Transformers & Recursive Networks

DSE 3151 DEEP LEARNING

B.Tech Data Science & Engineering
August 2023
Rohini R Rao & Abhilash K Pai
Department of Data Science and Computer Applications
MIT Manipal
Slide -4 of 5

Attention Is All You Need

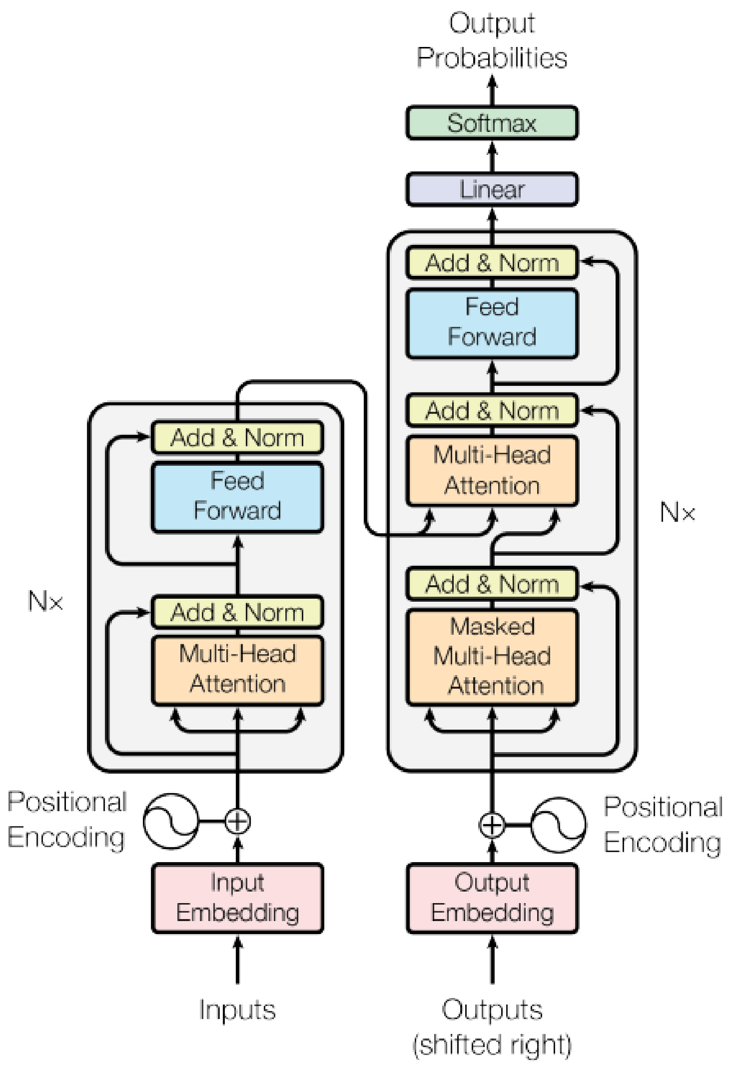
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* 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 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformers

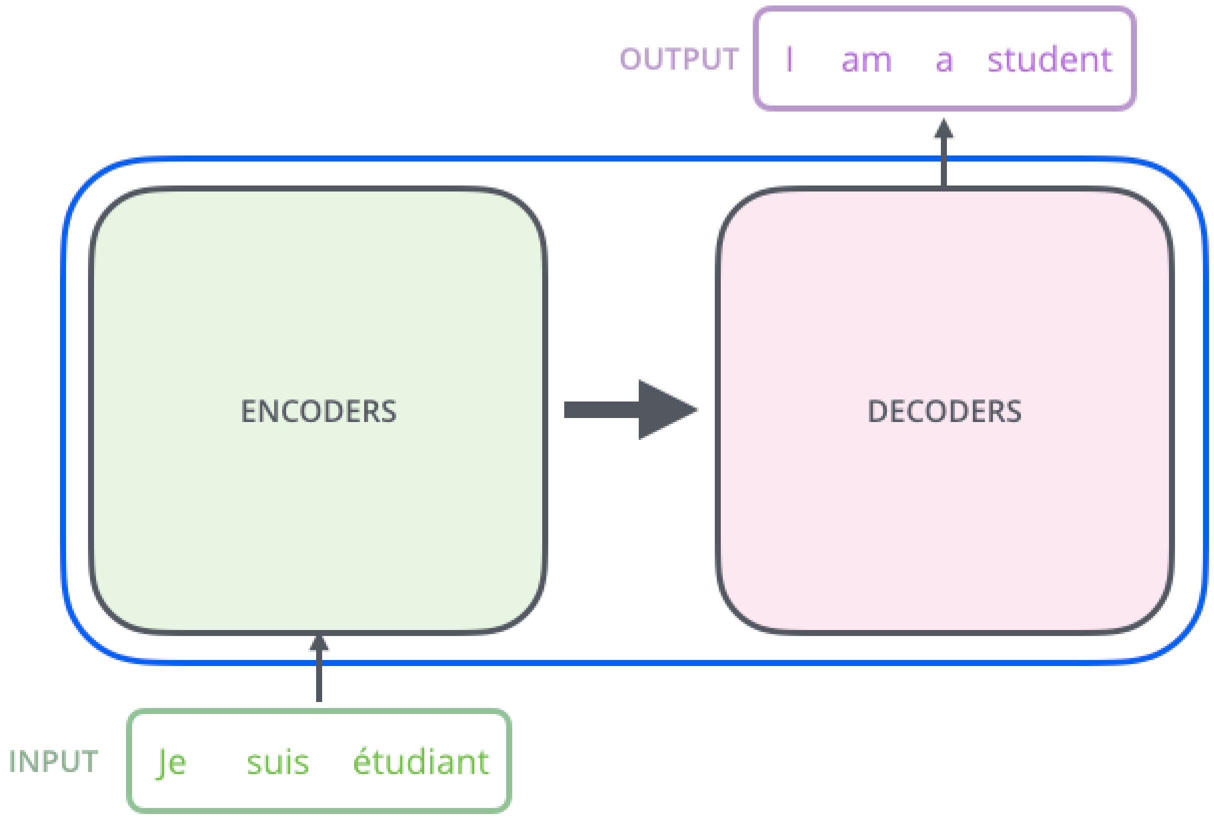


- Vaswani et al. 2017 proposed the transformer model, entirely built on **self attention mechanism without using sequence aligned recurrent architectures.**

- Key components:
  - Self attention
  - Multi-head attention
  - Positional encoding
  - Encoder-Decoder architecture

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

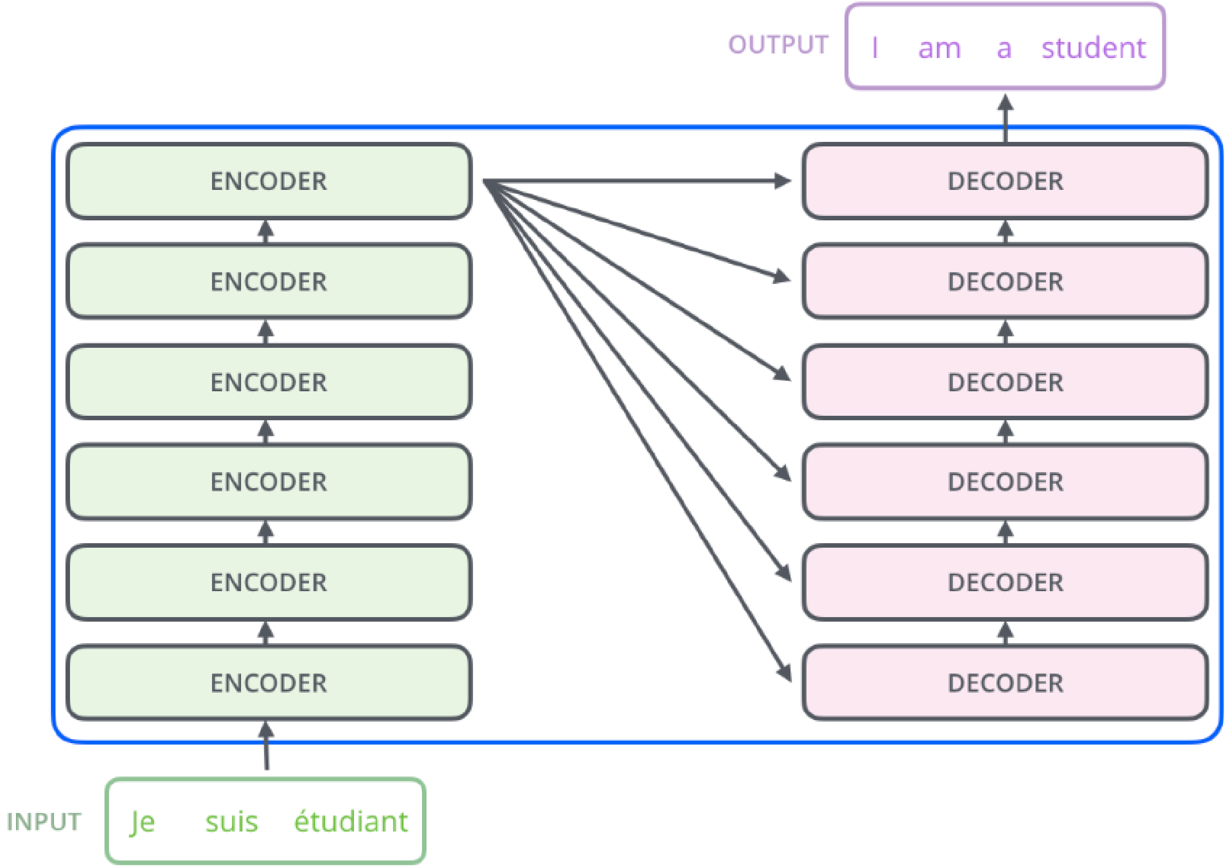
Transformers



The entire network can be viewed as an encoder decoder architecture

[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

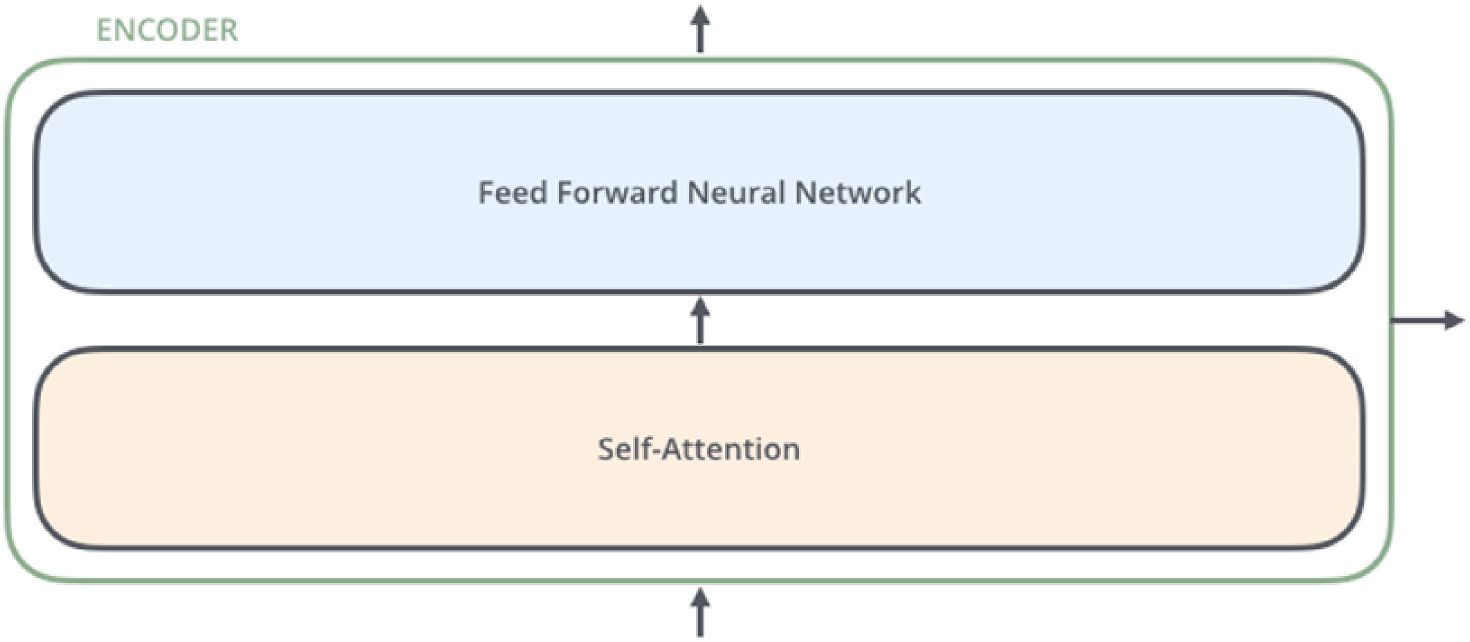
Transformers



The encoding component is a stack of encoders and the decoding component is also a stack of encoders of the same number (in the paper, this number = 6)

[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)](https://jalammar.github.io)

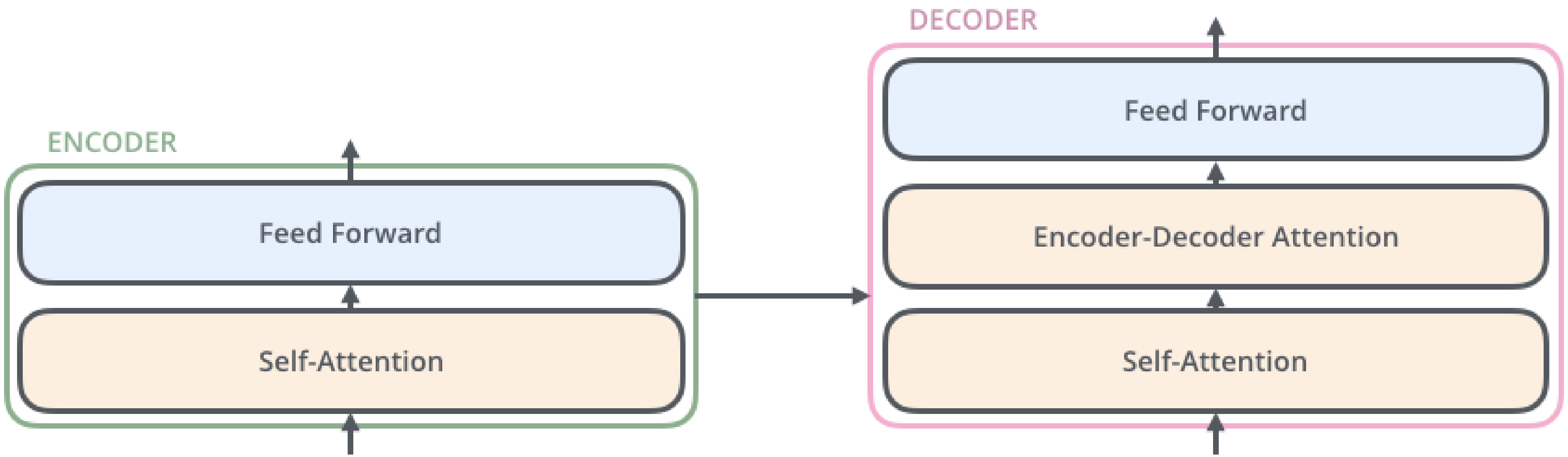
Transformers



- The encoders are all identical in structure (yet they do not share weights).
- Each one is broken down into two sub-layers
- The encoder's inputs first flow through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word – to the feed forward neural network.

[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)](jalammar.github.io)

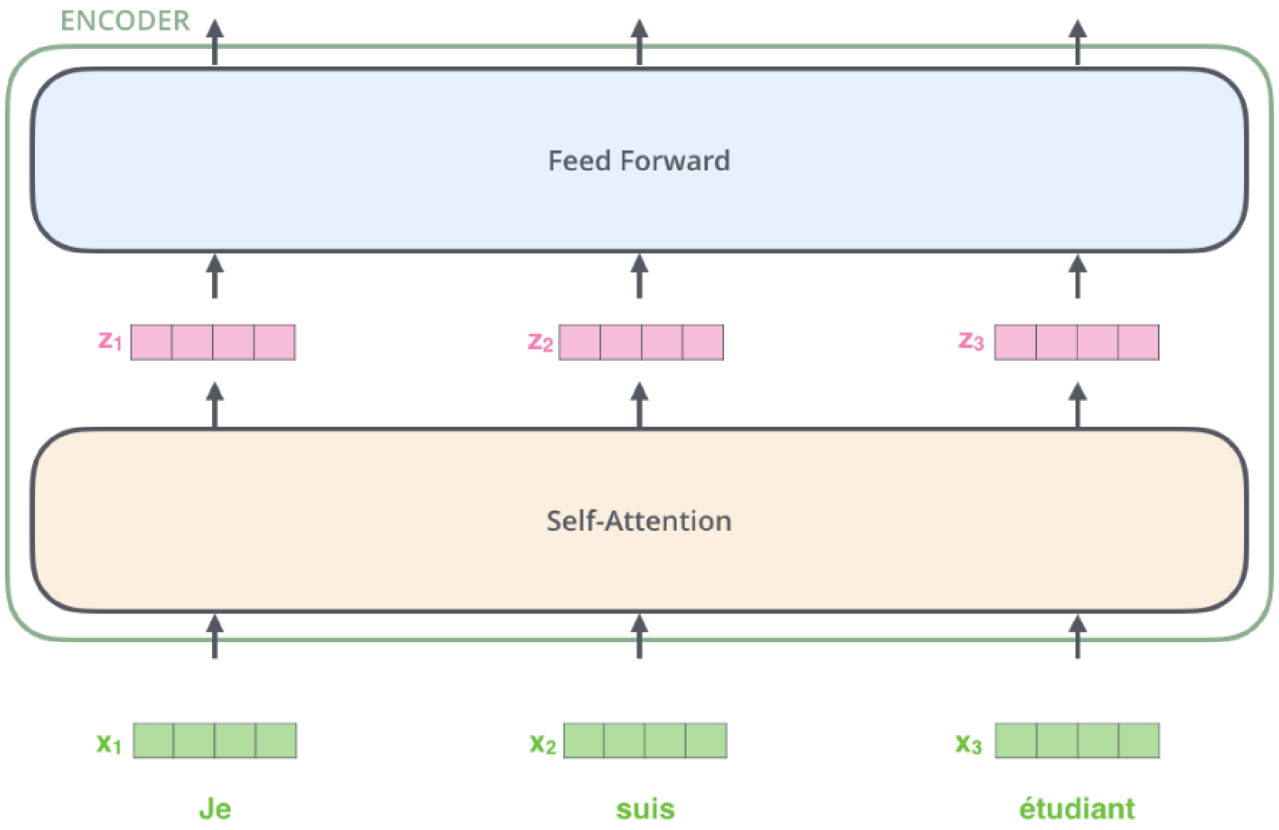
Transformers



The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence (similar what attention does in seq2seq models).

[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

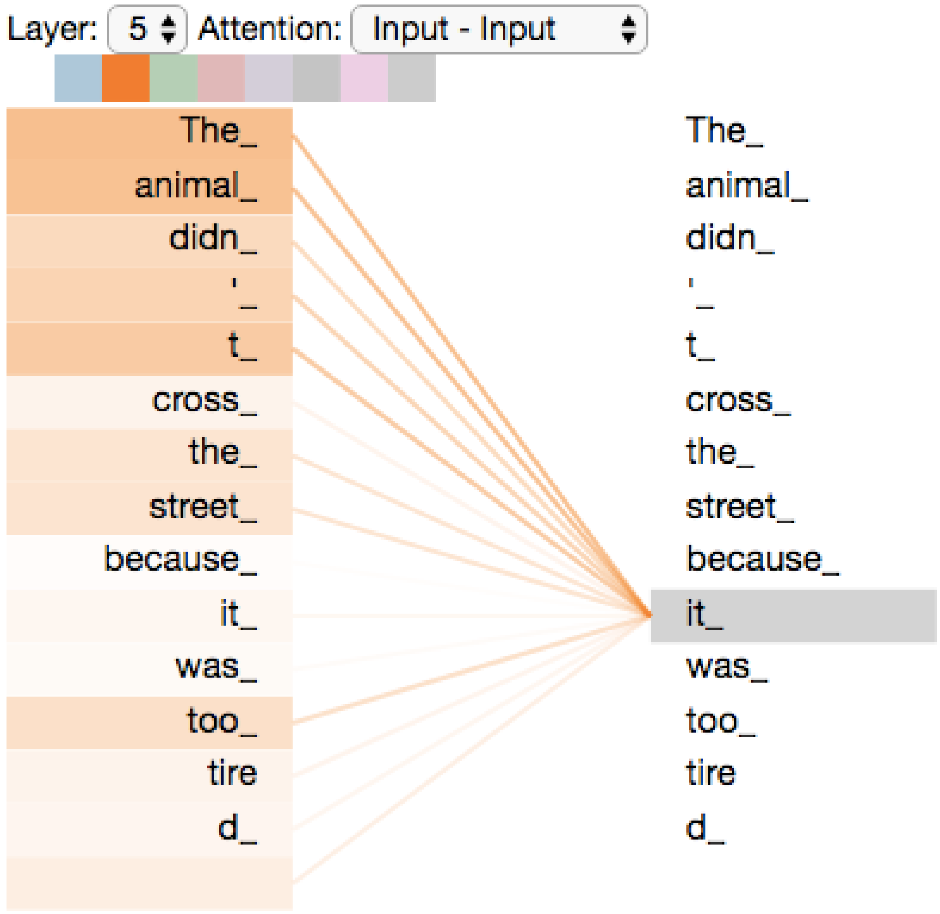
Transformers



- A word in each position flows through its own path in the encoder.
- There are dependencies between these paths in the self-attention layer.
- The feed-forward layer does not have those dependencies.
- Thus, the various paths can be executed in parallel while flowing through the feed-forward layer.
- This is a key property of the transformer.

The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Transformers : Self attention



S1: Animal didn't cross the street because it was too tired

S2: Animal didn't cross the street because it was too wide

- In sentence S1, "it" refers to "animal" and in sentence S2 "it" refers to "street".
- Such deductions are hard for traditional sequence to sequence models.
- While processing each word in a sequence, self-attention mechanism allows the model to decide as to which other parts of the same sequence it needs to focus on, which makes such deductions easier and allows better encoding.
- RNNs which maintain a hidden state to incorporate the representation of previous vectors are no longer needed!

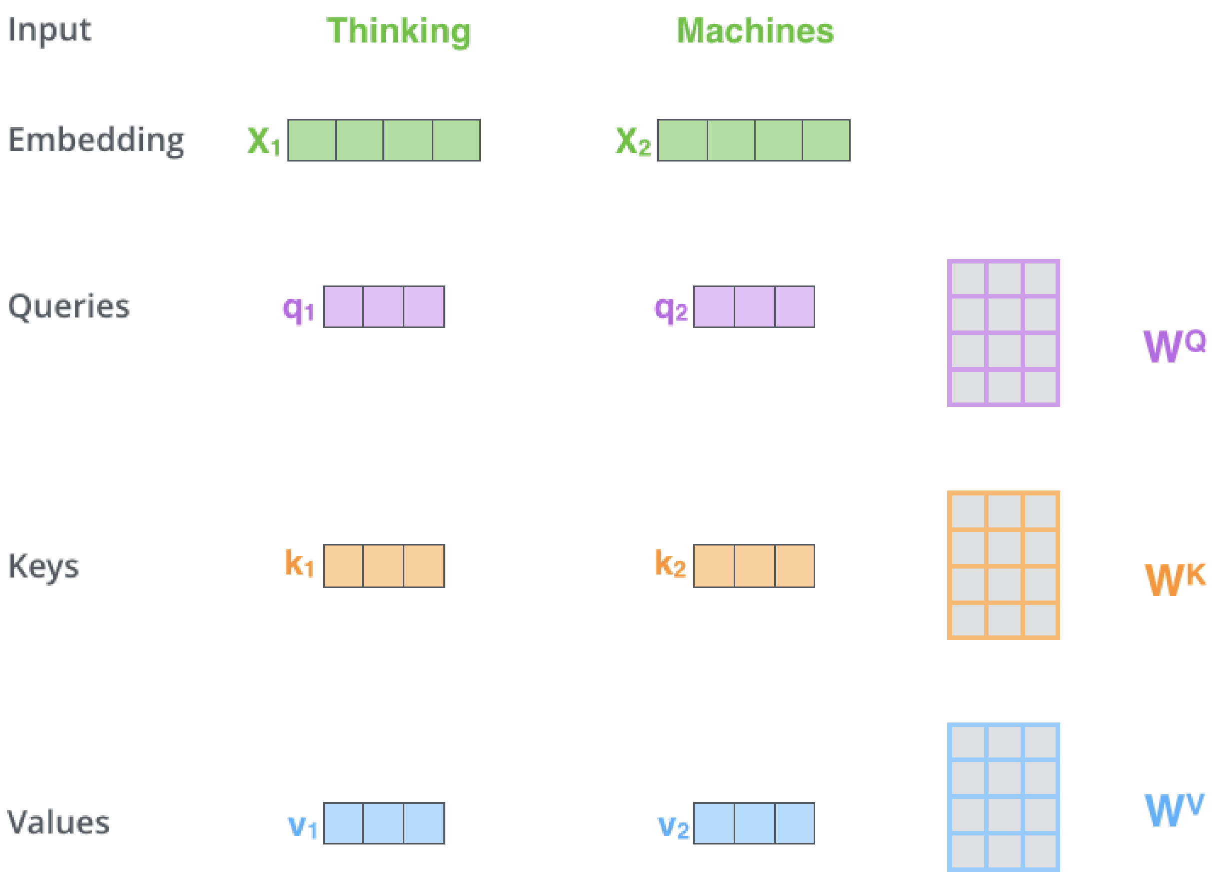
The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Transformers : Self attention

Step 1: Create three vectors from encoder's input vector (x_i):

- Query vector (q_i)
- Key vector (k_i)
- Value vector (v_i)

These are created by multiplying the input with weight matrices W^q, W^k, W^v, learned during training.



- Note: In the paper by Vaswani et al, the **q, k and w** vectors has dimension of **64** and the input vector **x** has a dimension of **512,**

The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

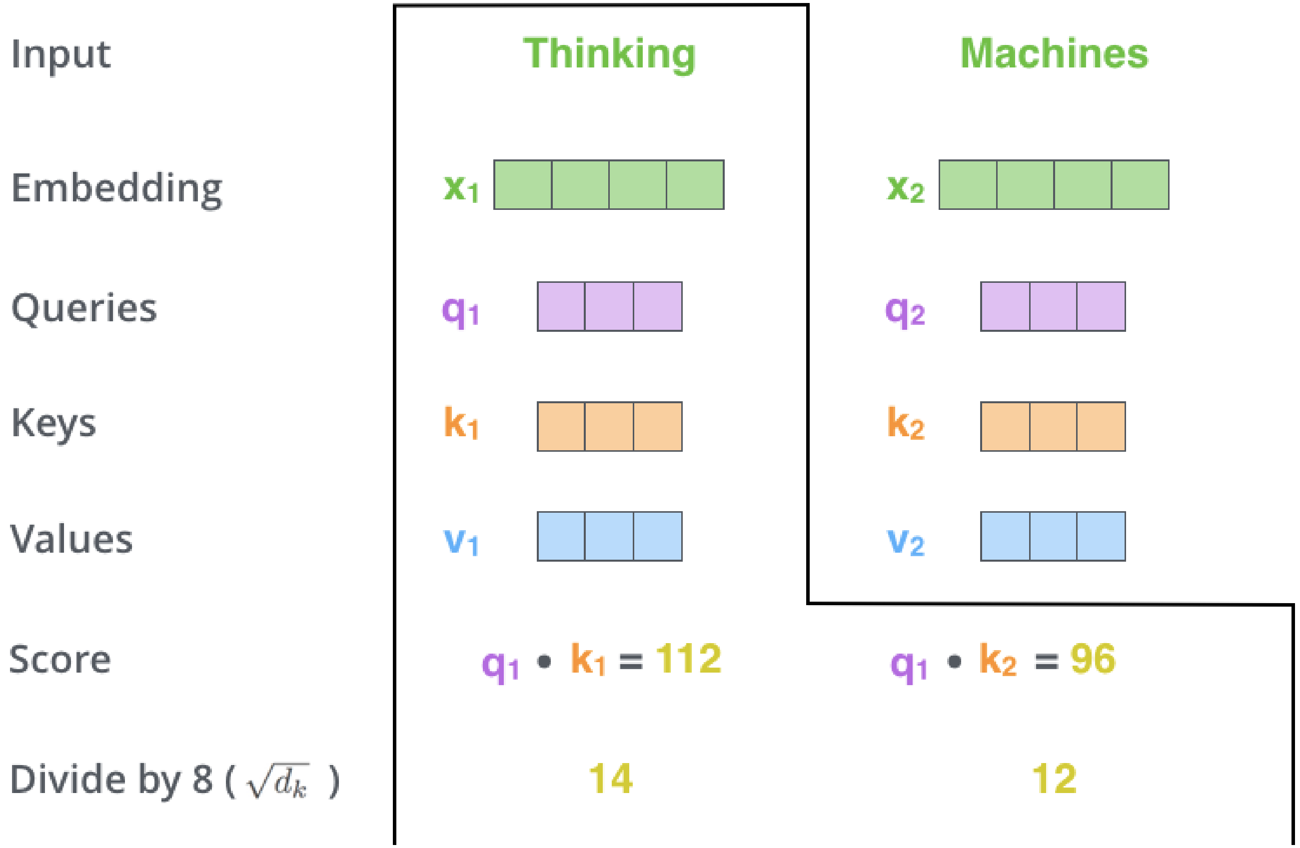
Transformers : Self attention

Step 2: Calculate self attention scores of each word against the other words in the same sentence.

- This is done by taking the dot product of query vector with the key vector of respective words.

- The scores are the divided by square root of the key length.

- This is called **Scaled Dot-Product attention** - it leads to more stable gradients.



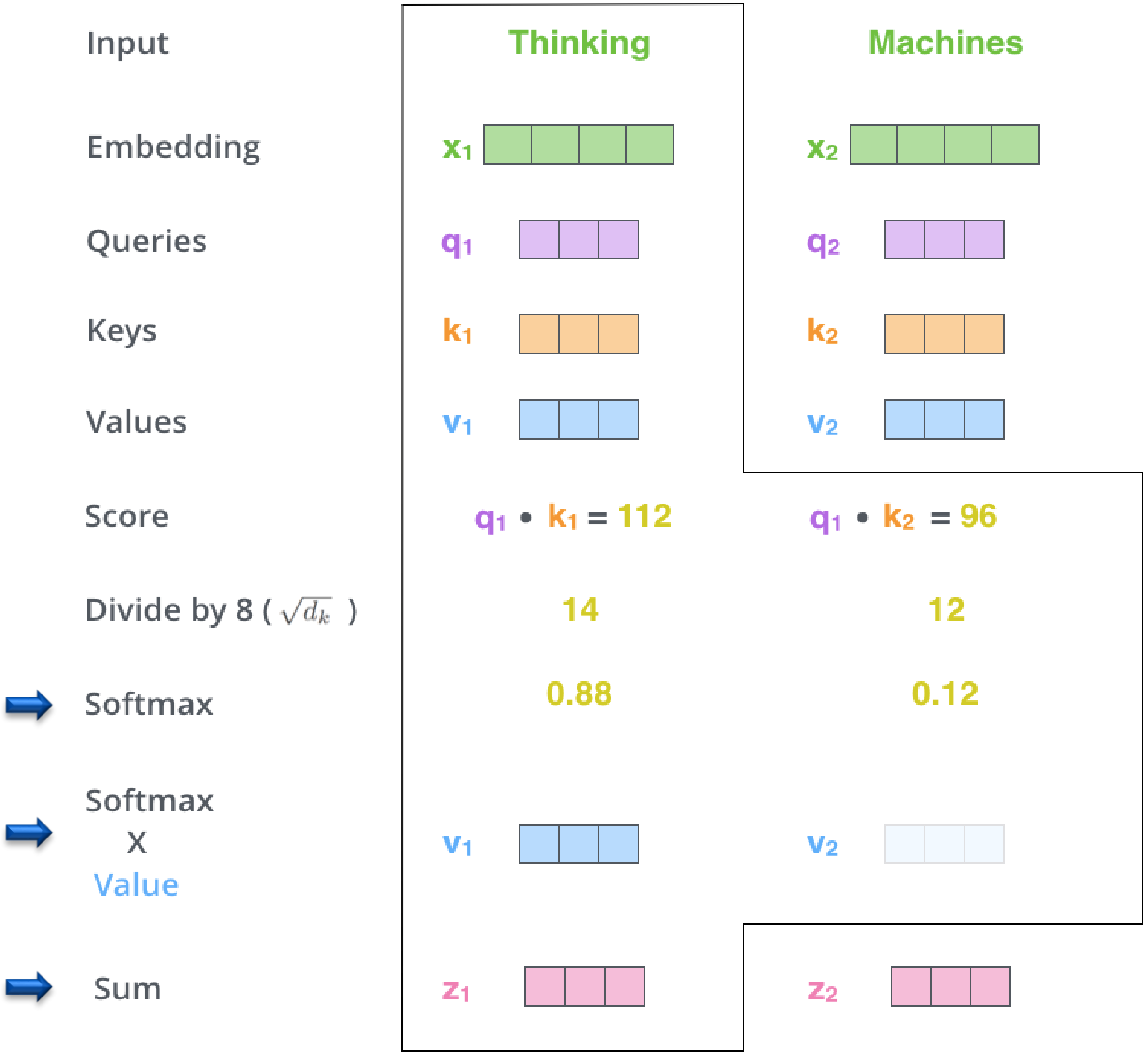
The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Transformers : Self attention

Step 3: Softmax is used to obtain normalized probability scores; determines how much each word will be expressed at this position.

Step 4: Multiply each value vector by the Softmax score; to keep values of words we want to focus on and drown out irrelevant words.

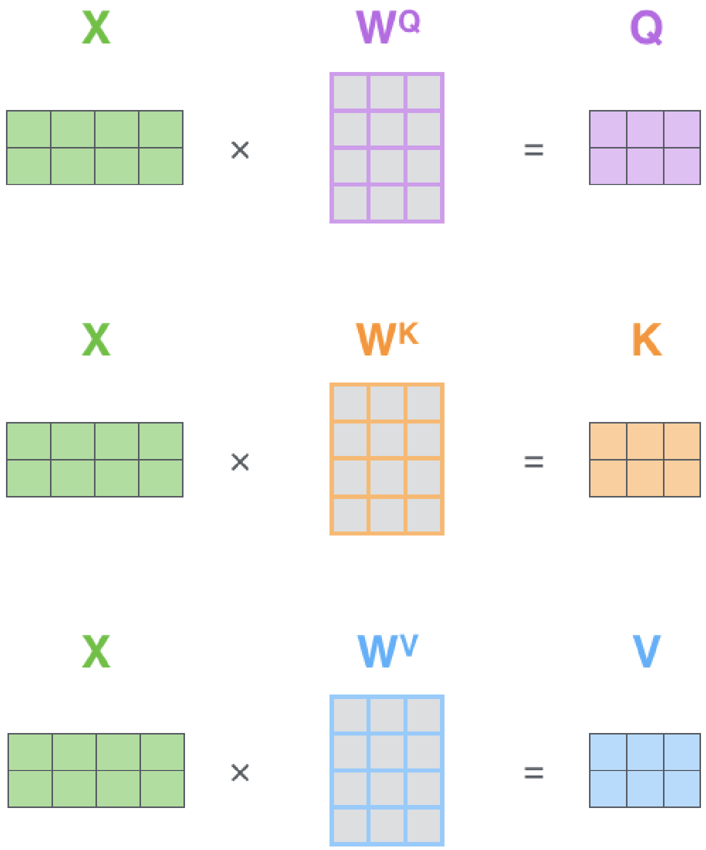
Step 5: Sum up the weighted value vectors ; produces the output of self-attention layer at this position (for first word)

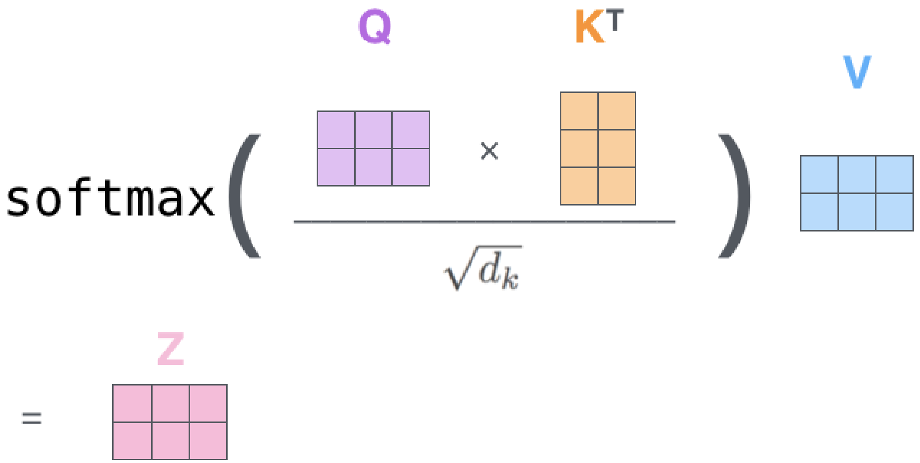


The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Transformers : Self attention

- In the actual implementation, however, Step 1 to Step 4 is done in matrix form for faster processing.

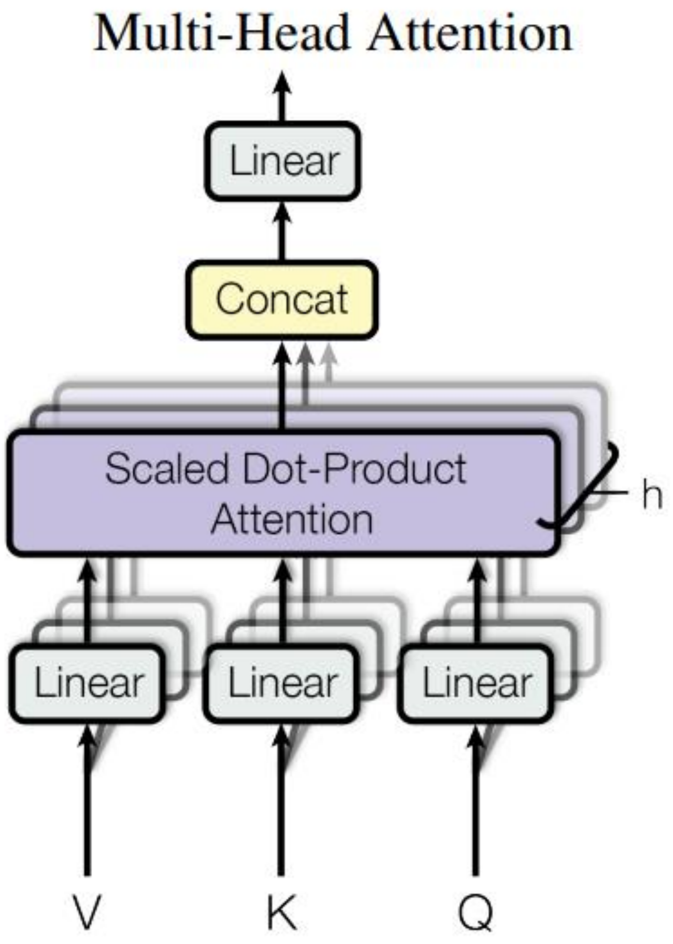




[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)](https://jalammar.github.io)

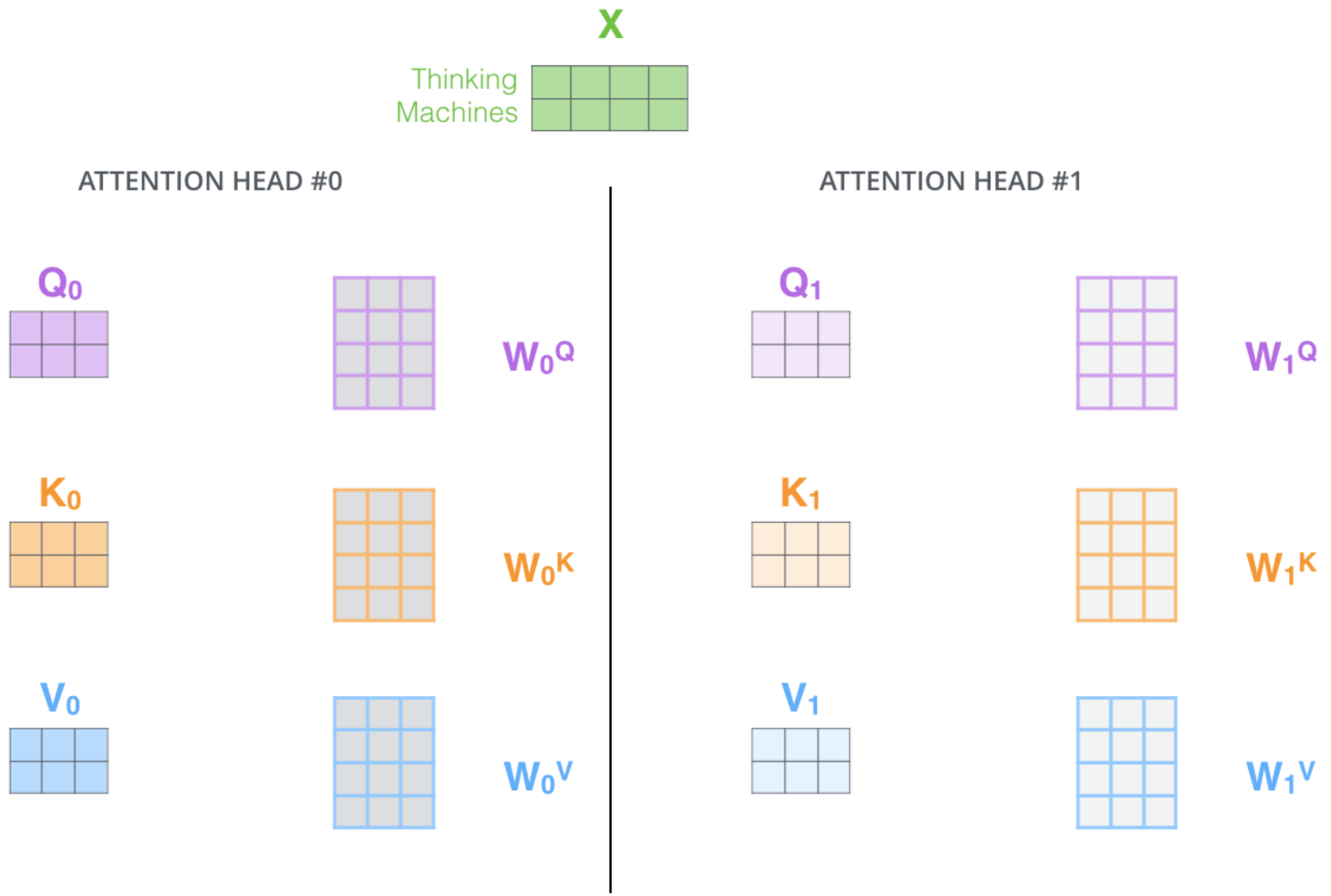
Transformers : Multi-head Attention

- Multi-head attention improves the performance of attention layer in two ways:
  - It expands the model’s ability to focus on different positions.
  - It gives the attention layer multiple “representation subspaces” i.e., we have not only one, but multiple sets of Query/Key/Value weight matrices.



[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

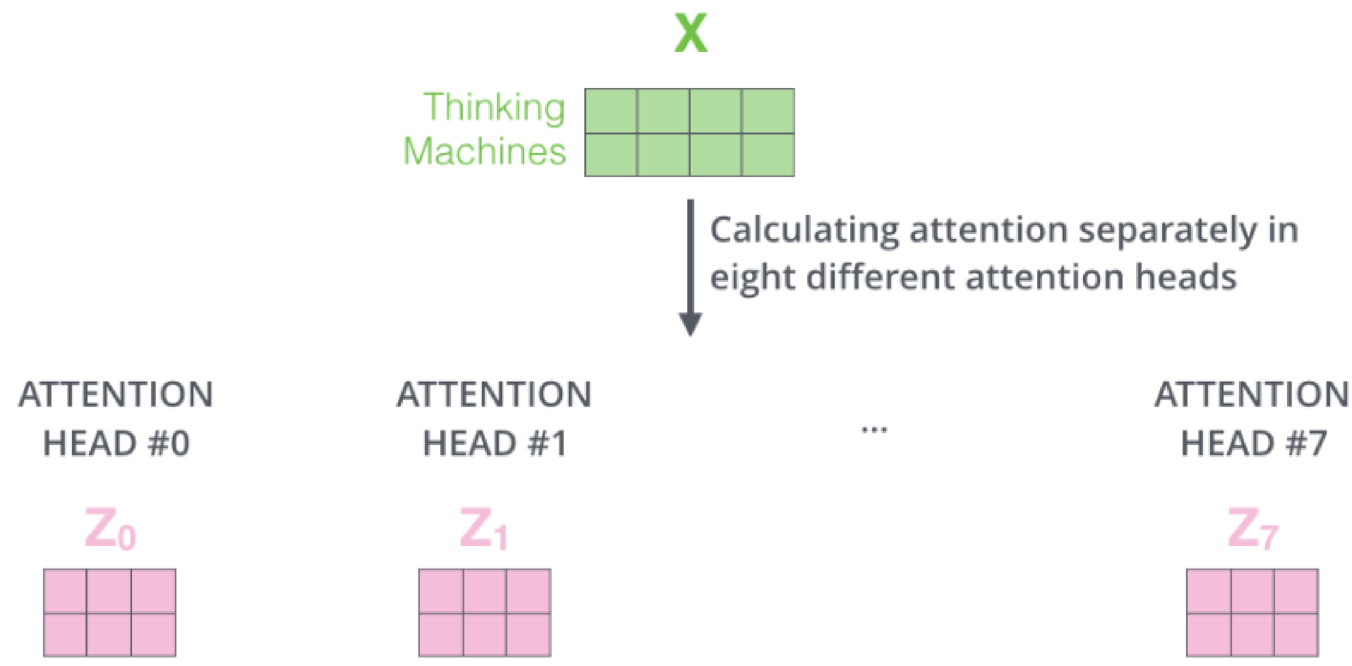
Transformers : Multi-head Attention



With multi-headed attention, we maintain separate Q, K, V weight matrices for each head resulting in different Q, K, V matrices.

[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

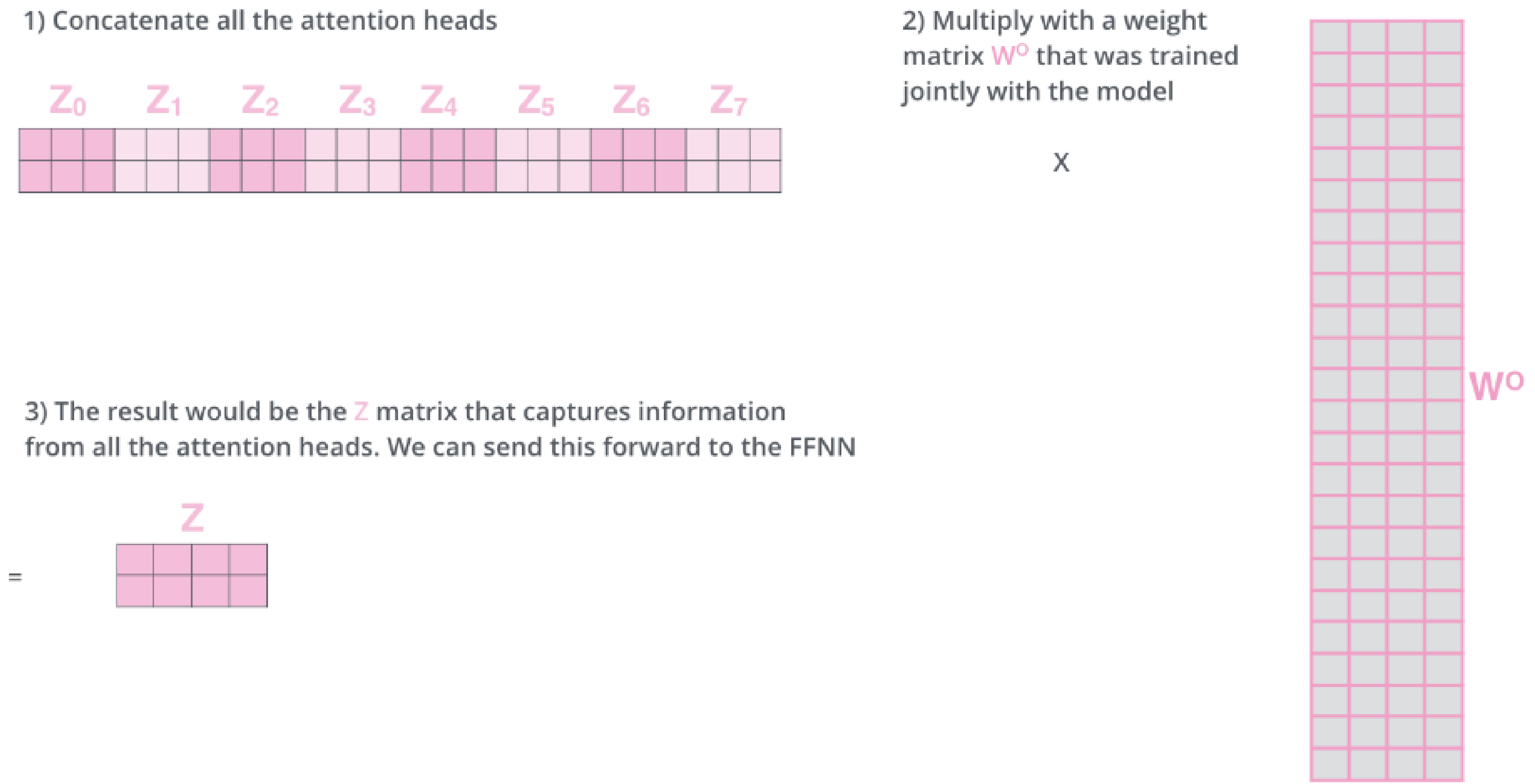
Transformers : Multi-head Attention



- Eight different weight matrices are obtained as an o/p.
- However, the feed-forward layer is not expecting eight matrices – it's expecting a single matrix (a vector for each word).

[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)](jalammar.github.io)

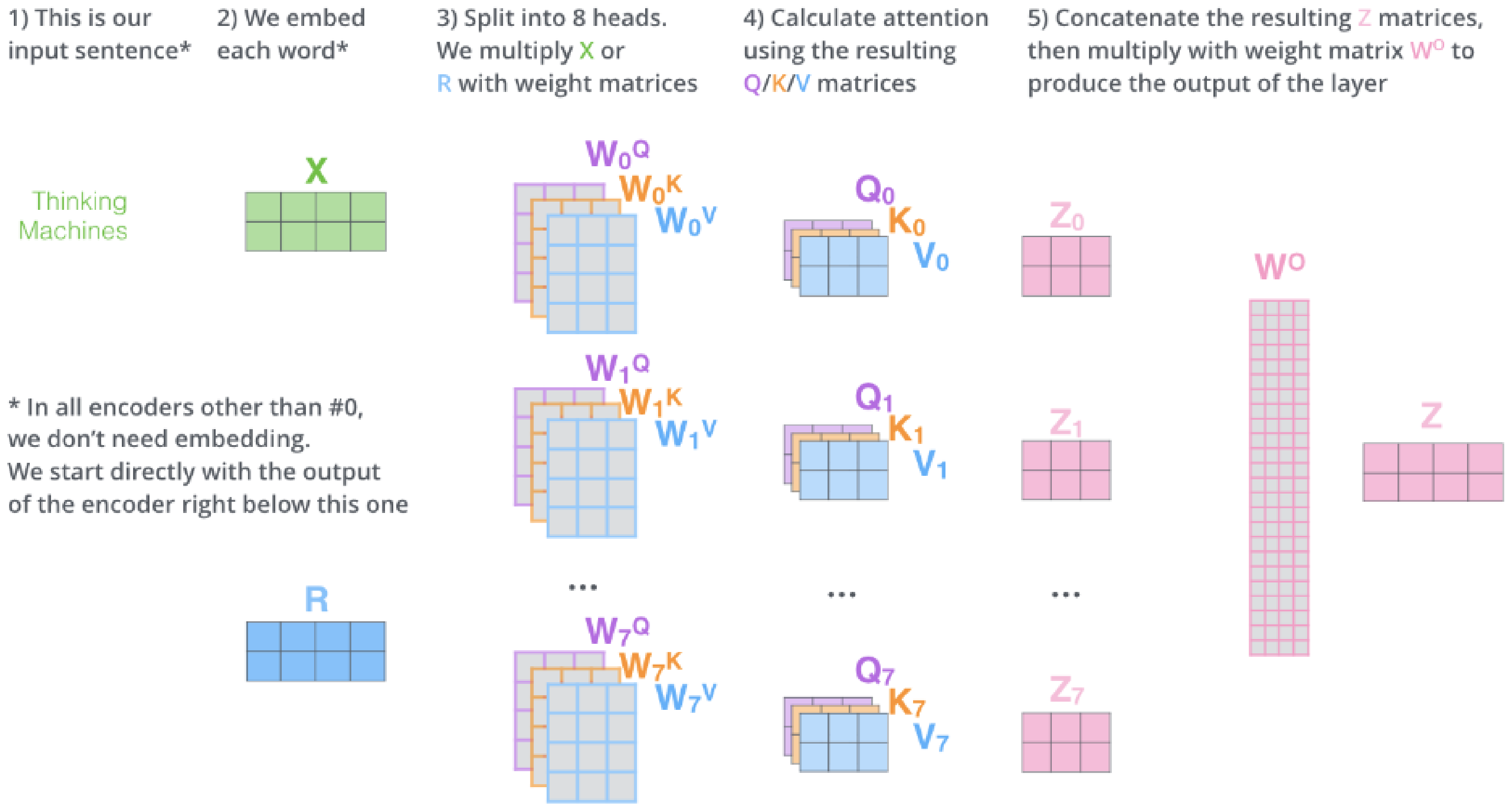
Transformers : Multi-head Attention



To tackle the issue, the eight matrices are concatenated and multiplied with additional weight matrix W^O

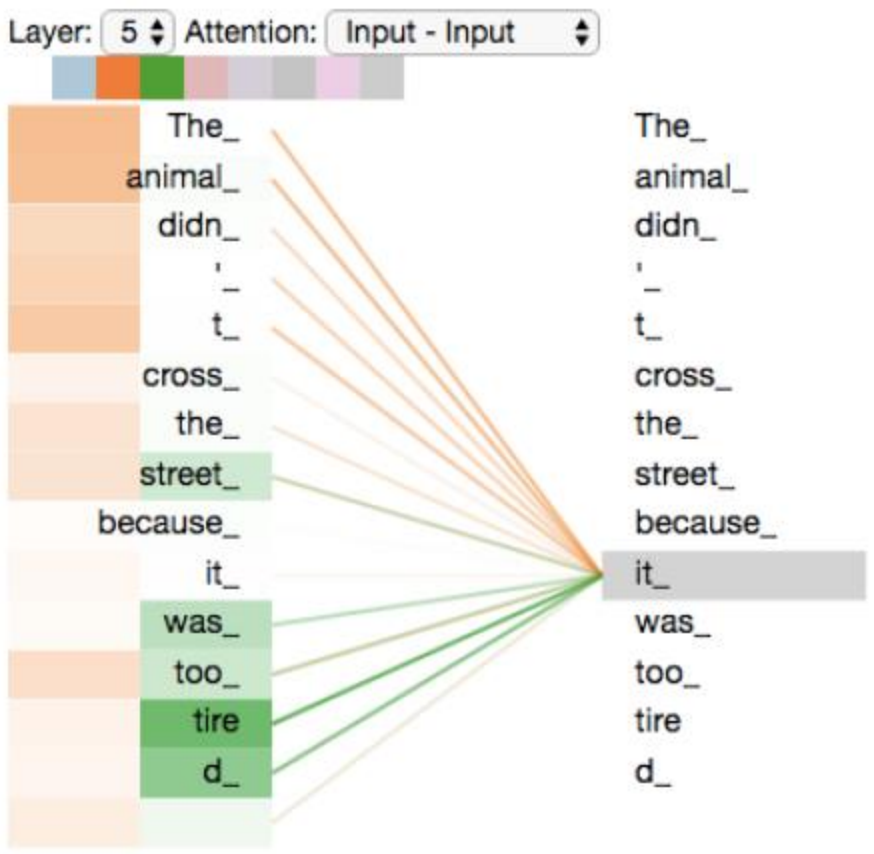
The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

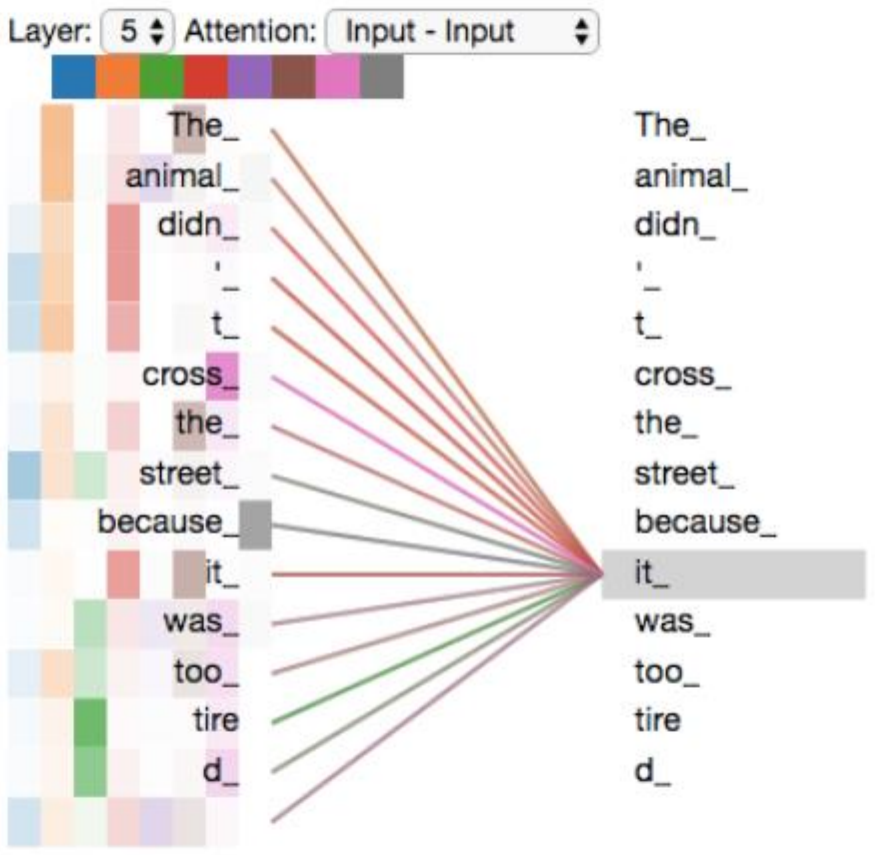
Transformers : Multi-head Attention



[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

Transformers : Multi-head Attention





- As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" – in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

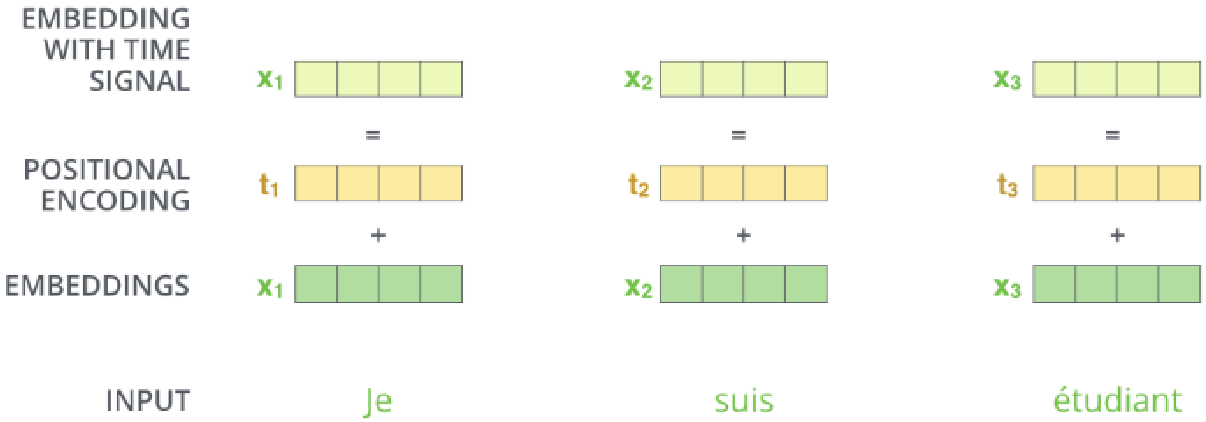
[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

Transformers : Positional Encoding

- Order of the sequence conveys important information for machine translation tasks and language modeling.
- Position encoding is a way to account for the order of words in the input sequence.
- The positional information of the input token in the sequence is added to the input embedding vectors.

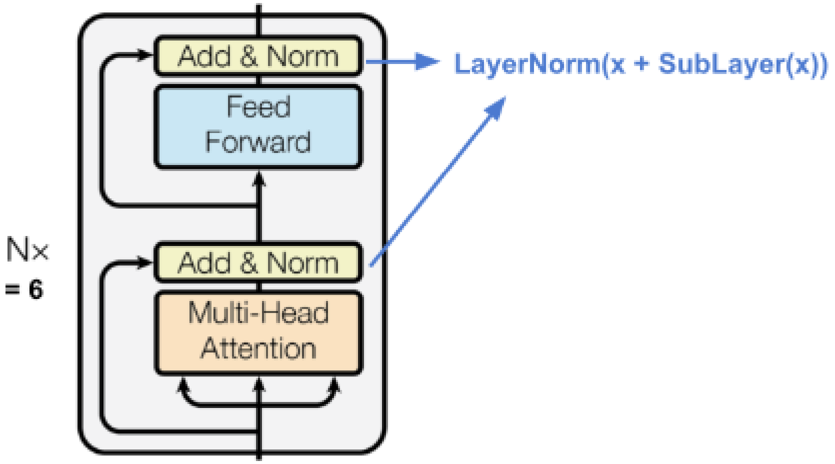
512

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



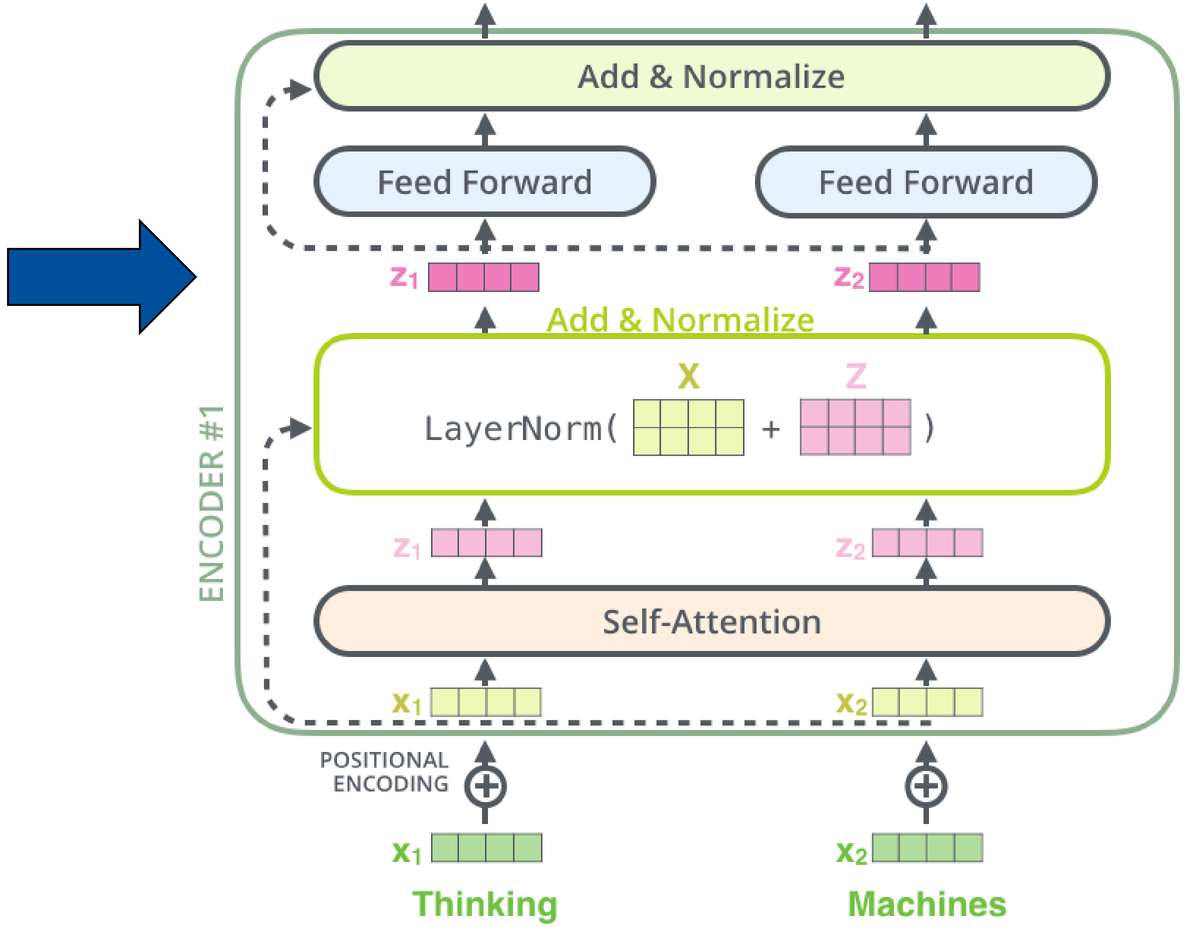
[The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

Transformers : Encoder



Layer Norm

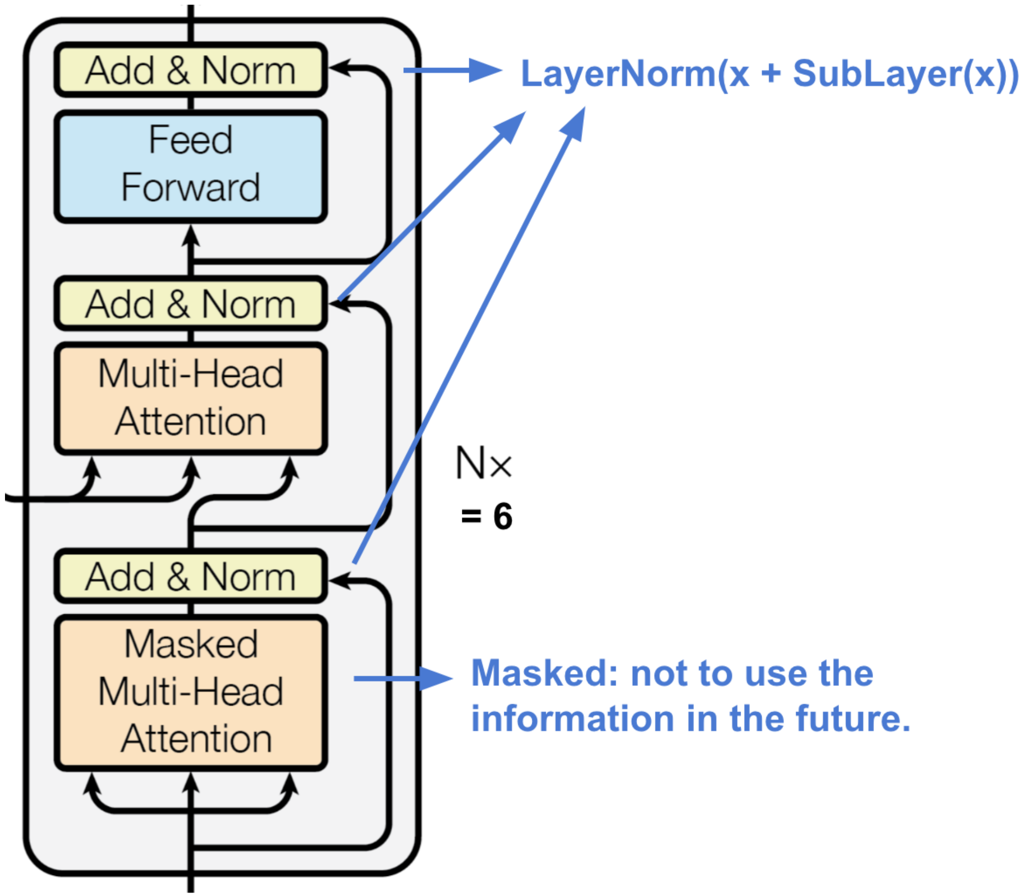
statistics are calculated across all features and all elements (words), for each instance(sentence) independently.



Attention? Attention! | Lil'Log (lilianweng.github.io)

The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Transformers : Decoder

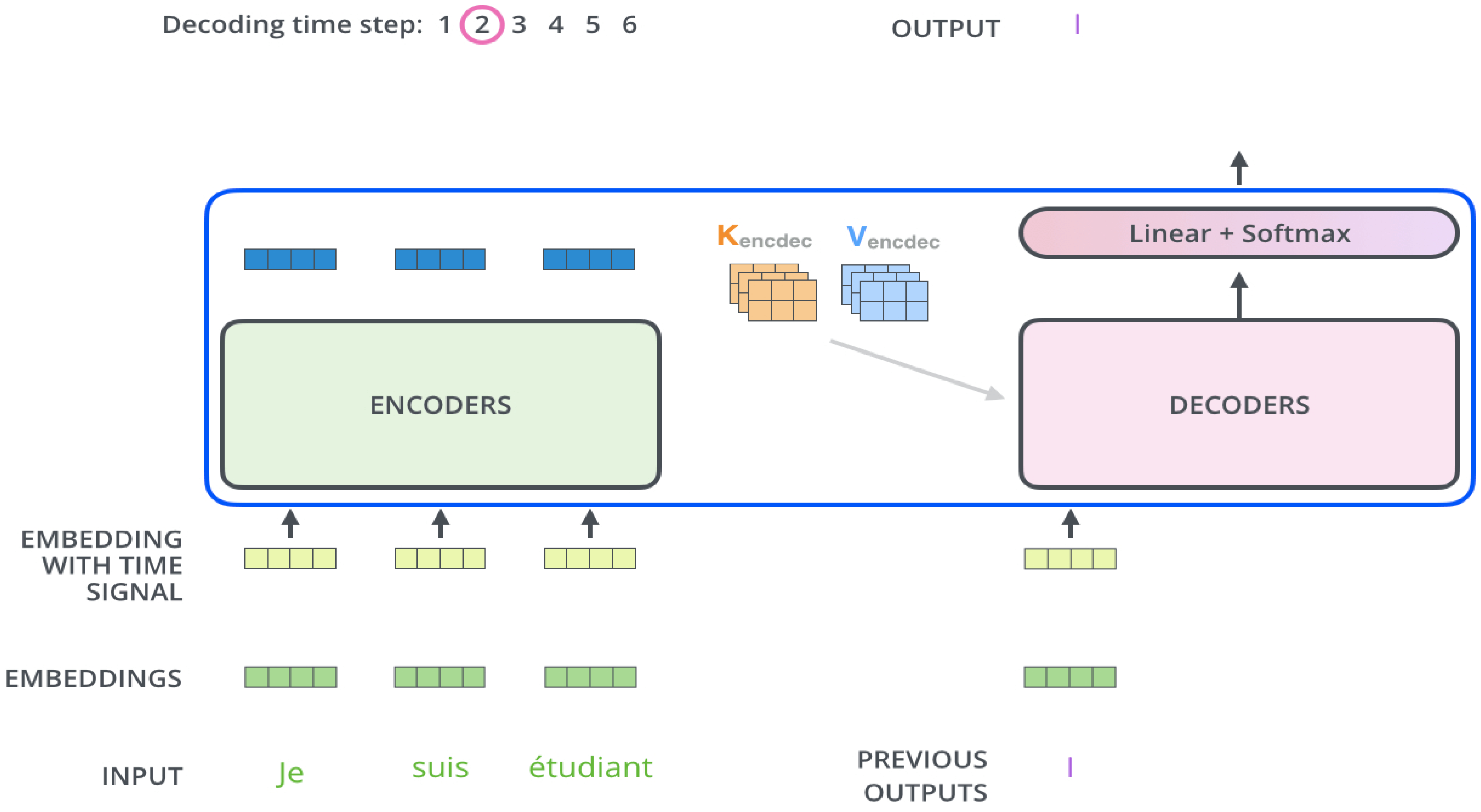


- **The Encoder**
  - processes the input sequence.
  - The output of the top encoder is transformed into a set of attention vectors K and V.
- Vectors K & V are used by each decoder in its “encoder-decoder attention” layer which helps the decoder focus on appropriate places in the input sequence
- **The Decoder**
  - has masked multi-head attention layer to prevent the positions from seeing subsequent positions
  - The “Encoder-Decoder Attention” layer creates its Queries matrix from the layer below it, and takes the Keys and Values matrix from the output of the encoder stack

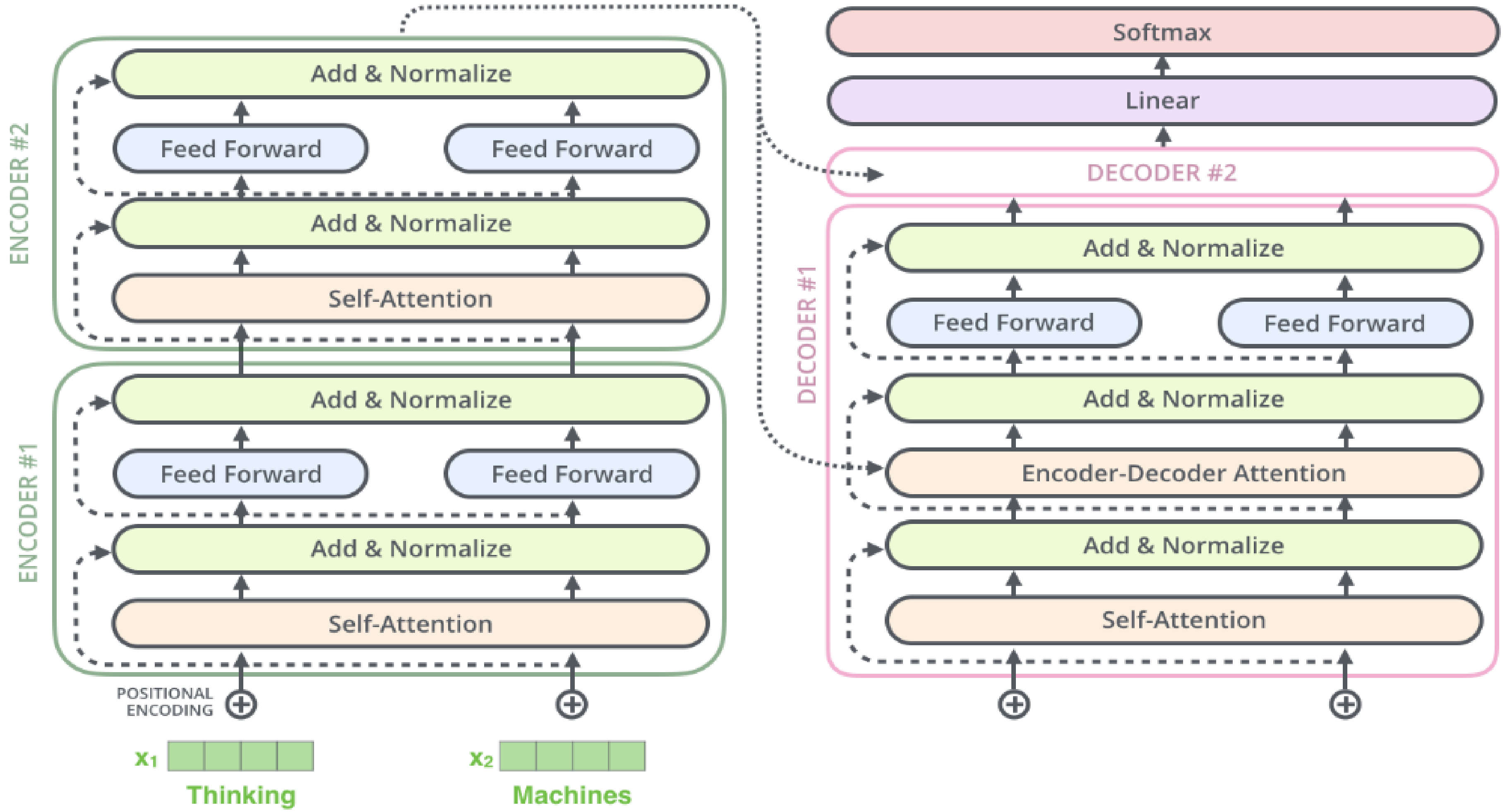
Attention? Attention! | Lil'Log (lilianweng.github.io)

The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Transformers: Encoders & Decoders



Transformers : Decoder



The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Massive Deep Learning Language models

- **Language models**
  - estimate the probability of words appearing in a sentence, or of the sentence itself existing.
  - building blocks in a lot of NLP applications
- **Massive deep learning language models**
  - pretrained using an enormous amount of unannotated data to provide a general-purpose deep learning model.
  - Downstream users can create task-specific models with smaller annotated training datasets (transfer learning)
- **Tasks executed with BERT and GPT models:**
  - **Natural language inference**
    - enables models to determine whether a statement is true, false or undetermined based on a premise.
    - For example, if the premise is “tomatoes are sweet” and the statement is “tomatoes are fruit” it might be labelled as undetermined.
  - **Question answering**
    - model receives a question regarding text content and returns the answer in text, specifically marking the beginning and end of each answer.
  - **Text classification**
    - is used for sentiment analysis, spam filtering, news categorization

GPT (Generative Pre-Training) by Open AI Pretraining

- Unsupervised learning served as **pre-training** objective for supervised fine-tuned models
  - generative language model using unlabeled data
  - then fine-tuning the model by providing examples of specific downstream tasks like classification, sentiment analysis, textual entailment etc.
- **Semi supervised learning using 3 components**
  1. **Unsupervised Language Modelling** (Pre-training)

$$L_1(T) = \sum_i \log P(t_i | t_{i-k}, \ldots, t_{i-1}; \theta)$$

  2. where
     1. k is the size of the context window, and ii) conditional probability P is modeled with the help of a neural network (NN) with parameters Θ

GPT (Generative Pre-Training) by Open AI

2. **Supervised Fine-Tuning**: maximising the likelihood of observing label y, given features or tokens x_1,…,x_n.

$$L_2(C) = \sum_{x,y} log\, P(y|x_1, \ldots, x_n)$$

where C was the labeled dataset made up of training examples.

3. **Auxiliary learning objective for supervised fine-tuning to get better generalisation and faster convergence**.

$$L_3(C) = L_2(C) + \lambda L_1(C)$$

where $L_1(C)$ was the auxiliary objective of learning language model

λ was the weight given to this secondary learning objective. λ was set to 0.5.

- Supervised fine-tuning is achieved by adding a linear and a softmax layer to the transformer model to get the task labels for downstream tasks.
- “zero-shot” framework
  - measure a model’s performance having *never* been trained on the task.

GPT-n series created by OpenAI (2018 onwards)

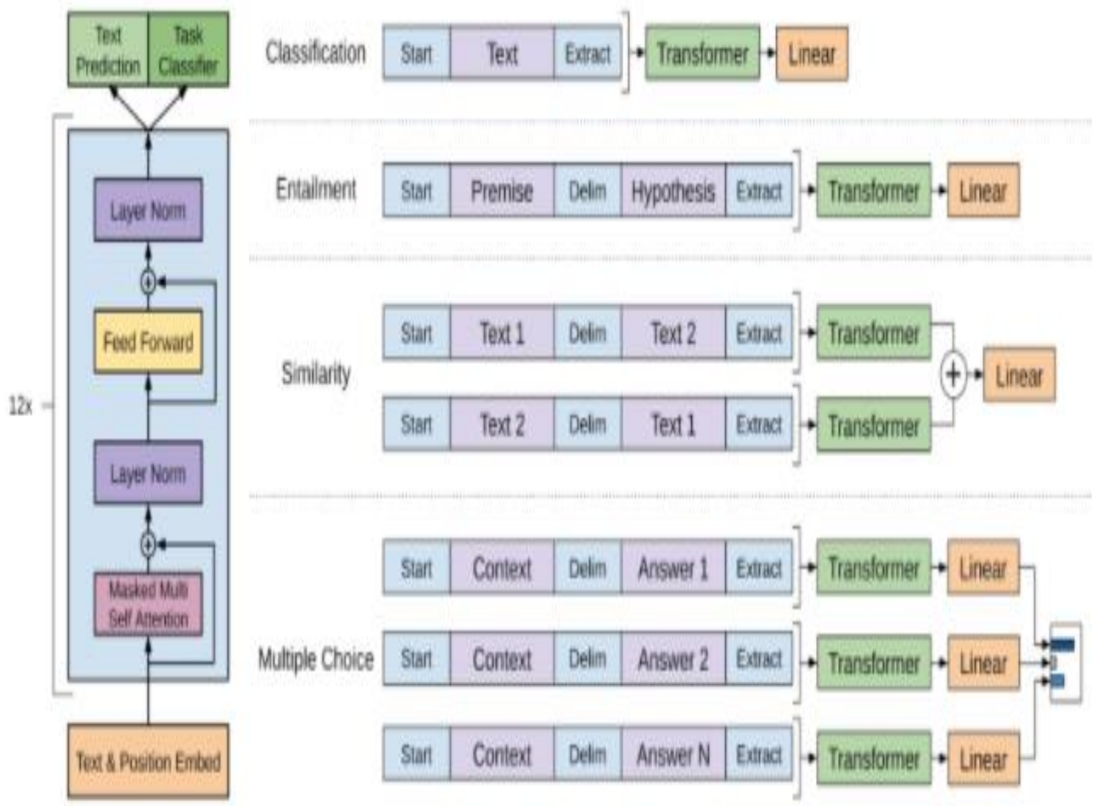


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

- **Generative** models are a type of statistical model that are used to generate new data points.
  - learn the underlying relationships between variables in a dataset in order to generate new data points similar to those in the dataset.
- Trained on BooksCorpus dataset contains over 7,000 unique unpublished books from a variety of genres
- 12-layer decoder-only transformer with masked self-attention heads
- GPT-2
  - Application - generate long passages of coherent text
  - https://transformer.huggingface.co/doc/gpt2-large

GPT-3 (2020)

- step to **A**rtificial **G**eneral **I**ntelligence(AGI)

"I am open to the idea that a worm with 302 neurons is conscious, so I am open to the idea that GPT-3 with 175 billion parameters is conscious too." — David Chalmers

- 3rd-gen language prediction model in capacity of **175 billion** parameters.
- trained with 499 Billion tokens
- trained using next word prediction
- Context window size was increased from 1024 for GPT-2 to 2048 tokens for GPT-3.
- Size of word embeddings was increased to 12888 for GPT-3 from 1600 for GPT-2.
- To train models of different sizes, the batch size is increased according to number of parameters, while the learning rate is decreased accordingly.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

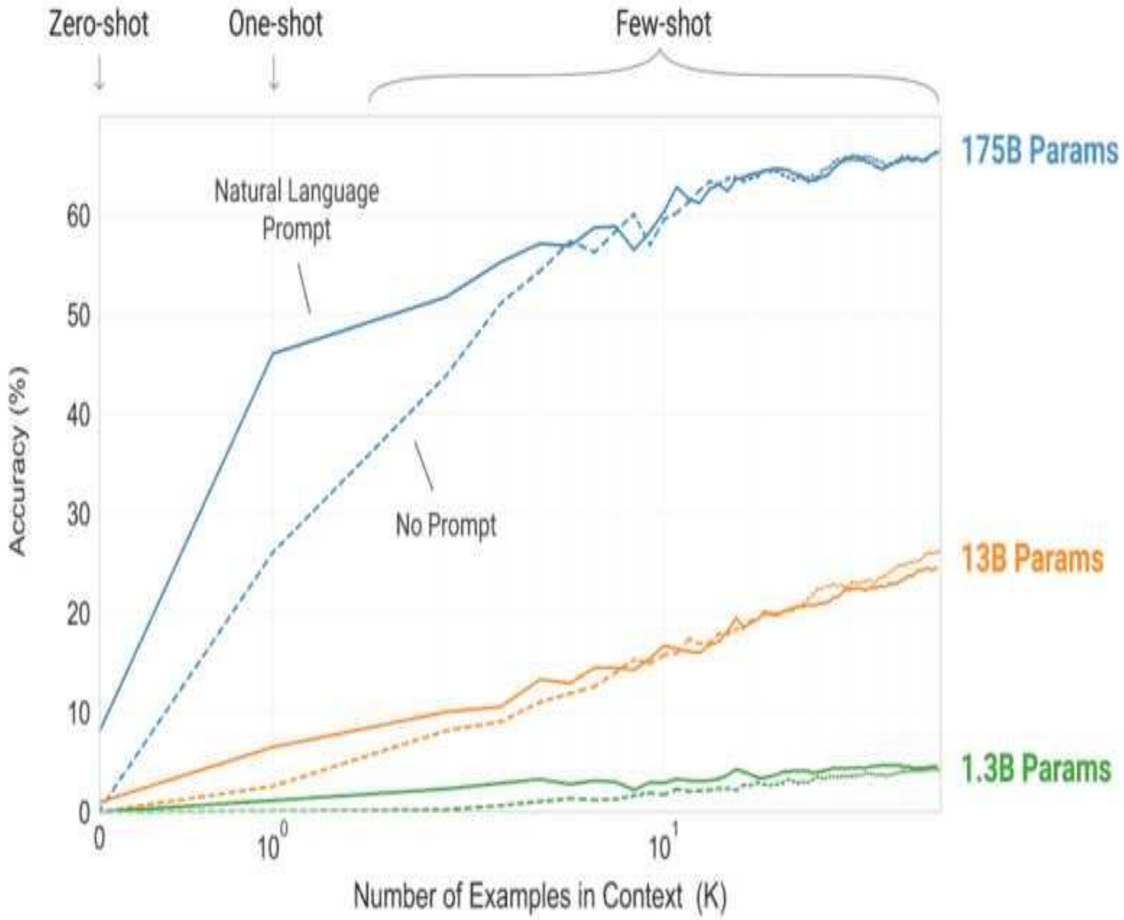
GPT-3

zero-shot task transfer OR meta learning

- The model is supposed to understand the task based on the examples and instruction.
- For English to French translation task, the model was given an English sentence followed by the word French and a prompt (:)

GPT-3 Task Agnostic Model

- **LIMITATIONS OF GPT-3**
- GPT-3 can perform a wide variety of operations such as compose prose, write code and fiction, business articles
- Does not have any internal representation of what these words even mean. misses the *semantically grounded model* of the topics on which it discusses.
- If the model is faced with data that is not in a similar form or is unavailable from the Internet's corpus of existing text that was used initially in the training phase, then the language generated is a loss.
- Expensive and complex inferencing due to hefty architecture, less interpretability of the language, and uncertainty around what helps the model achieve its few-shot learning behavior.
- The text generated carries bias of the language it is initially trained on.
- The articles, blogs, memos generated by GPT-3 may face gender, ethnicity, race, or religious bias.
- model is capable of producing high-quality text, sometimes looses coherence with data while generating long sentences and thus may repeat sequences of text again and again in a paragraph.

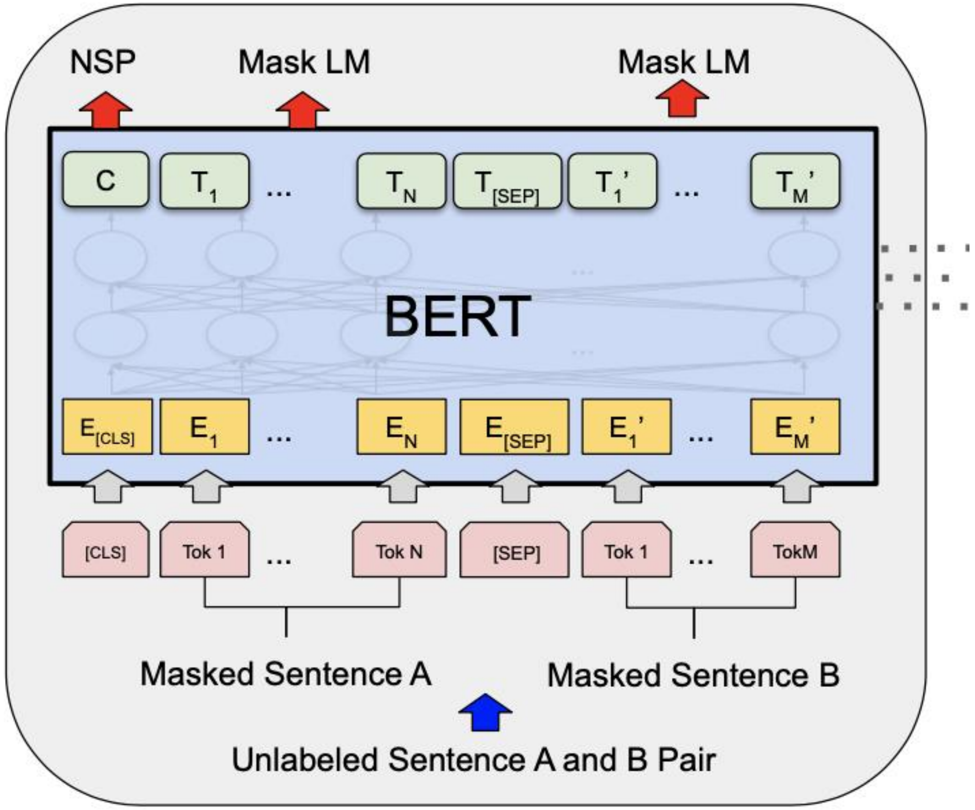


BERT (Bidirectional Encoder Representations from Transformers) by google

- "BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks."
- Two variants
  - BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters
  - BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters
- **BERT is pre-trained on two NLP tasks**:
  - Masked Language Modeling
    - replace 15% of the input sequence with [MASK] and model learns to detect the masked word
  - Next Sentence Prediction
    - two sentences A and B are separated with the special token [SEP] and are formed in such a way that 50% of the time B is the actual next sentence and 50% of the time is a random sentence.

BERT (Bidirectional Encoder Representations from Transformers) by google

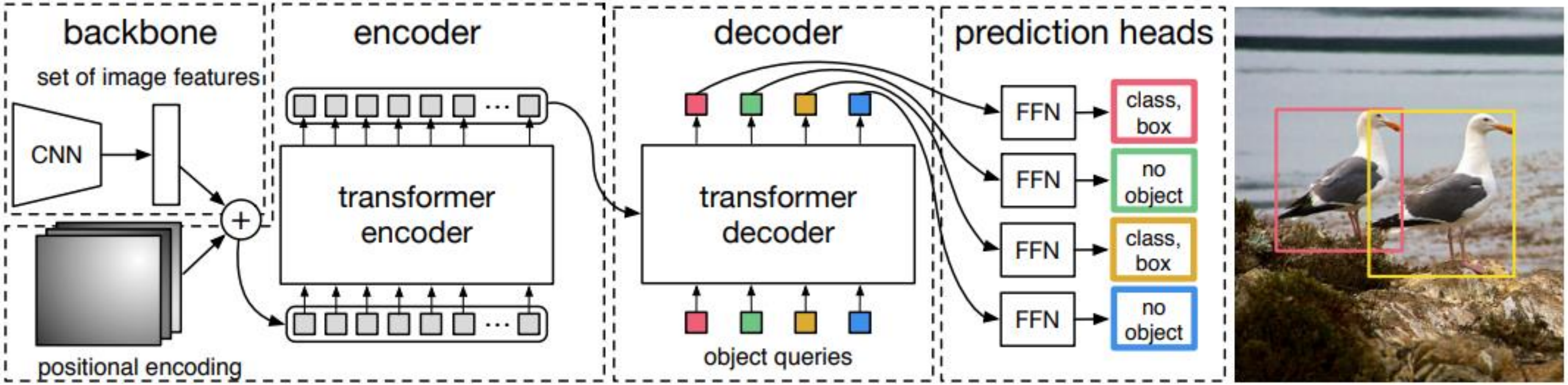
- every input embedding is a combination of 3 embeddings:
  - Position Embeddings: captures “sequence” or “order” information
  - Segment Embeddings: can also take sentence pairs as inputs for tasks (Question-Answering)
  - Token Embeddings: learned for the specific token from the WordPiece token vocabulary
- Output is an embedding since it has only encoder and no decoder



Pre-training

Transformers in Computer Vision

DEtection TRansformer (DETR)

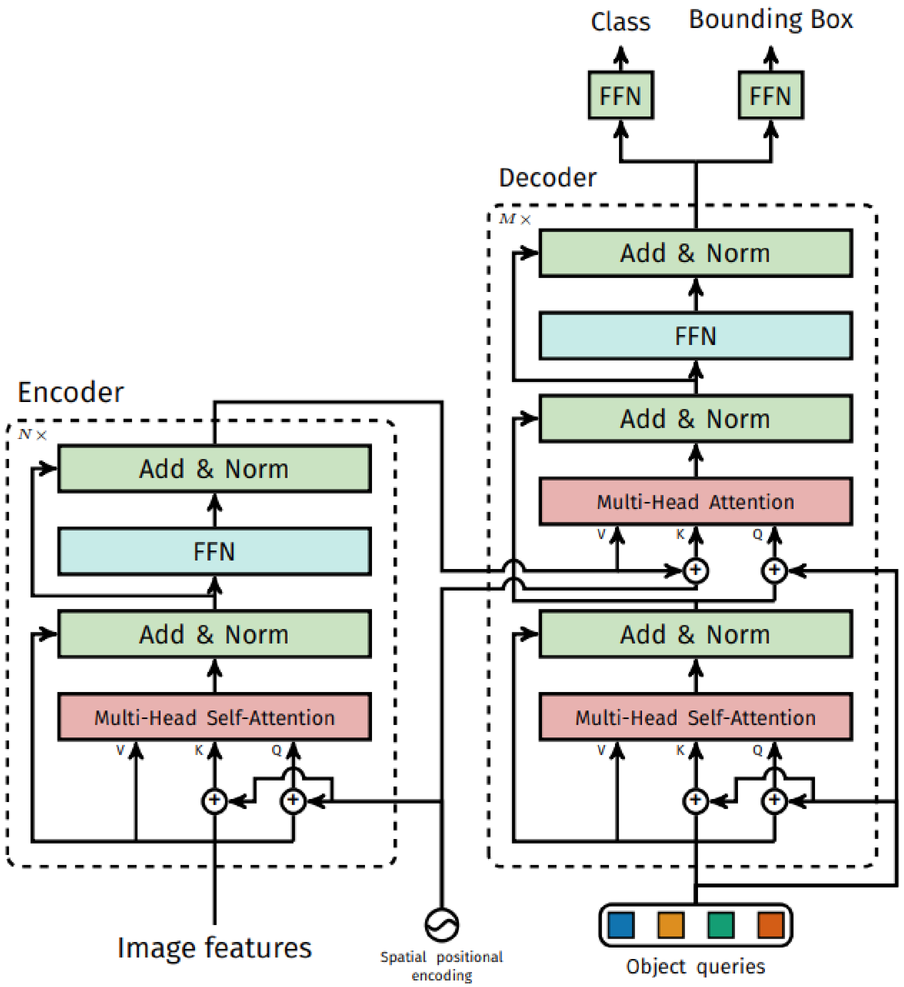


https://github.com/facebookresearch/detr

Carion, Nicolas, et al. "End-to-end object detection with transformers." *European conference on computer vision*. Springer, Cham, 2020.

Transformers in Computer Vision

DEtection TRansformer (DETR)



https://github.com/facebookresearch/detr

Carion, Nicolas, et al. "End-to-end object detection with transformers." *European conference on computer vision*. Springer, Cham, 2020.

Transformers in Computer Vision

DEtection TRansformer (DETR): Results on COCO 2017 dataset (AP = Average Precision)

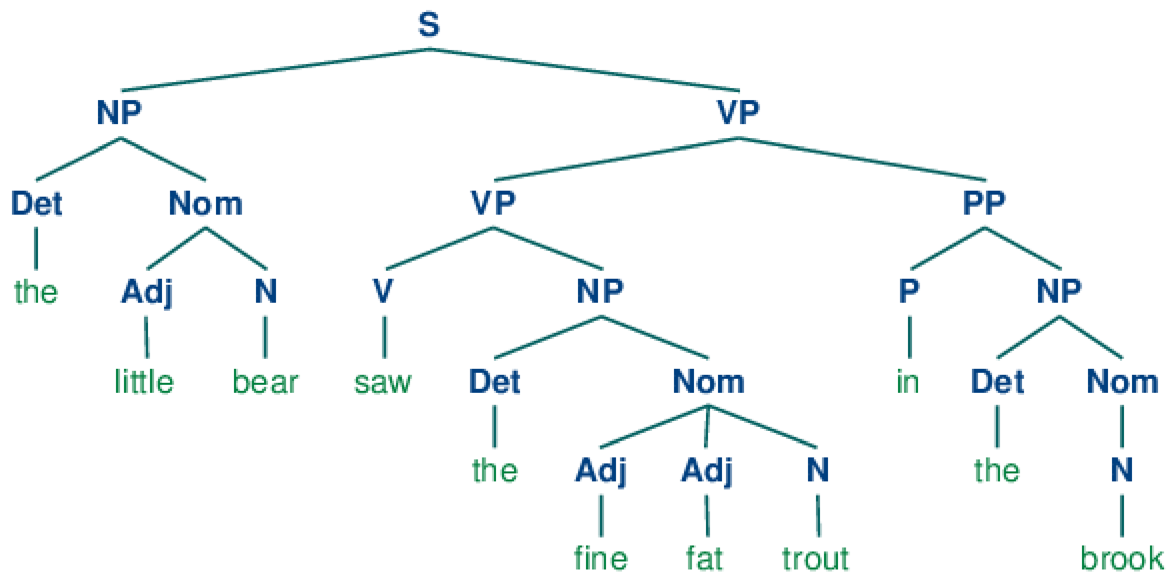
Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

https://github.com/facebookresearch/detr

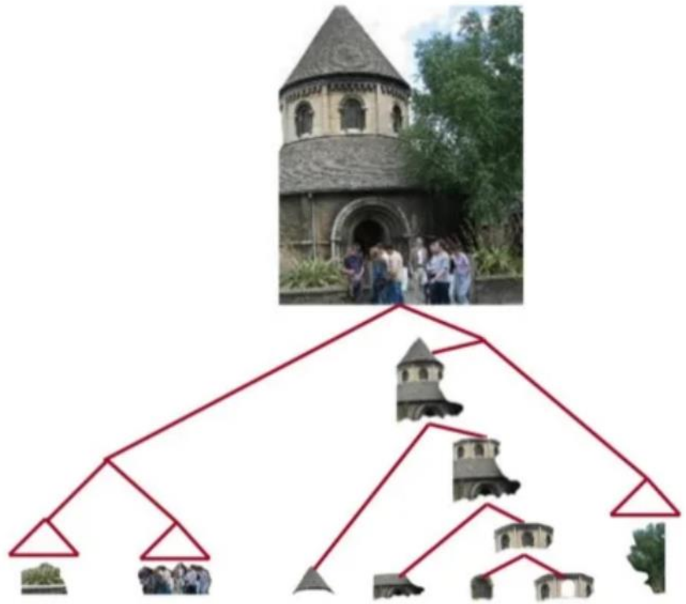
Carion, Nicolas, et al. "End-to-end object detection with transformers." *European conference on computer vision*. Springer, Cham, 2020.

Recursive Neural Networks

- Many real world entities have recursive structure.



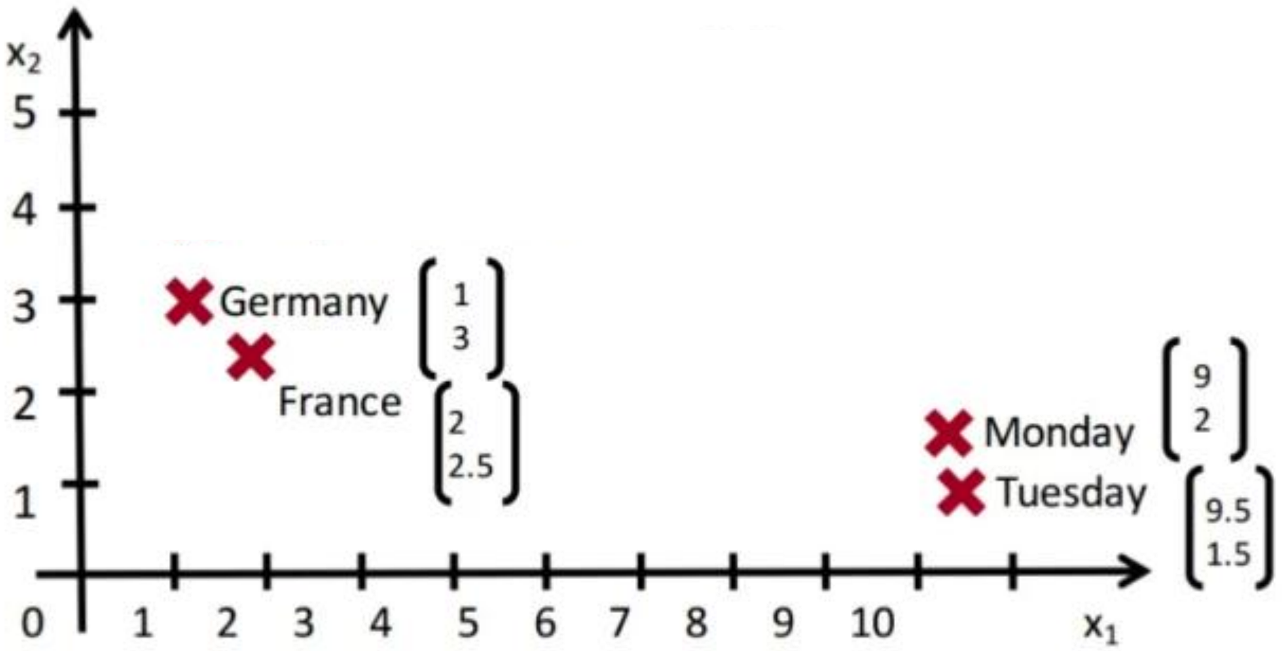
Eg: A syntactic tree structure representing a sentence.



Eg: A tree representation of different segments in an image

Recursive Neural Networks: Introduction

- Can we learn a good representation for these recursive structures?



the country of my birth

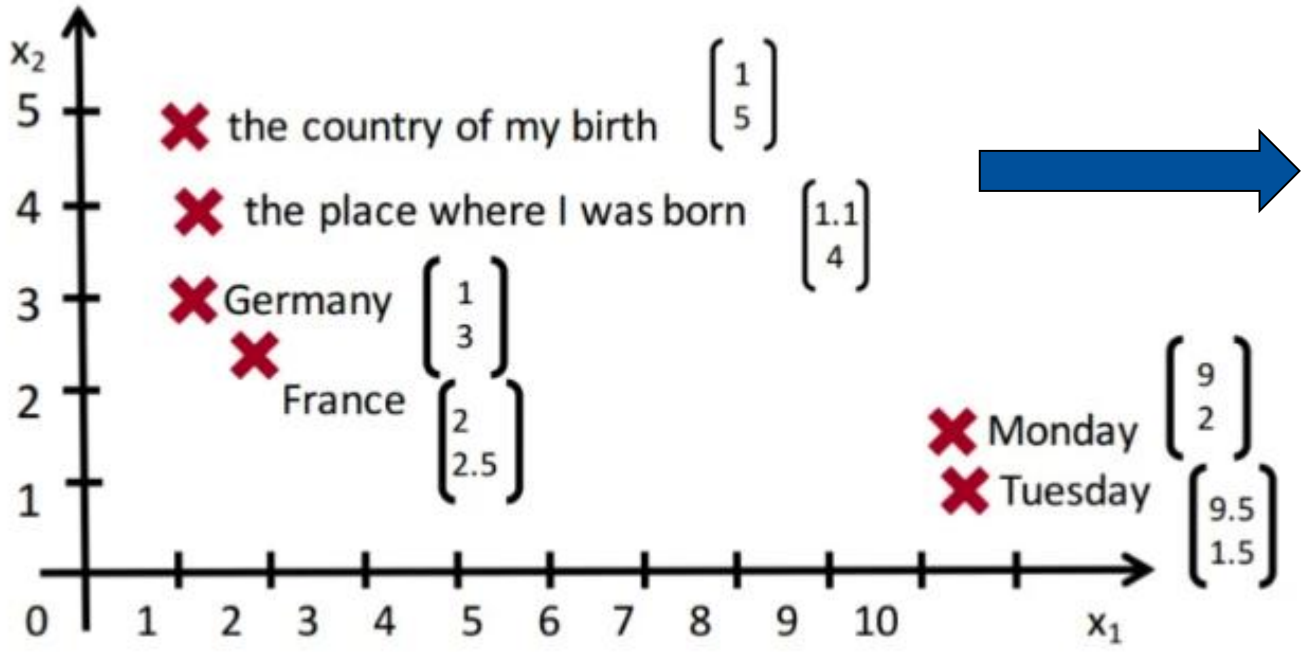
the place where I was born

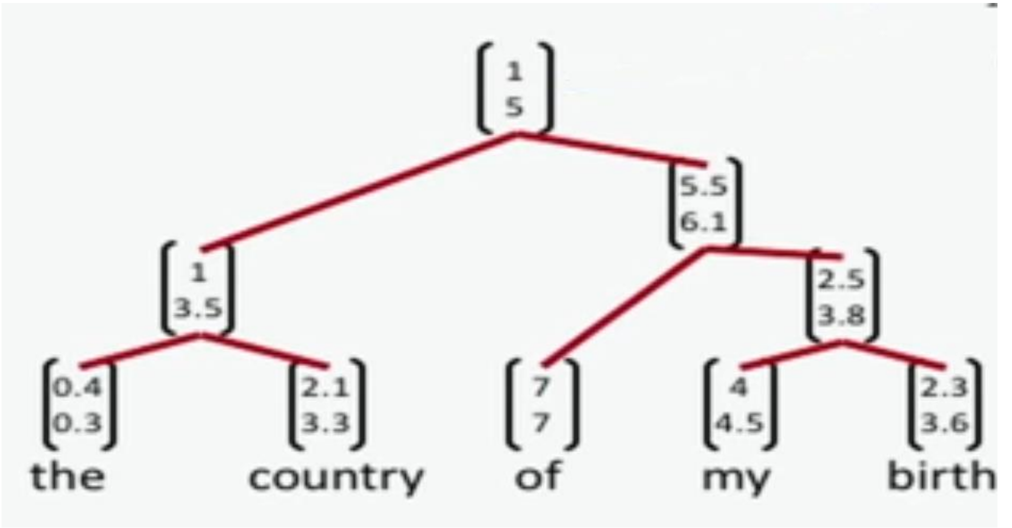
Eg: Word phrase representations (learning representations of phrases of arbitrary length)

Source: Socher et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML'11

Recursive Neural Networks: Introduction

- Can we learn a good representation for these recursive structures?

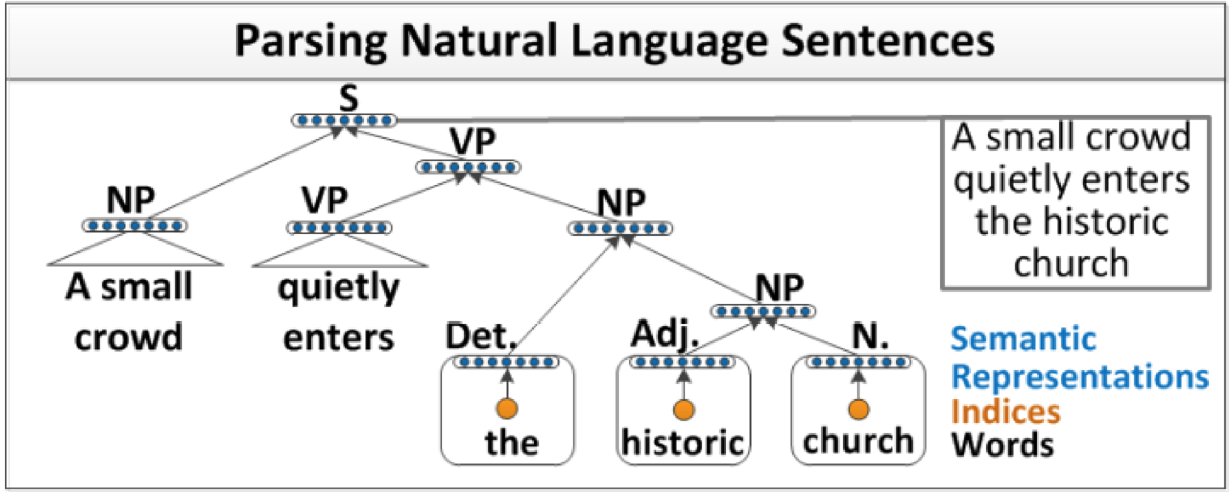


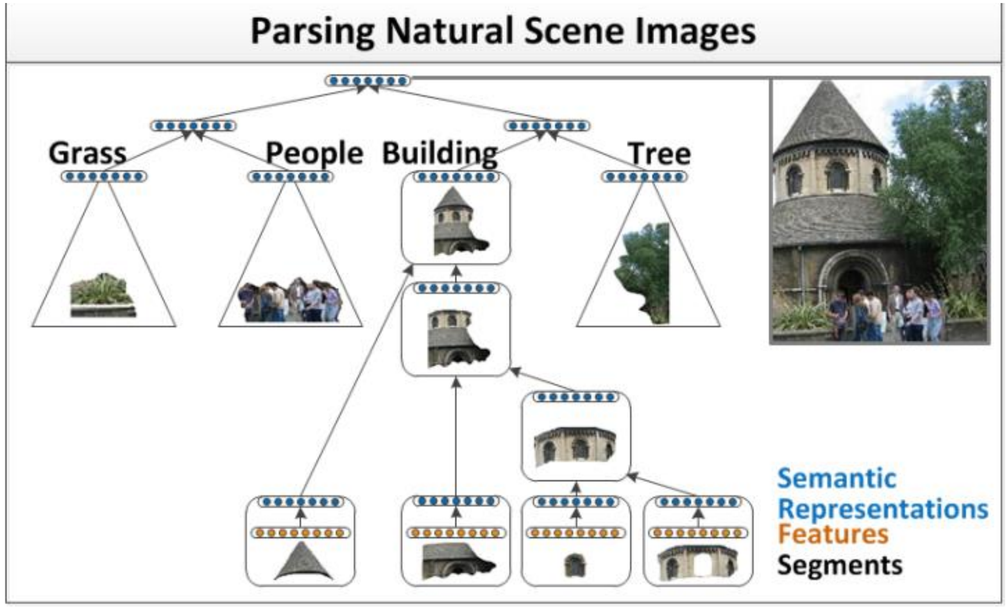


- The meaning of a sentence is determined by meaning of its words and the rules that combine them.

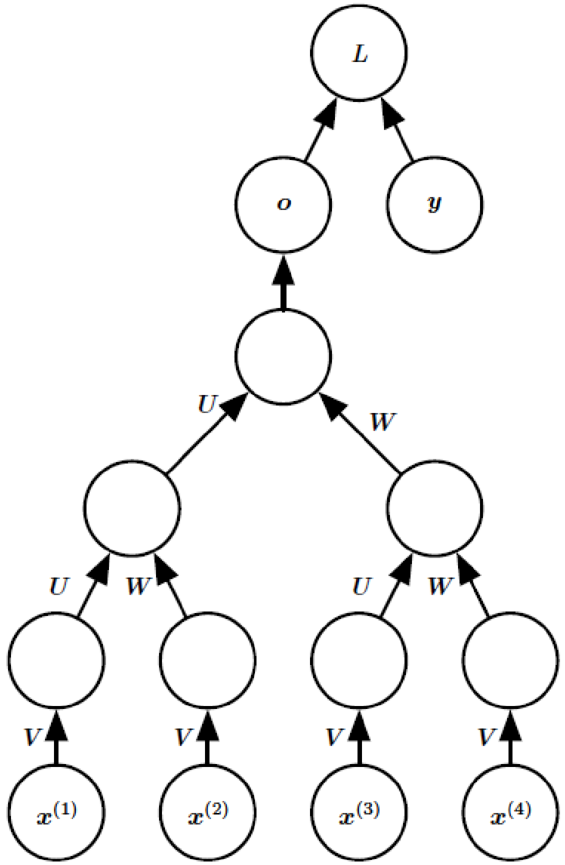
Source: Socher et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML'11

Recursive Neural Networks



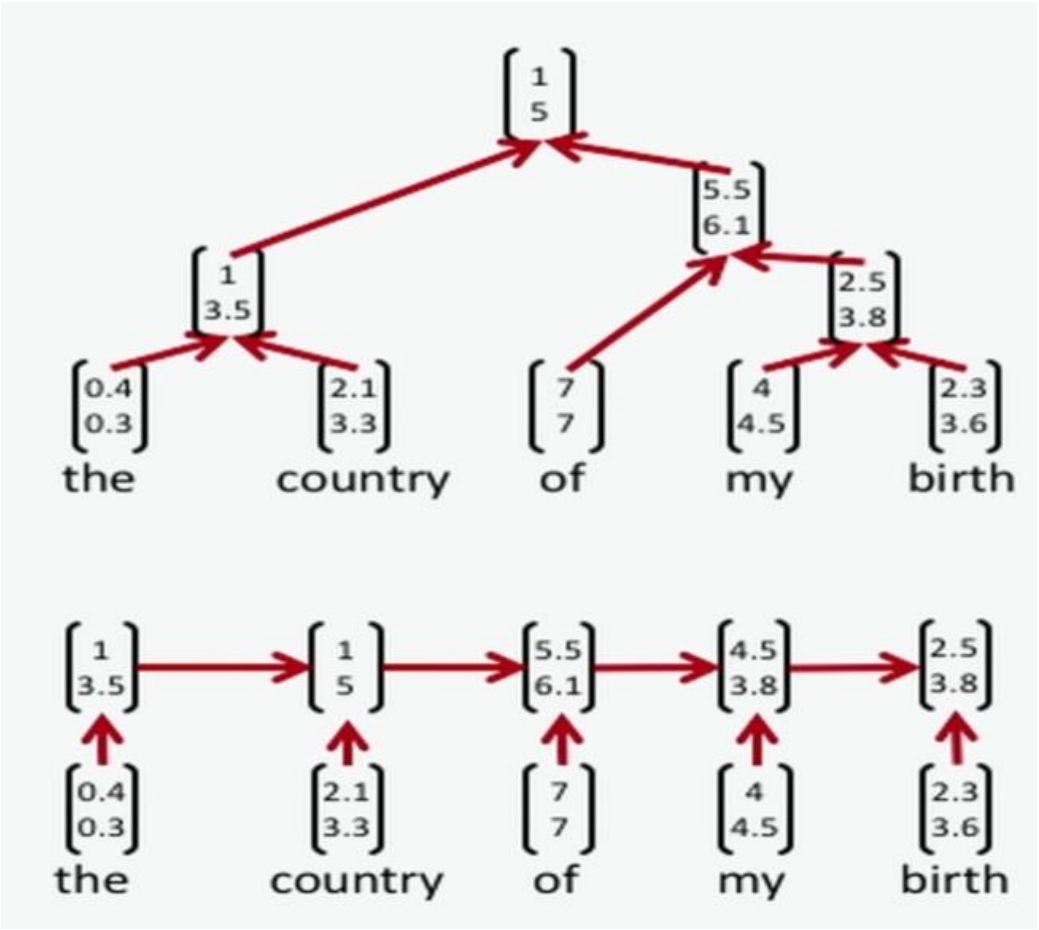


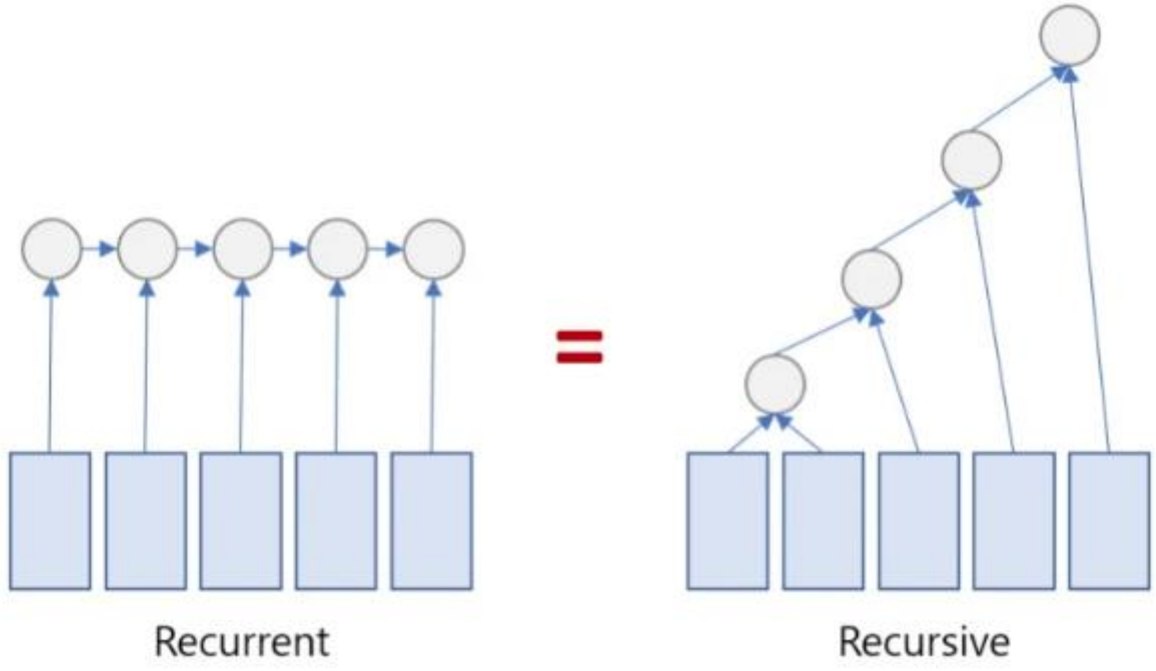
Source: Socher et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML'11



Source: Ian Goodfellow et al. "Deep Learning" MIT Press'15

Recursive Neural Networks vs Recurrent Neural Networks

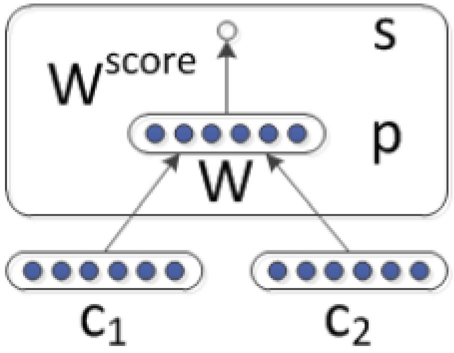




Source: Socher et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML'11

Recursive Neural Networks

- To build a recursive neural network, we need two things:
  - A model that merges pairs of representations.
  - A model that determines the tree structure.



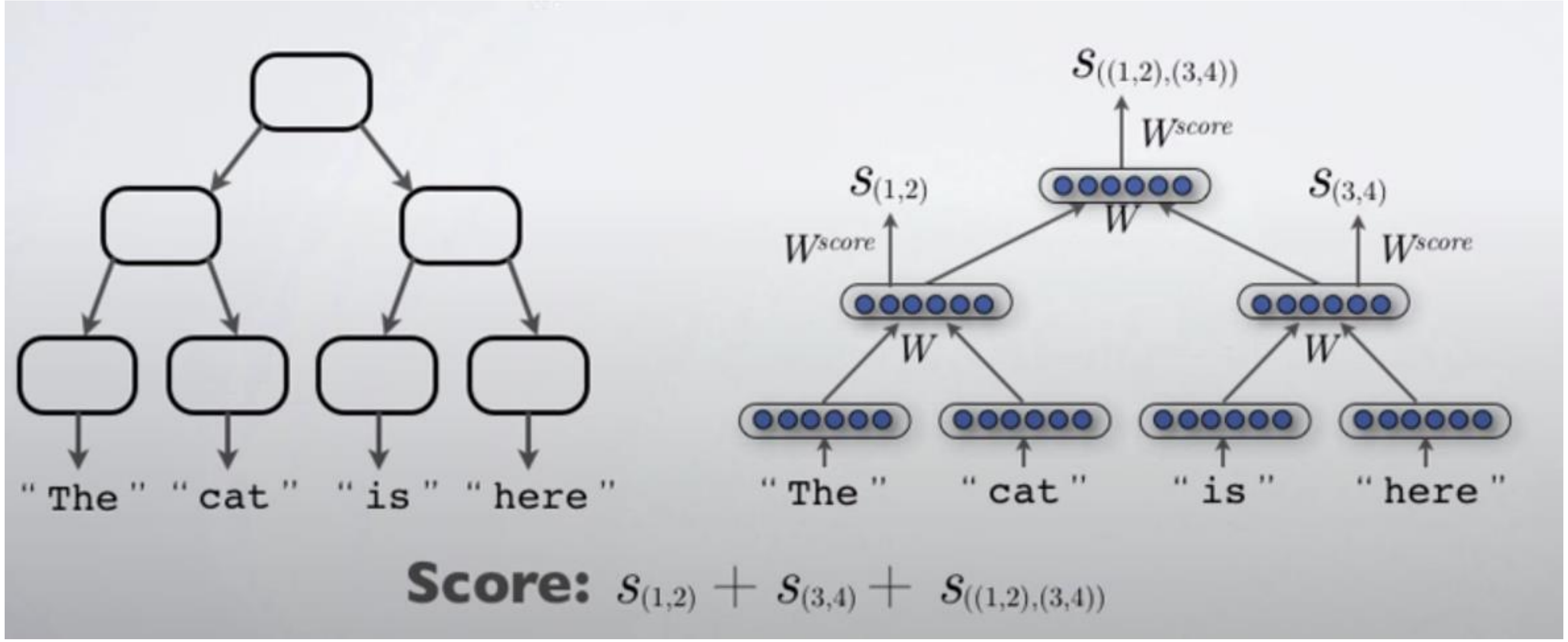
$$\boxed{s = W^{score} p}$$
$$p = f(W[c_1; c_2] + b)$$

A score to determine which pairs of representations to merge first

Source: Socher et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML'11

Recursive Neural Networks

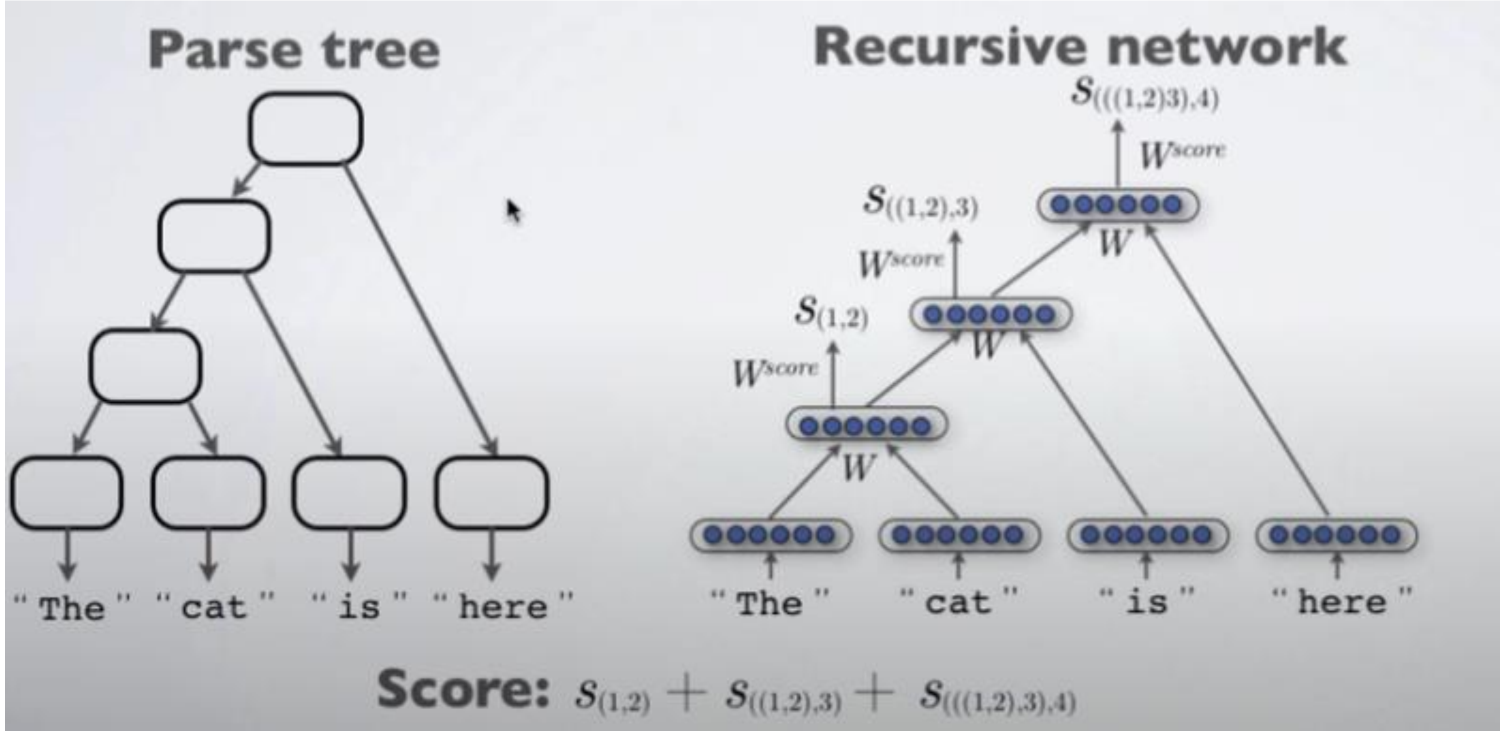
- To build a recursive neural network, we need two things:
  - A model that merges pairs of representations.
  - A model that determines the tree structure.



Source: Socher et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML'11

Recursive Neural Networks

- To build a recursive neural network, we need two things:
  - A model that merges pairs of representations.
  - A model that determines the tree structure.



- Approximate the best tree by locally maximizing each subtree

Source: Socher et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML'11