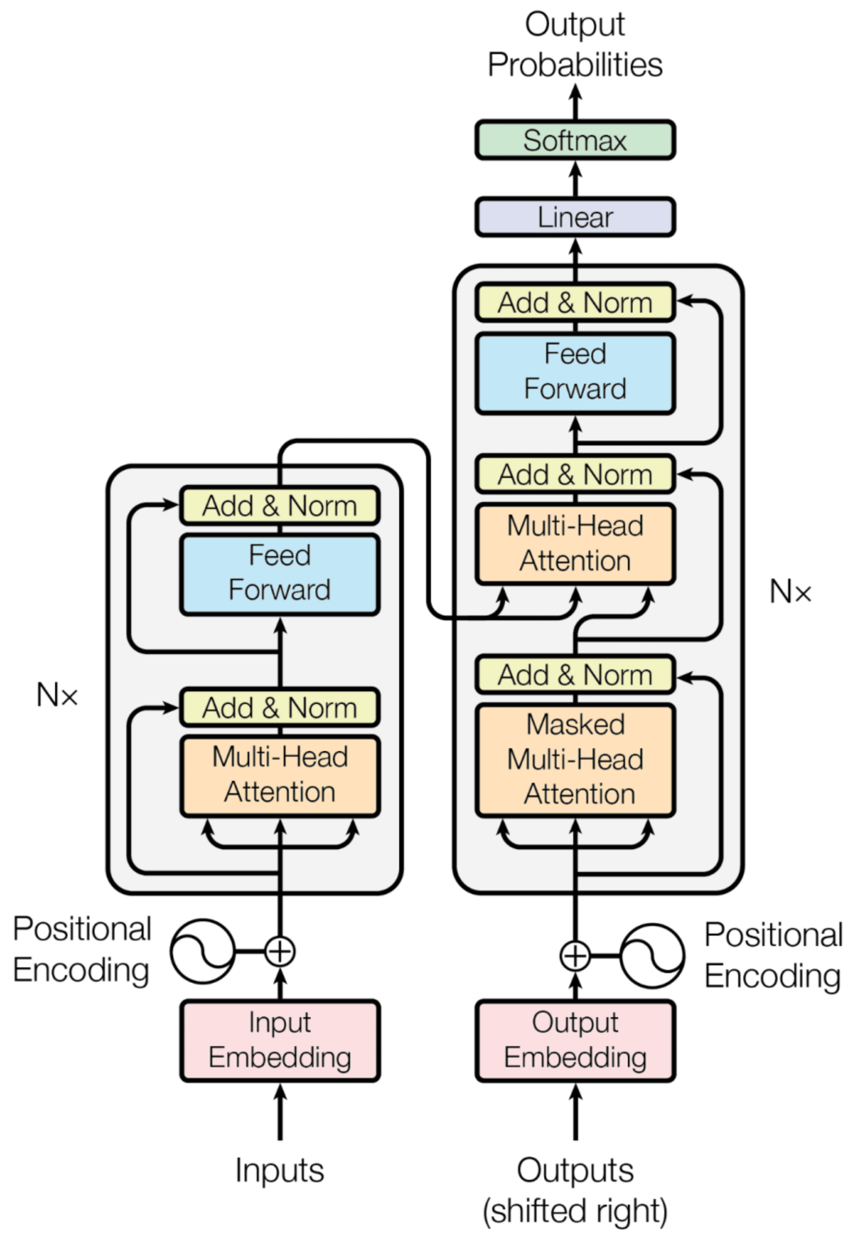
Ashish Vaswani et al., Attention is all you need. (NIPS'17)

Transformer architecture

Self-attention mechanism: The Transformer model uses multi-head self-attention to capture long-range dependencies in text, allowing it to weigh the importance of different words regardless of their position.

Layered encoder-decoder structure: originally designed for sequence-to-sequence tasks, the Transformer consists of stacked encoder and decoder layers, each containing self-attention and feedforward networks with normalization and residual connections.

Parallelization & efficiency: Unlike RNNs, Transformers process **entire sequences simultaneously**, making them highly parallelizable and enabling faster training on large datasets.



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

Sequence tokenization

Transformers require tokenized input for processing.

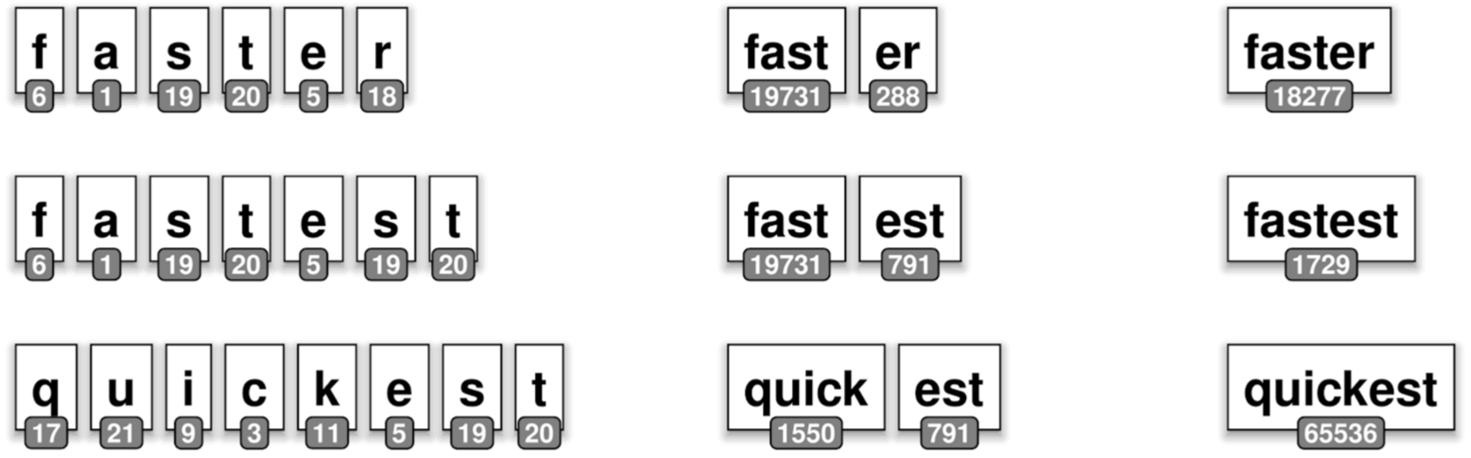
Padding [PAD] & truncation: Ensures fixed input length.

Special tokens: [CLS], [SEP] for sentence classification.

Character based

WordPiece based

Word based



f 6 a 1 s 19 t 20 e 5 r 18
f 6 a 1 s 19 t 20 e 5 s 19 t 20
q 17 u 21 i 9 c 3 k 11 e 5 s 19 t 20
fast 19731 er 288
fast 19731 est 791
quick 1550 est 791
faster 18277
fastest 1729
quickest 65536

Sequence tokenization

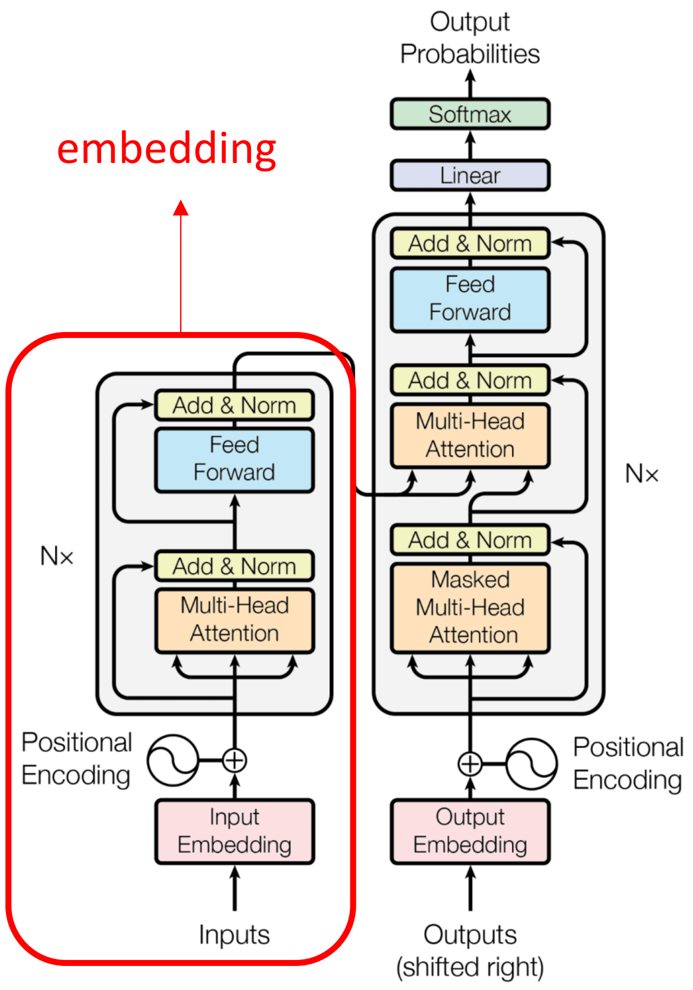
##prefix (denotes subwords, e.g., ##ing in running → [run, ##ing])

Reduces out-of-vocabulary issues:

- Rare or unseen words: "electromagnetism" → ["electro", "##magnet", "##ism"]
- Misspelled words: "happyness" → ["happy", "##ness"]
- Compound words: "smartwatch" → ["smart", "##watch"]
- Foreign or loanwords: "sushimaster" → ["sushi", "##master"]

BERT (Bidirectional Encoder Representations from Transformers)

Uses only the encoder part of the original Transformer architecture.

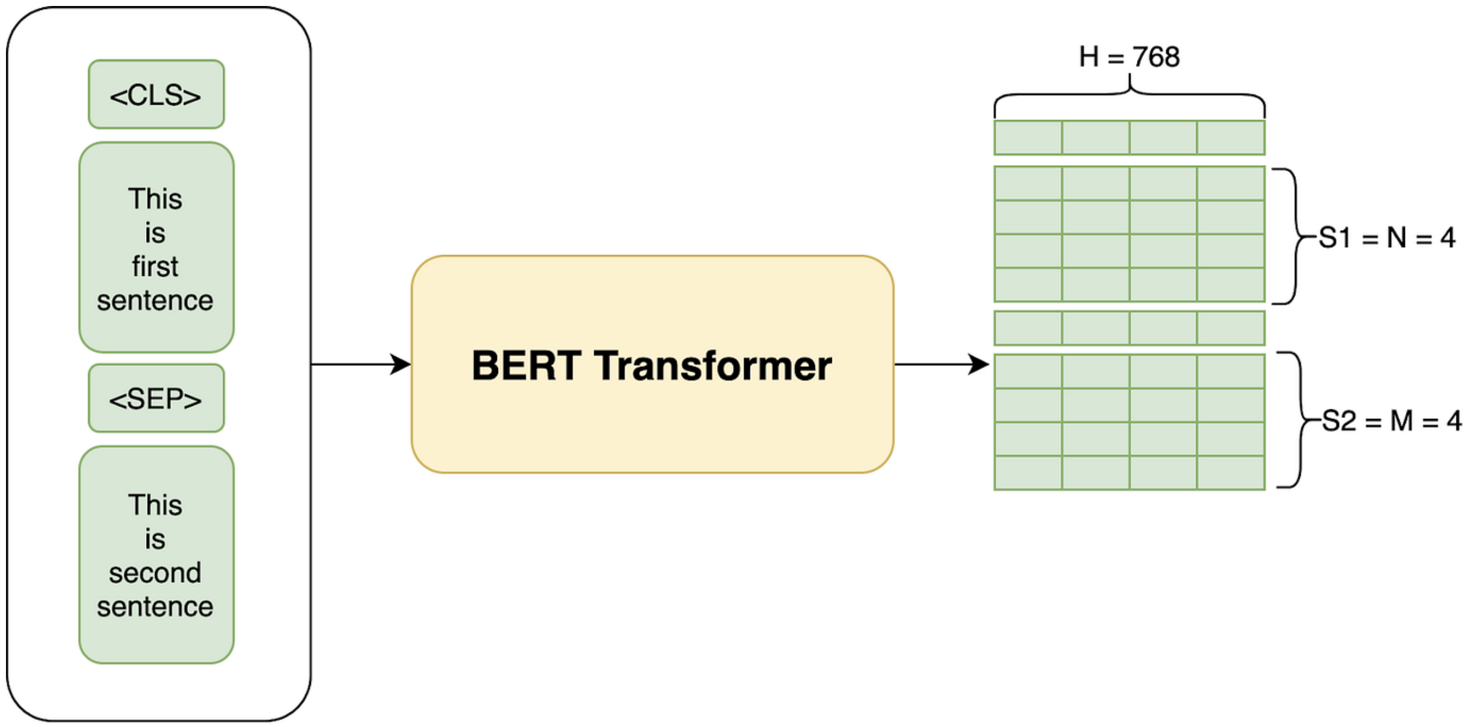


Ashish Vaswani et al., Attention is all you need. (NIPS'17)

BERT (Bidirectional Encoder Representations from Transformers)

Uses only the encoder part of the original Transformer architecture.

Outputs a 768-dimensional embedding vector for each word in the input sequence.



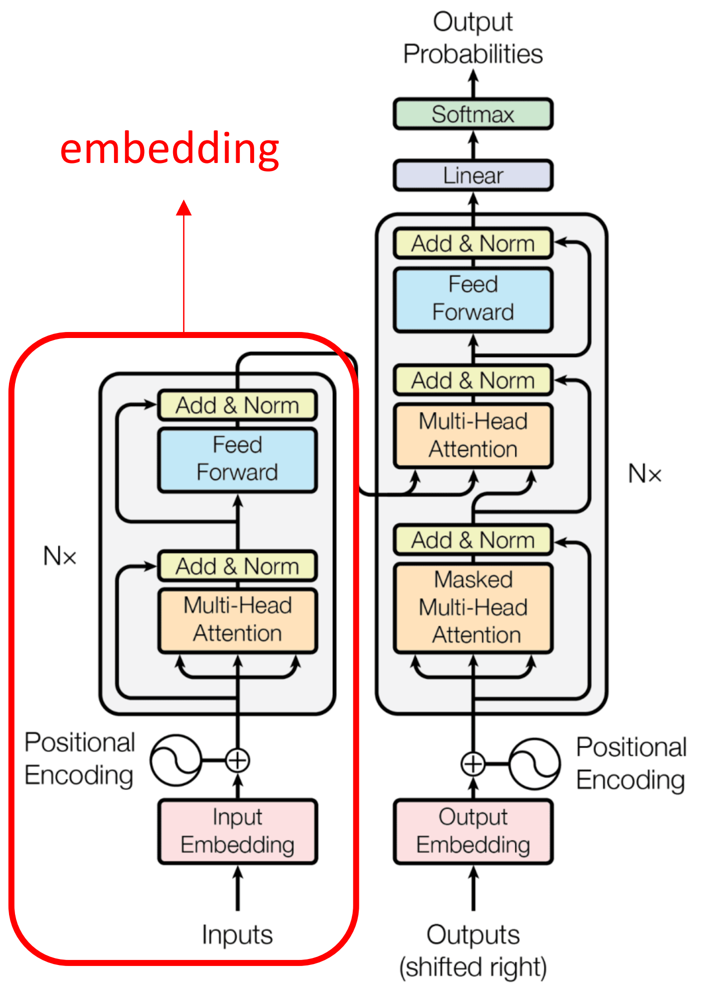
BERT (Bidirectional Encoder Representations from Transformers)

Uses only the encoder part of the original Transformer architecture

Outputs a 768-dimensional embedding vector for each word in the input sequence.

Reads text in both directions (uses all tokens in context window).

Pre-trained on a large corpus of text.



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

BERT (Bidirectional Encoder Representations from Transformers)

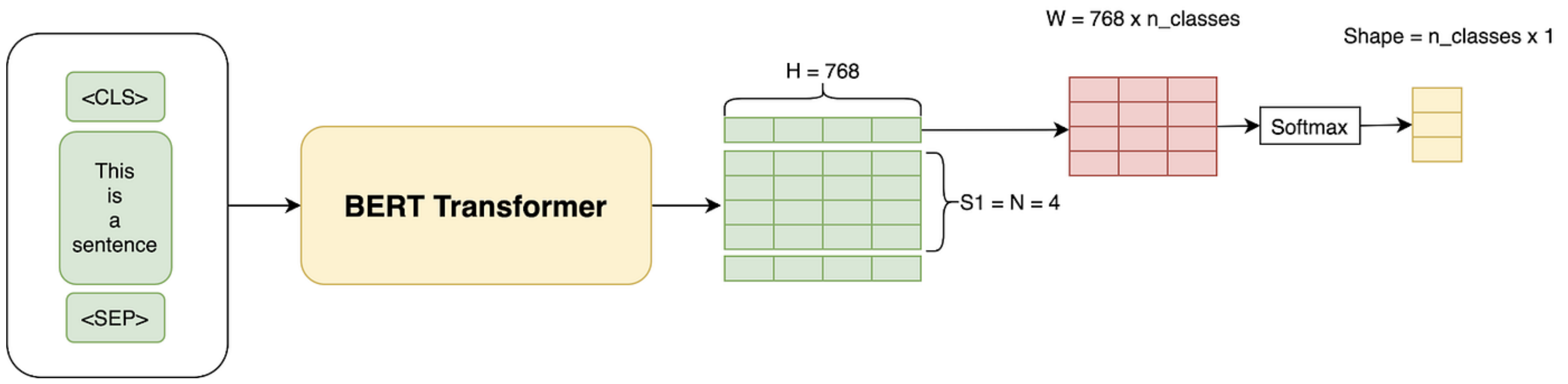
Uses only the encoder part of the original Transformer architecture

Outputs a 768-dimensional embedding vector for each word in the input sequence.

Reads text in both directions (uses all tokens in context window).

Pre-trained on a large corpus of text.

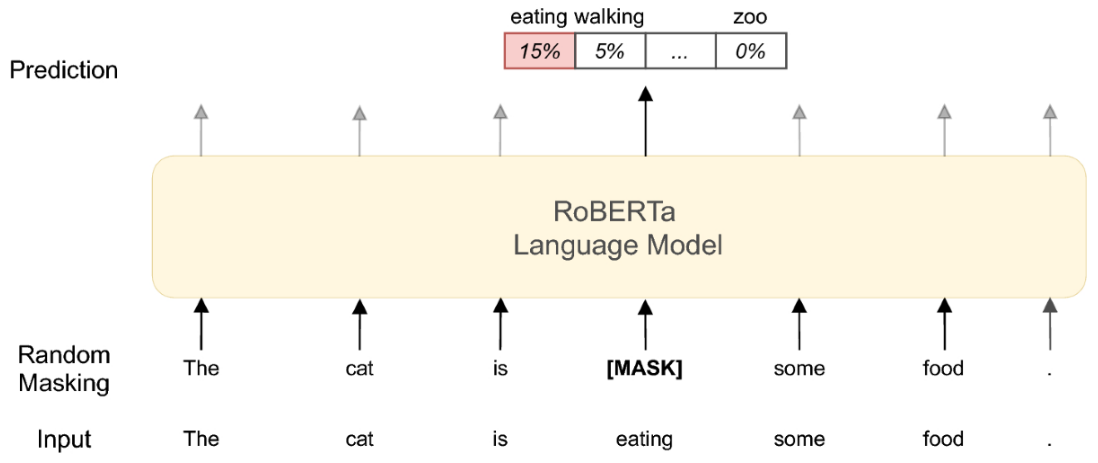
Fine-tuned for various NLP tasks such as:
- Sentiment analysis
- Named entity recognition
- Text classification
- …



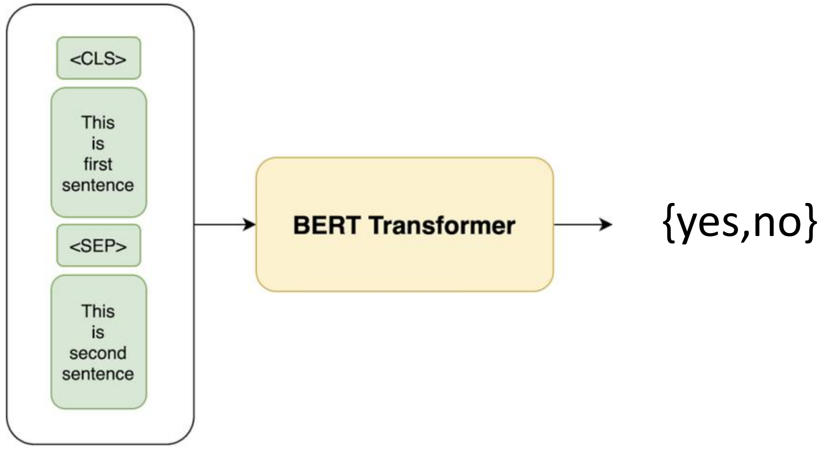
BERT pre-training

Pre-training. BERT is trained on unlabeled pair of sequences over two prediction tasks:

- Masked language modeling (MLM)



- Next sentence prediction (NSP)



BERT Flavours

BERT (Base & Large): The original BERT models introduced by Google in 2018. BERT-Base has 12 layers, 768 hidden units, and 12 attention heads, while BERT-Large has 24 layers, 1024 hidden units, and 16 attention heads.

DistilBERT: A lighter and faster version of BERT that retains 97% of its performance while being 60% smaller and running twice as fast. It uses knowledge distillation to learn from BERT while reducing size and computational cost.

BioBERT (Biomedical BERT): Pre-trained on PubMed abstracts and PMC full-text articles, making it highly effective for biomedical NLP tasks like named entity recognition (NER), question answering, and relation extraction.

ClinicalBERT: A variant of BERT trained on MIMIC-III clinical notes, making it more suitable for electronic health record (EHR) processing, medical note summarization, and patient outcome prediction.

PubMedBERT: Unlike BioBERT, which is initialized from standard BERT and then trained on biomedical data, PubMedBERT is trained from scratch using only PubMed abstracts and full-text papers, achieving better domain-specific performance.

…