TEXT PROCESSING WITH MACHINE LEARNING

MODULE 3: Advanced DNN systems

Dr. Aobo Wang
isswan@nus.edu.sg

Agenda

- Day 3 Advanced DNN systems
  - Attention
  - Transformer
  - Workshop
  - Sentence/ Document representation
  - Workshop

Attention Mechanism

- Many of the new advances in NLP starts from attention - transformer, BERT … …

- *“Every once in a while, a revolutionary product comes along that changes everything.” – Steve Jobs*

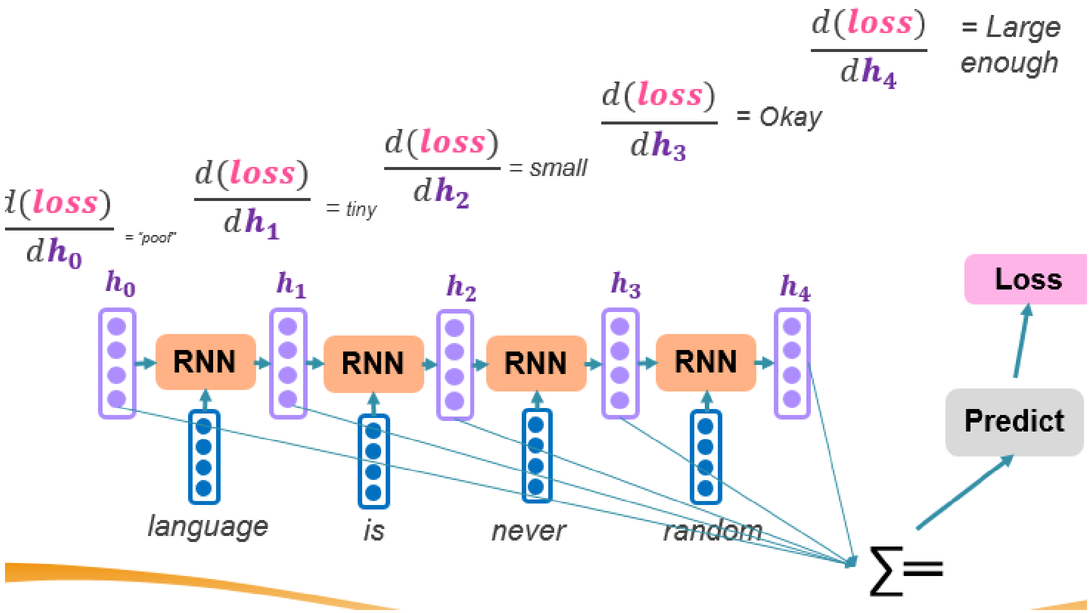
Attention is revolutionary!

- In psychology, attention is the cognitive process of selectively concentrating on one or a few things while ignoring others.

Attention Mechanism

- A neural network to mimic human brain actions in a simplified manner.
- Attention **selectively** concentrating on a few relevant things
- Useful in sequence modelling – which part of the sequence should you bring to attention?
- More parameters and calculations to capture contextual information not sequentially but selectively

Problem: Vanishing gradients



- prevents parallelization.
- 'C" does not sufficiently capture the prior information
- To resolve this, in comes the attention mechanism introduced by **Bahdanau et al** in 2015

Attention – what does it do exactly?

- What does it mean for ‘good’ attention? Suppose a statement:

Laurent	Lives	In	France	And	he	Speaks	Excellent	‘French’

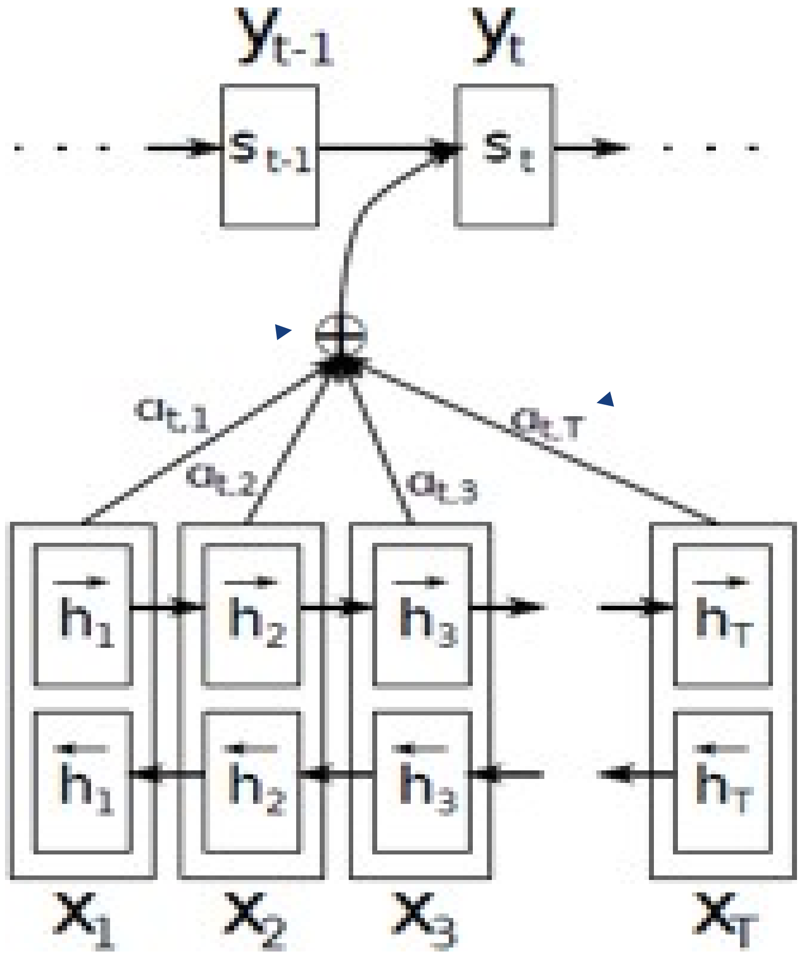
- To predict the next word ‘French’, which precedent word is most important with ‘weights’ (pay attention) to?
- In Bahhadau, the weights provide the necessary attention.

Bahdanau architecture model

- At every decoding step, the decoder allocates a set of ***attention weights*** **–** α, aligned to the input words.

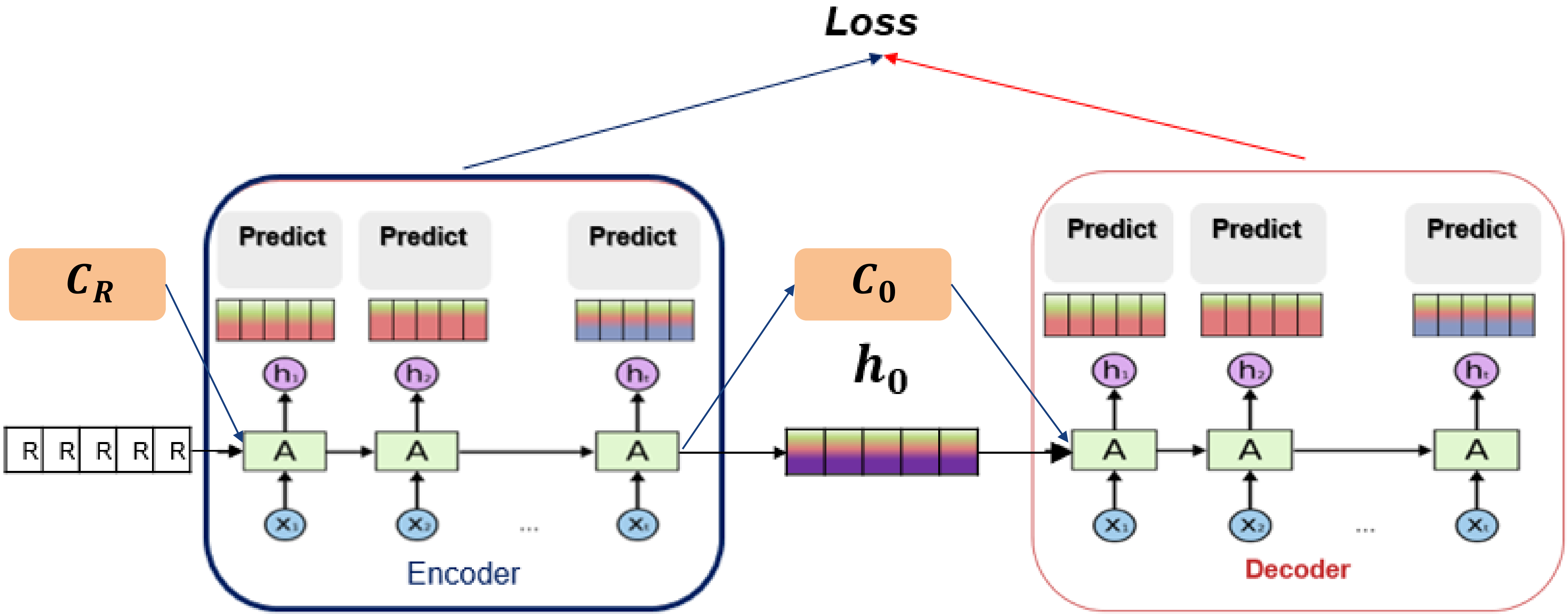
- Also known as **additive attention** as it performs a **linear combination** of encoder states and the decoder states.

- All hidden states of the encoder(forward and backward) and the decoder are used to generate the **context vector**, (instead of using just the last encoder hidden state)

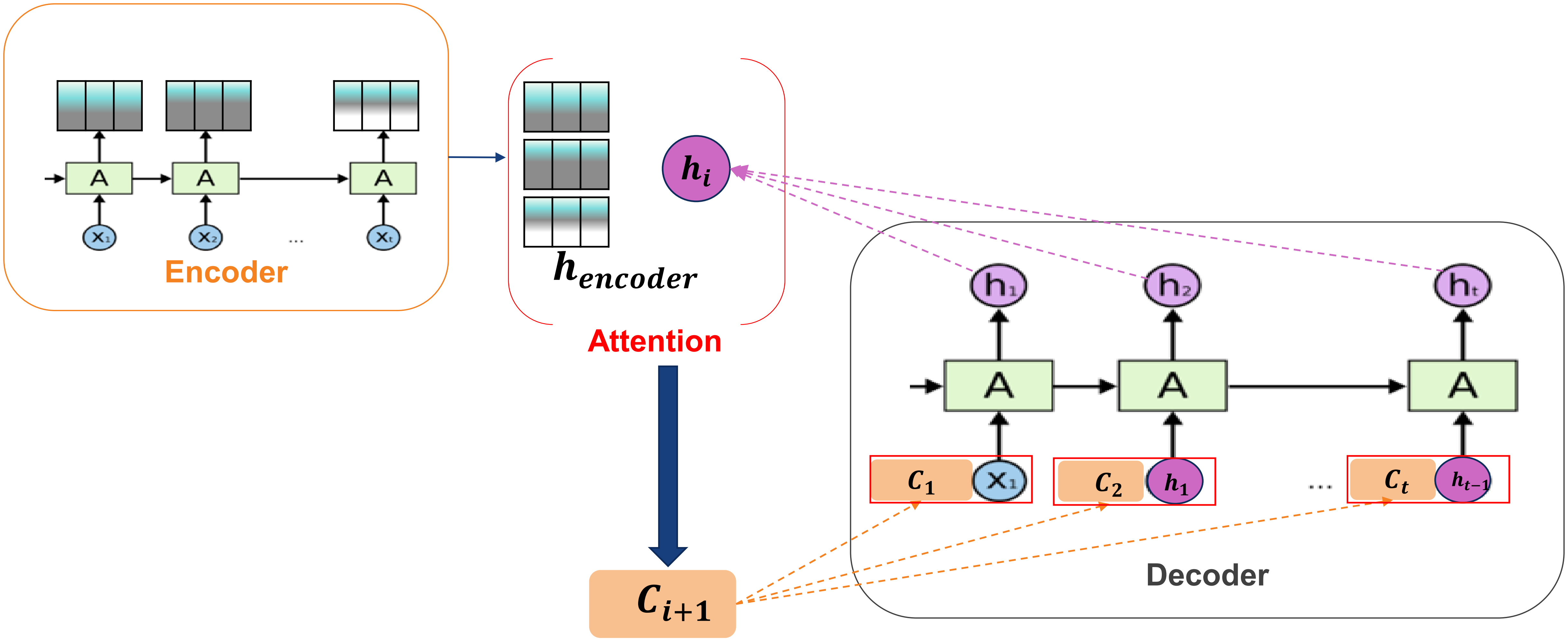


Sequence to Sequence

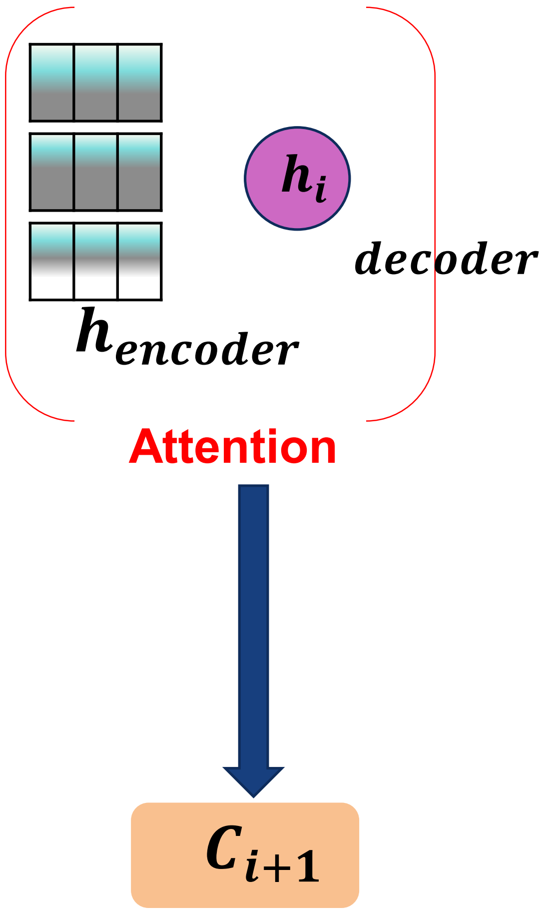
- "Translate" (Ideally) any object A to any object B

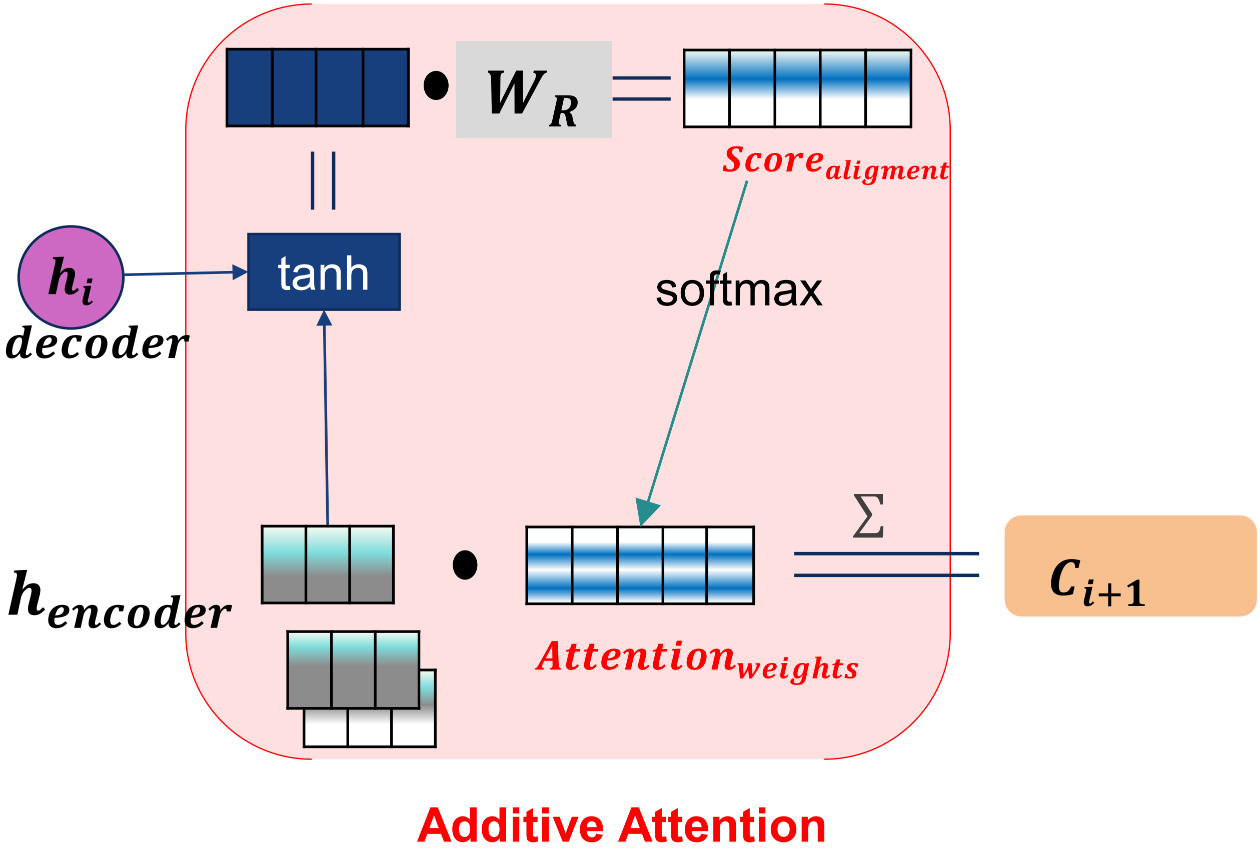


Attention based Sequence to Sequence



Attention based Sequence to Sequence

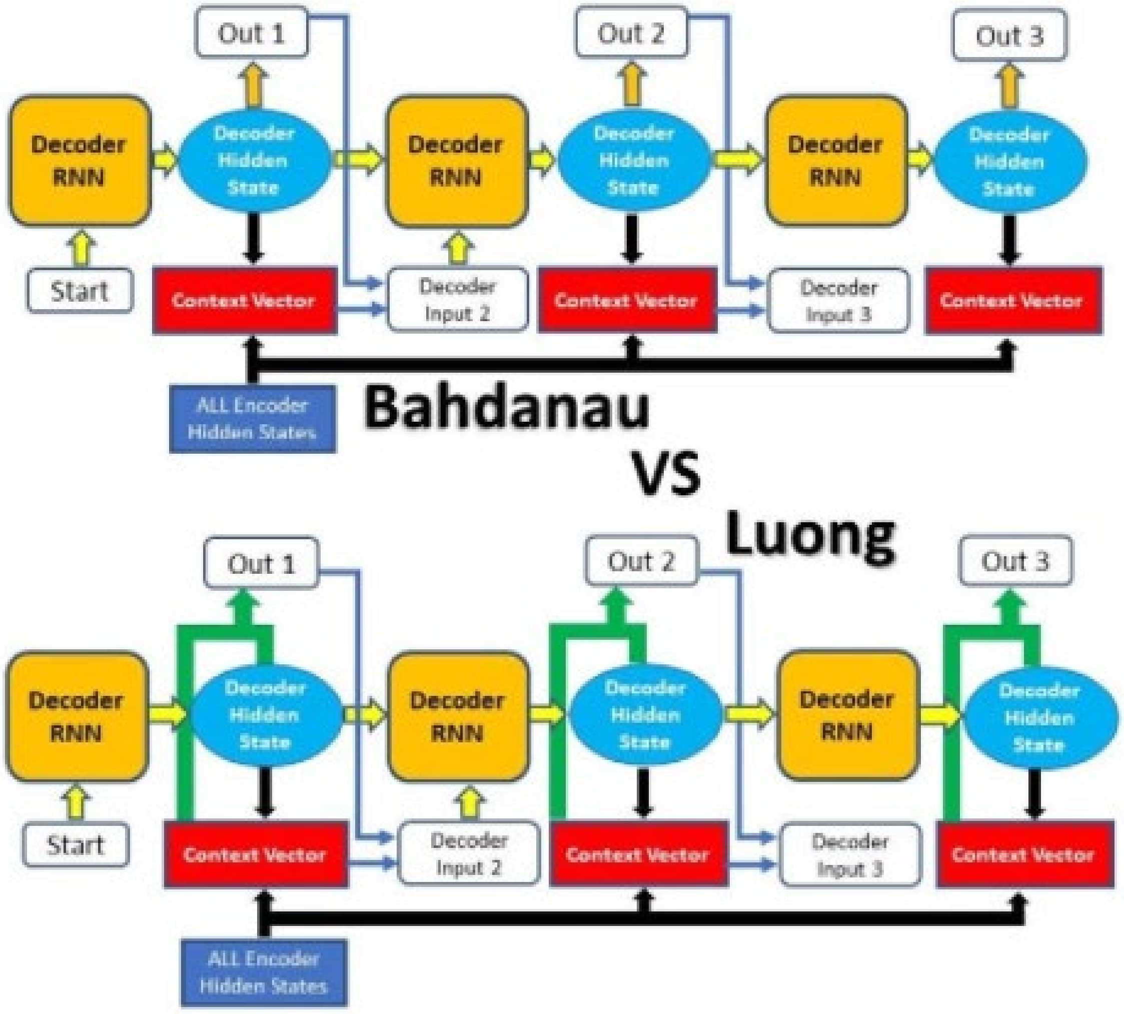




Luong architecture model

- The Bahdanau attention paper take concatenation to pass to the decoder

- Luong attention uses top hidden layer states in both of encoder and decoder. The hidden vectors are multiplied to form the 'scores' with 3 alternatives



Comparing Bahdanau Attention with Luong Attention

Global and Local Attention

- The Bahdanau use a global attention model, in which all the words in the sentence are weighted. A key issue is the length of the 'sentence' – or the attention span.

- Problem:
  - Using a global input means that as the input size increases, the matrix size also increases and also increasing computation.

- The solution is a local attention model that focuses on a specific area only. Details will not discussed here.

BE TRANSFORMED ~

Transformer

- **Attention is all you need!**

- The **Transformer in NLP** is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease.

- *Foundation of the **BERT** and **OpenAI GPT-2/3** algorithms*

Transformer Applications

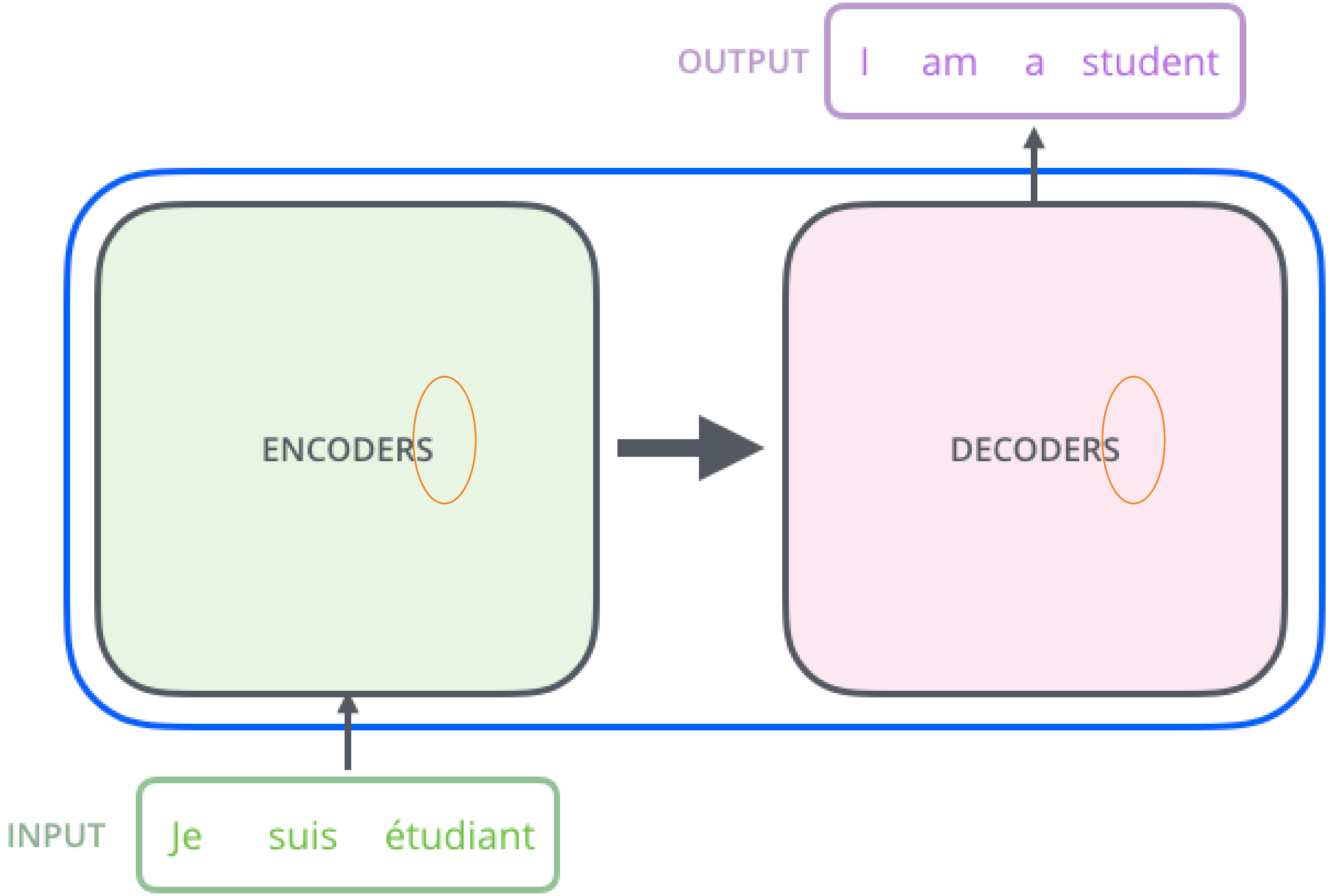
- speech recognition
- biological sequence analysis
- machine translation
- abstractive summarization
- natural language generation

Transformer

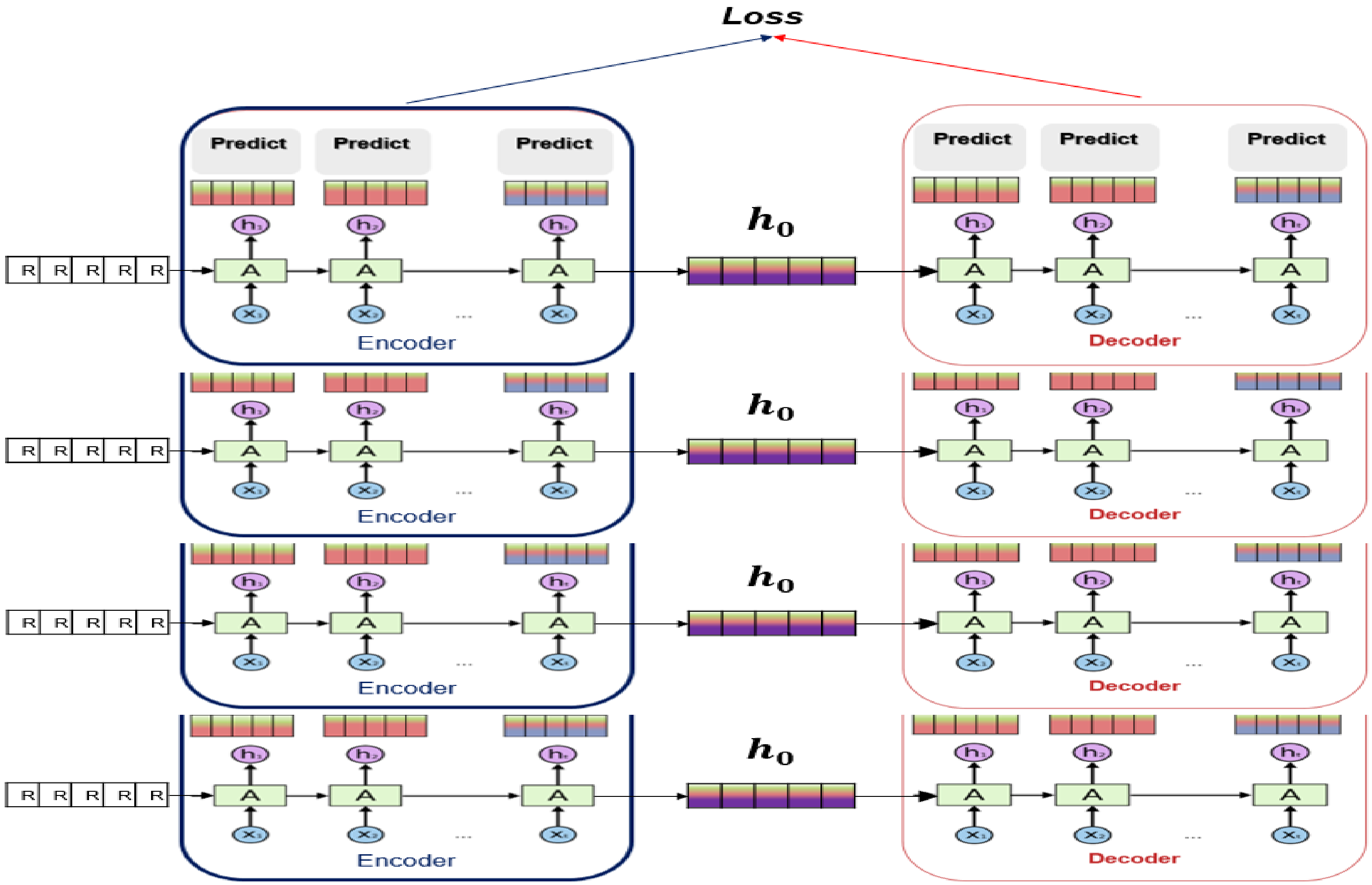
- What it does -> in a nutshell. Still a seq2seq model

- In the original paper, the transformer is used for machine translation, translating from French to English.

Transformer architecture

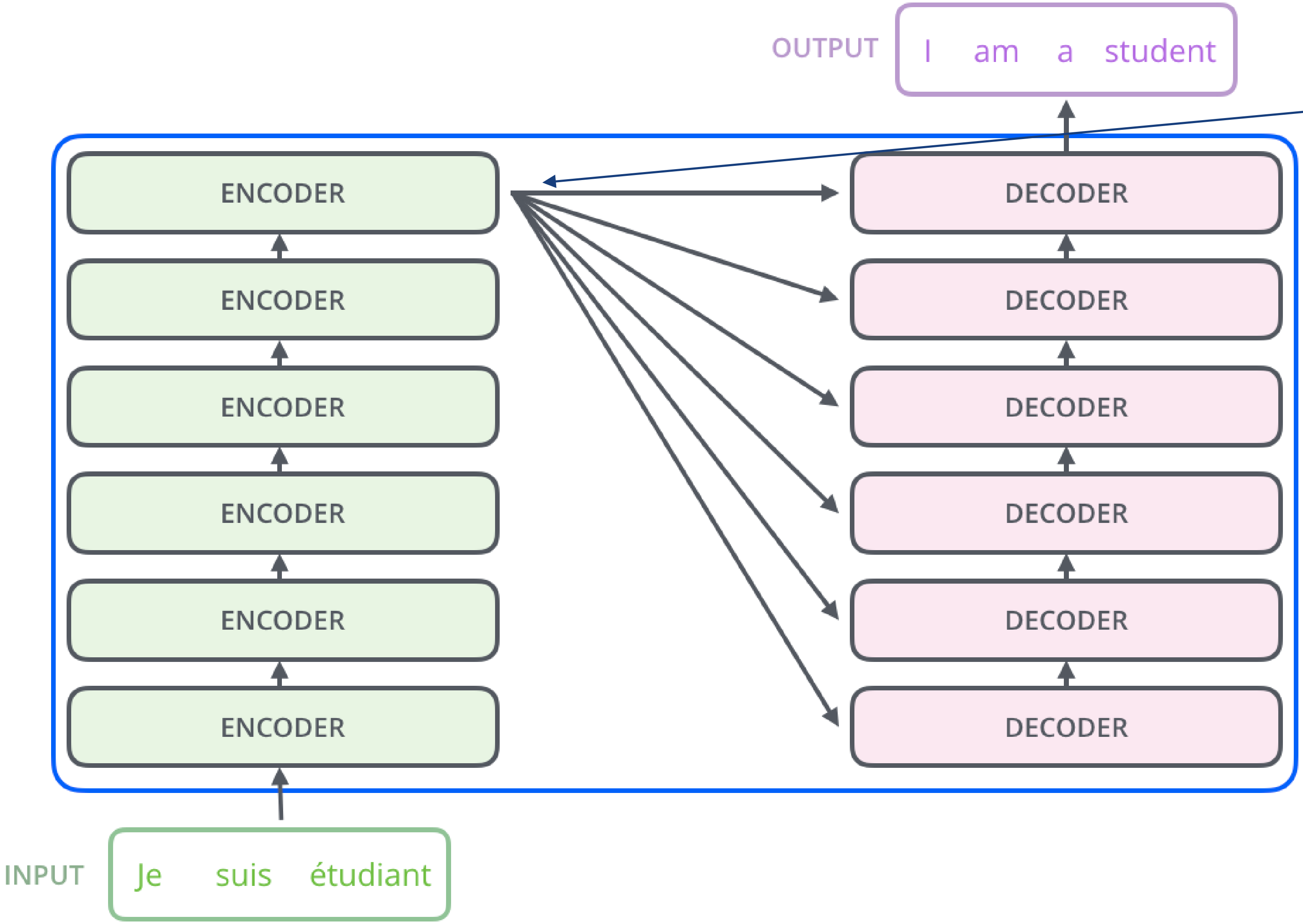


Stackable Encoders and Decoders

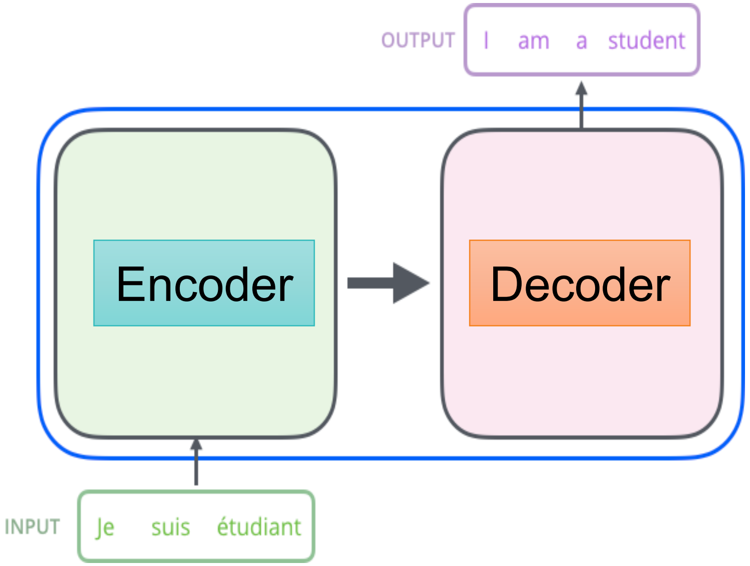


- Similar idea of stacking multiple layers of En/Decoders
- The Encoder and Decoder box will not be RNNs but Self-Attentioned layers

Breaking down the Transformer architecture



Last output of encoder feed into all the decoders.



Looking into the Transformer architecture

Encoder

One Single Encoder Box has 2 layers –

multi-head attention layer

feed forward layer

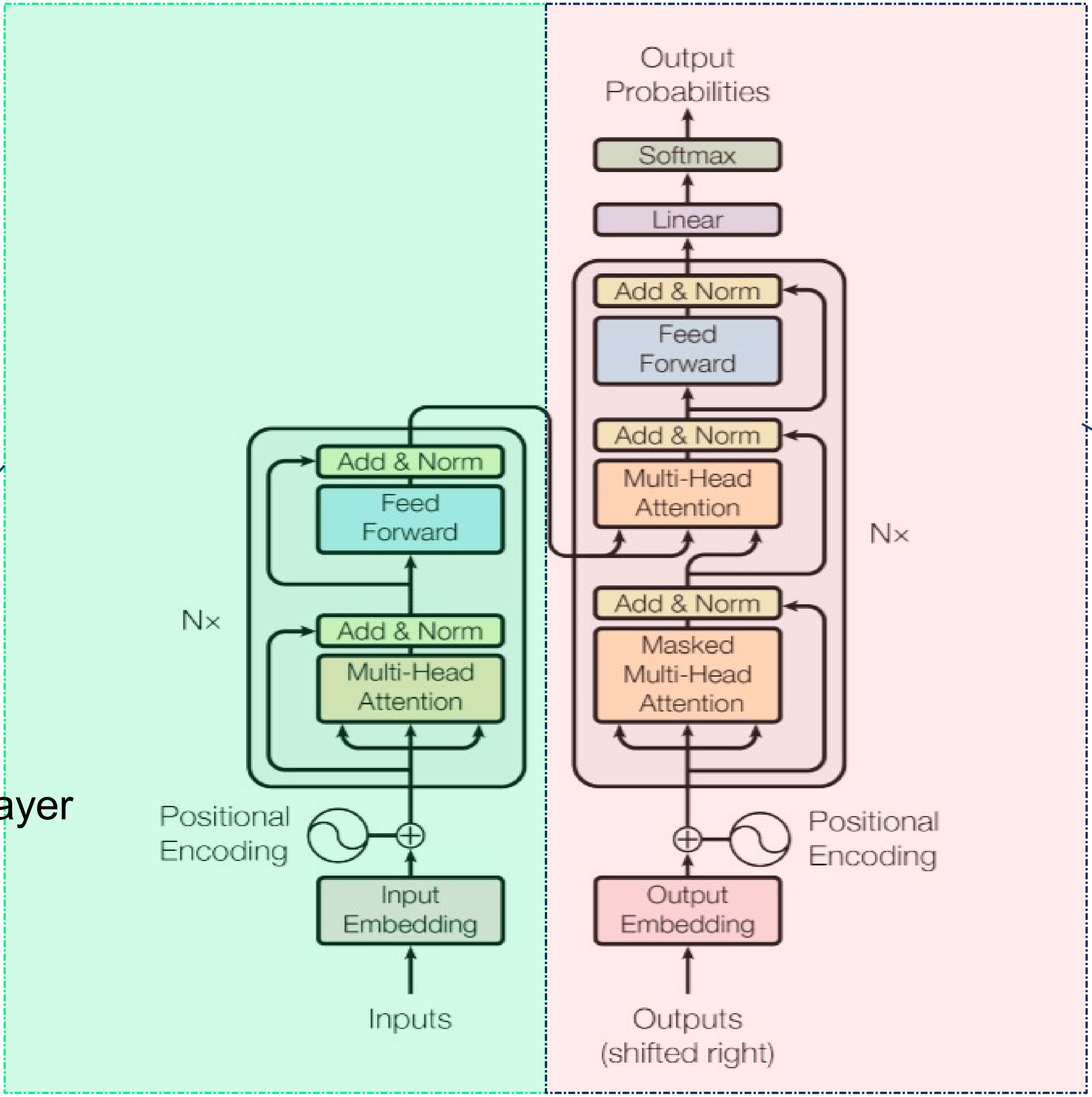
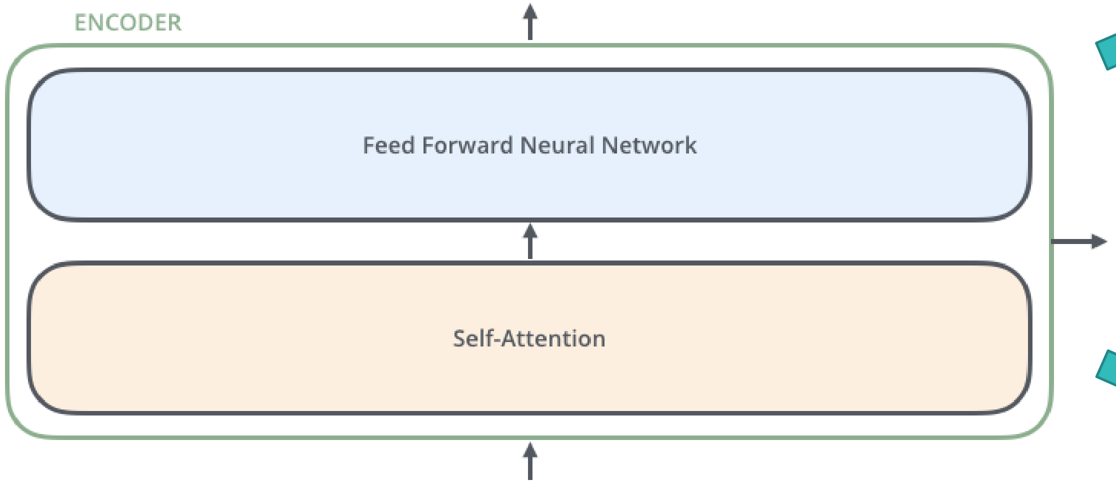


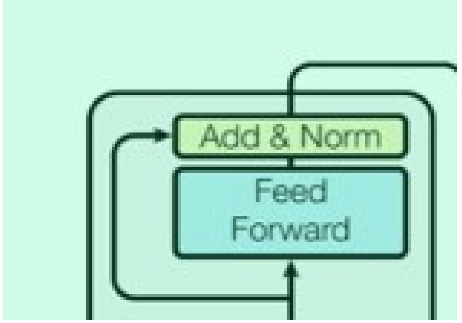
Figure 1: The Transformer - model architecture.

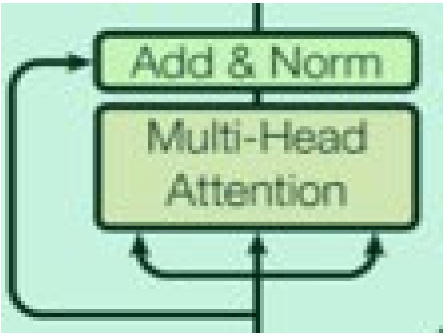
Decoder

One Single Decoder Box has 2 attention layers + feed forward layer

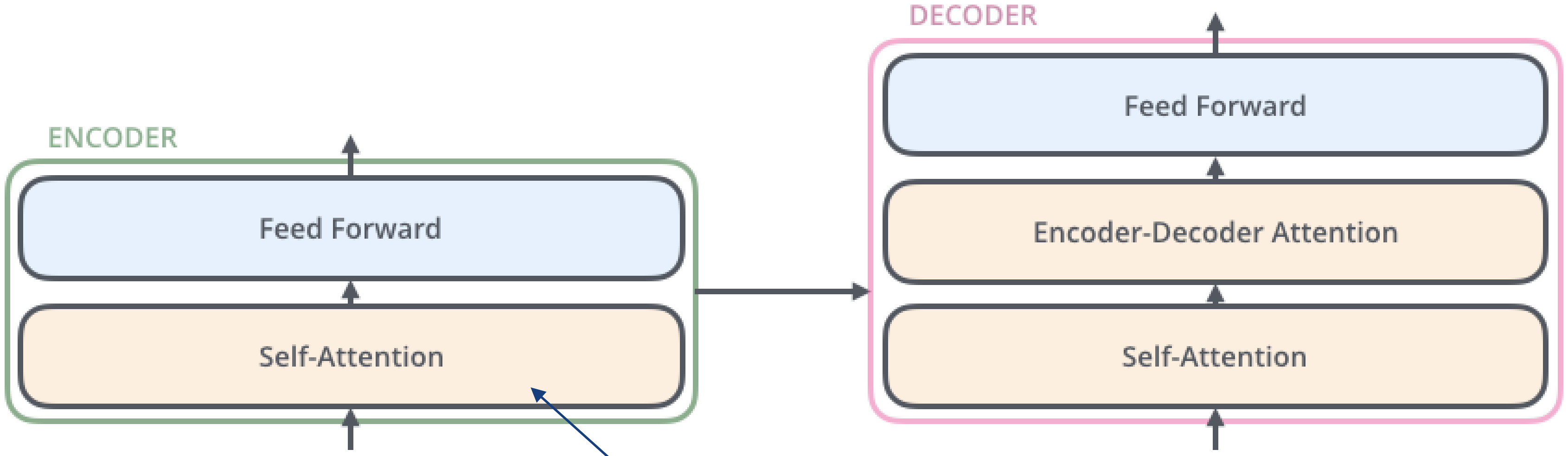
Inside the encoder







Inside the decoder



Self attention – novel feature of transformer

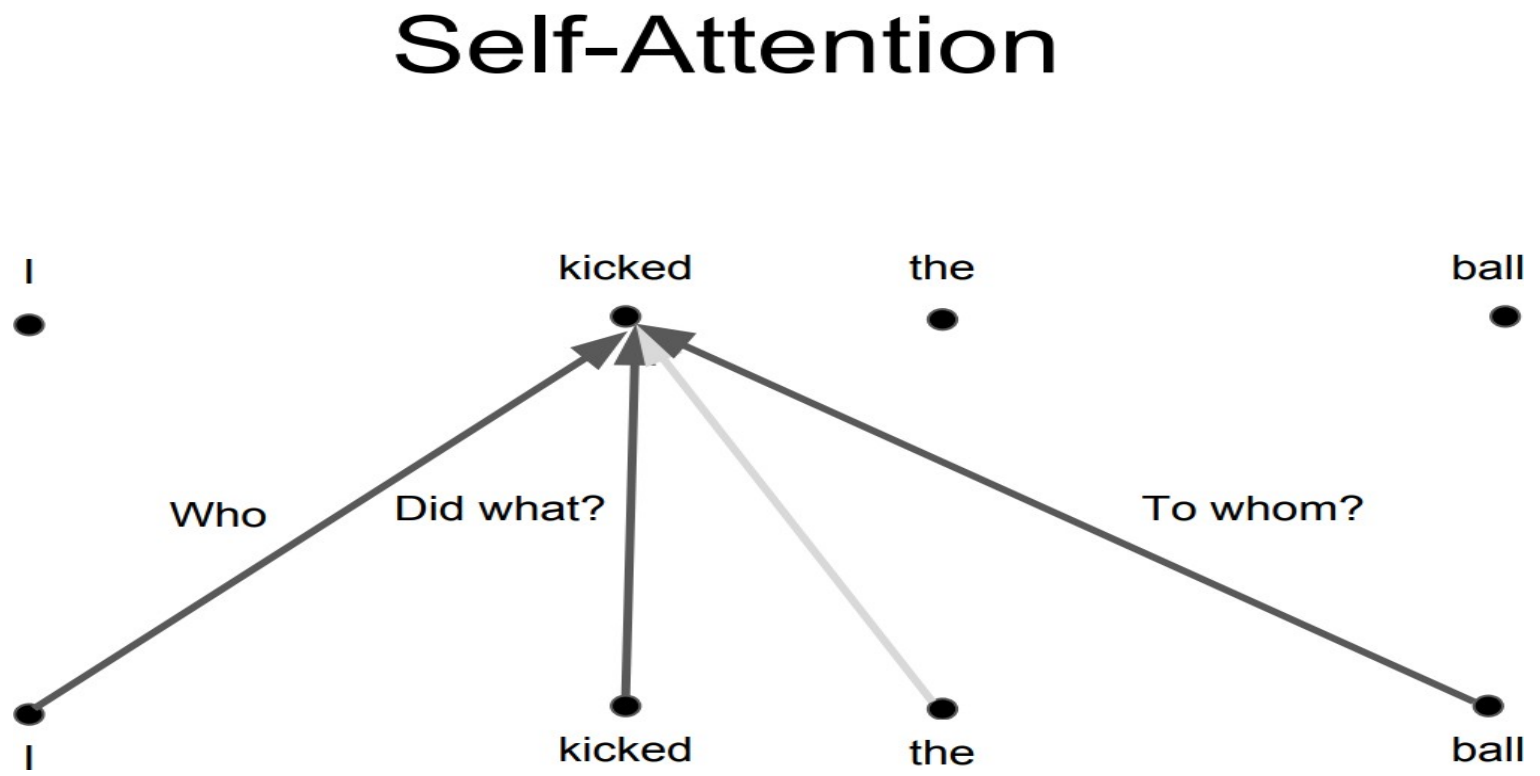
What is Self Attention?

- Original paper:
  - Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

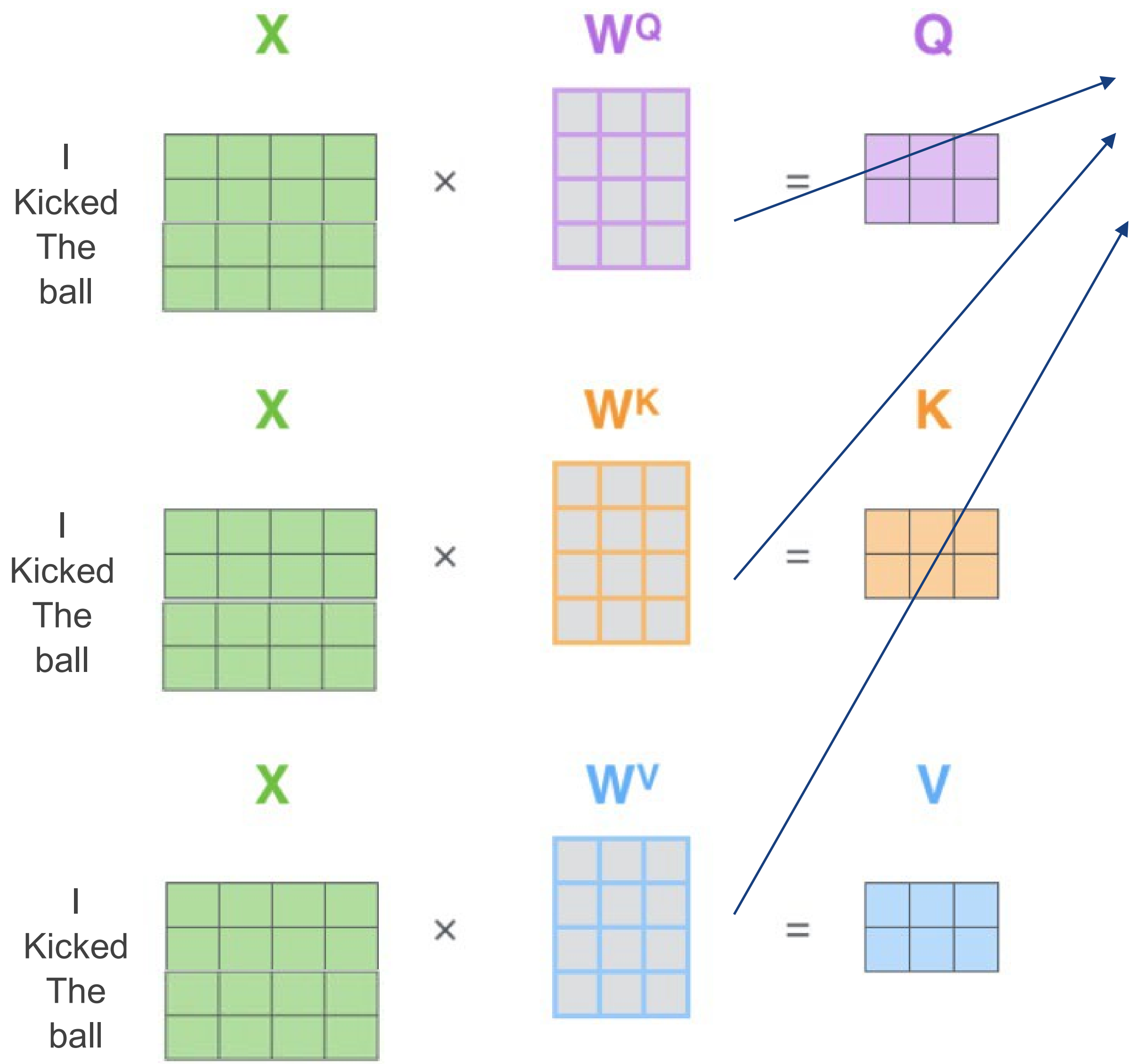
Transformers use self-attention instead of RNNs or CNNs

Self-Attention

- For the words : “I kicked the ball”, we want to know how the word ‘kicked’ relates to the different words in the sentence.

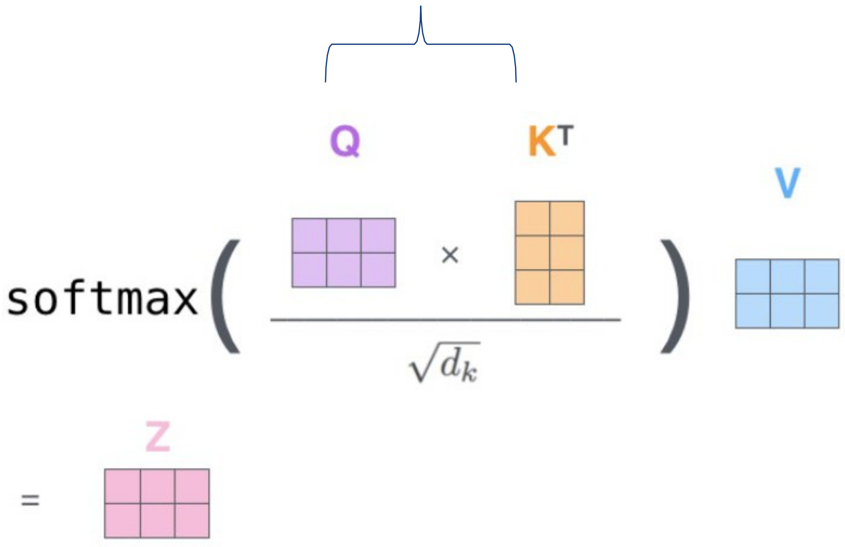


Training Self attention scores

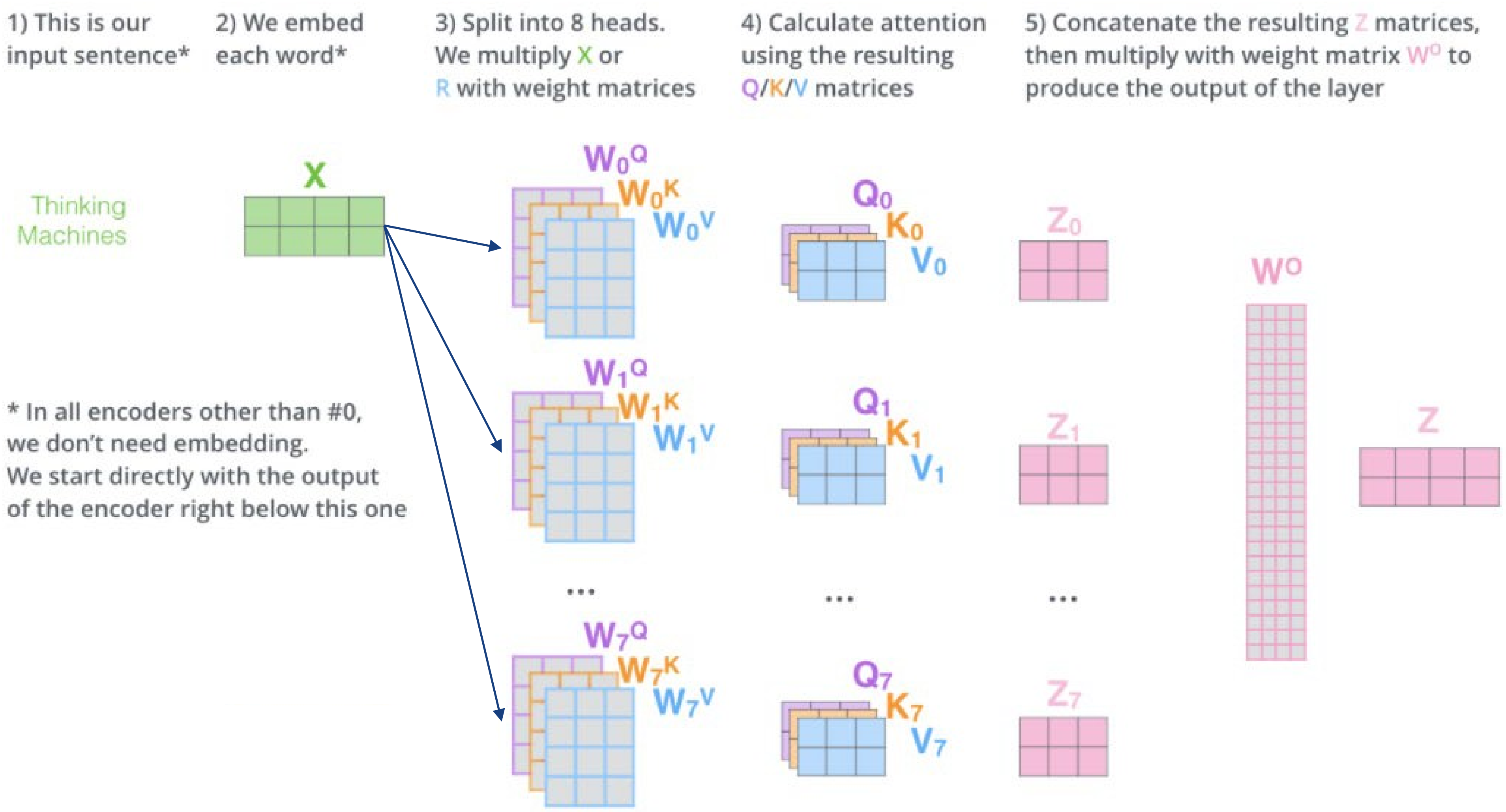


Parameter matrices to be learned

Computation summary



Summary of Attention

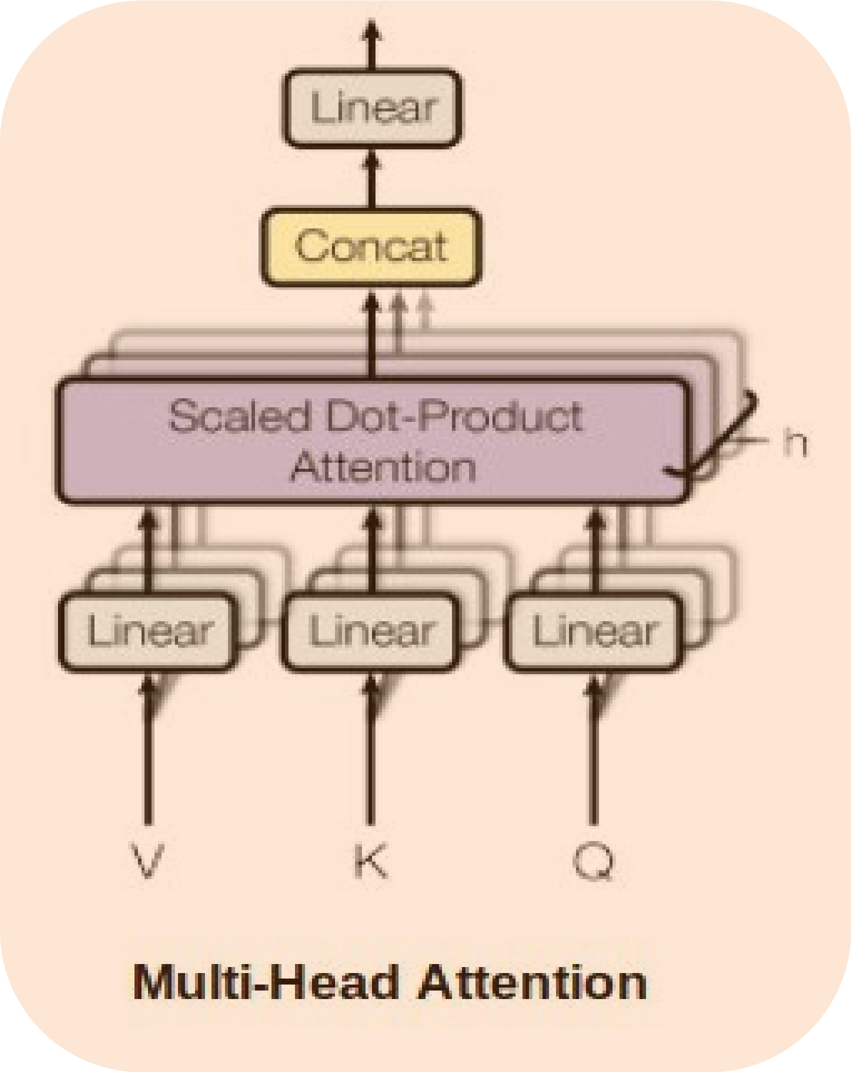


The beast with many heads

- The original paper has 8 attention heads, with each set of encoder/decoder. Each of the attention heads uses a slice of the original dataset (in this case 1/8th of the dataset) for training

- It gives the attention layer we have not only one, but multiple sets of Query/Key/Value weight matrices. The use of this multi-heads is supposed to provide multiple “representation subspaces” (or contexts) with multi-headed attention.

Multi-headed attention

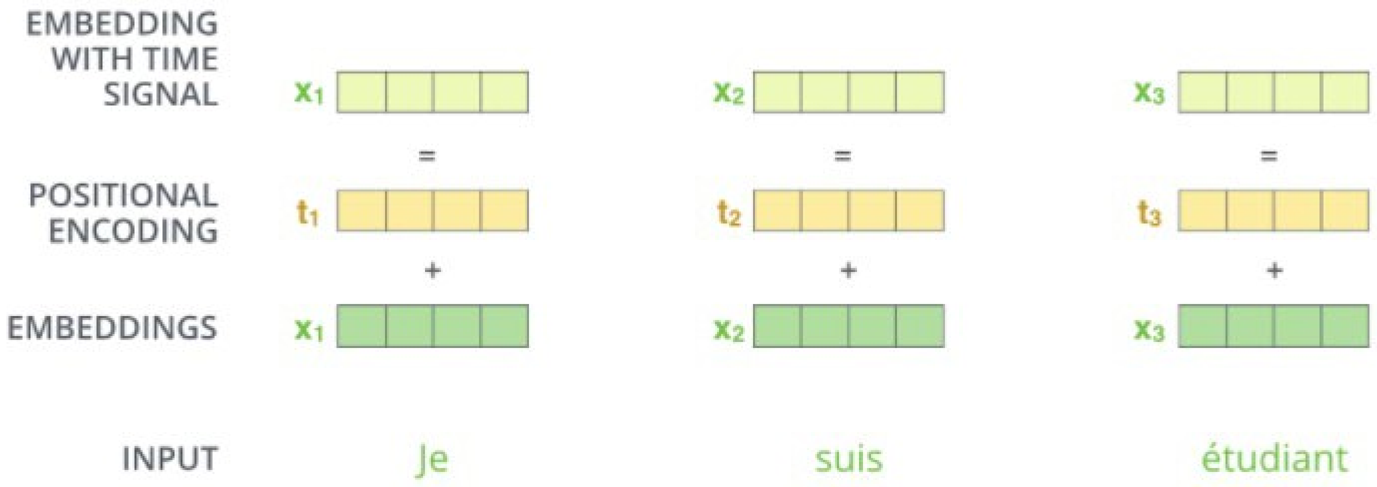


- Attention is computed multiple times independently and in parallel – 'multi-headed attention.'

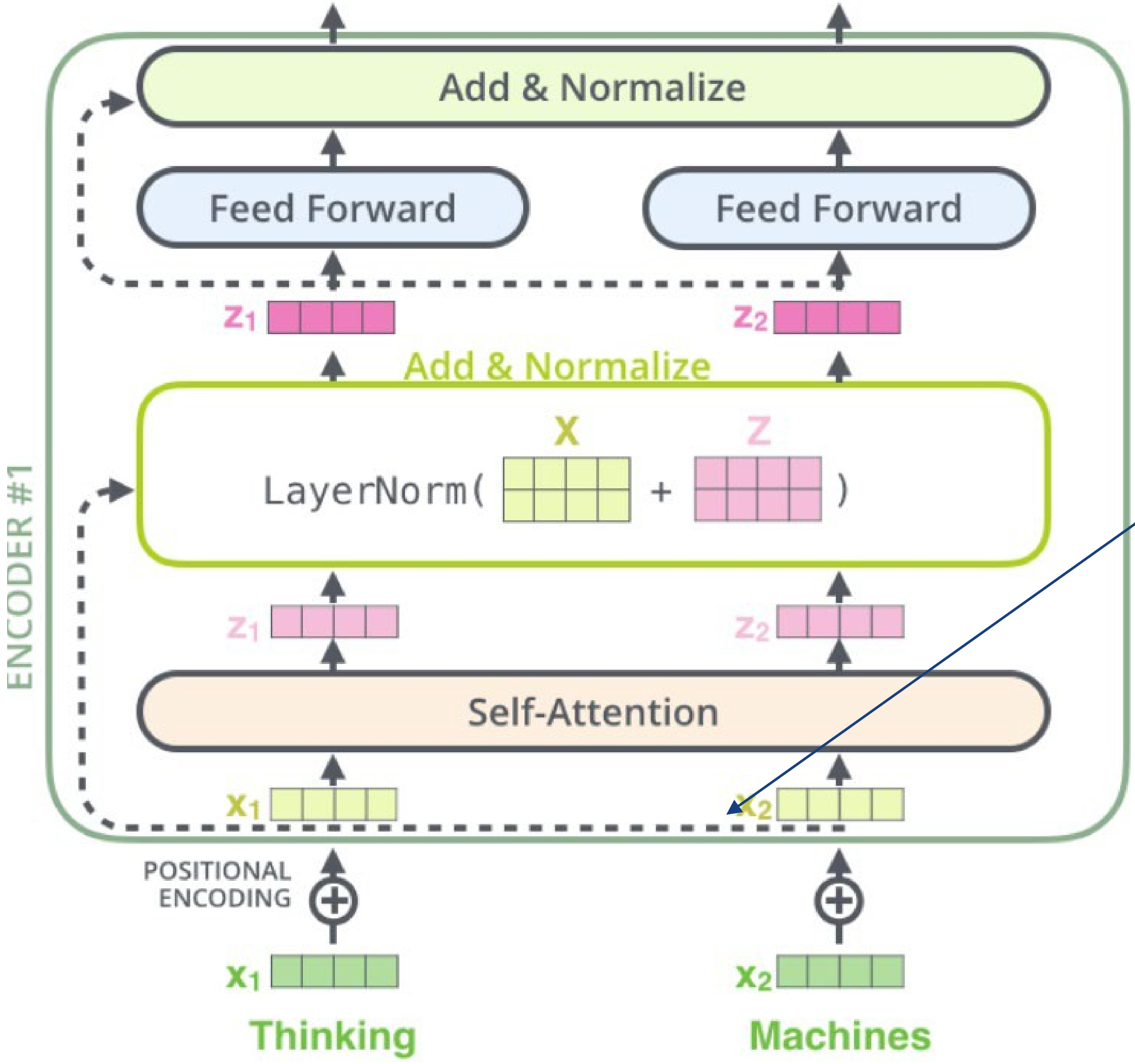
  These attentions are concatenated by a Wo weighting matrix.

Positional encoding

- The order of the words also matters, thence each word will have a vector for positional encoding that sheds details on where the word lies in the sentence. This results in a new vector – 'embedding with time signal'.

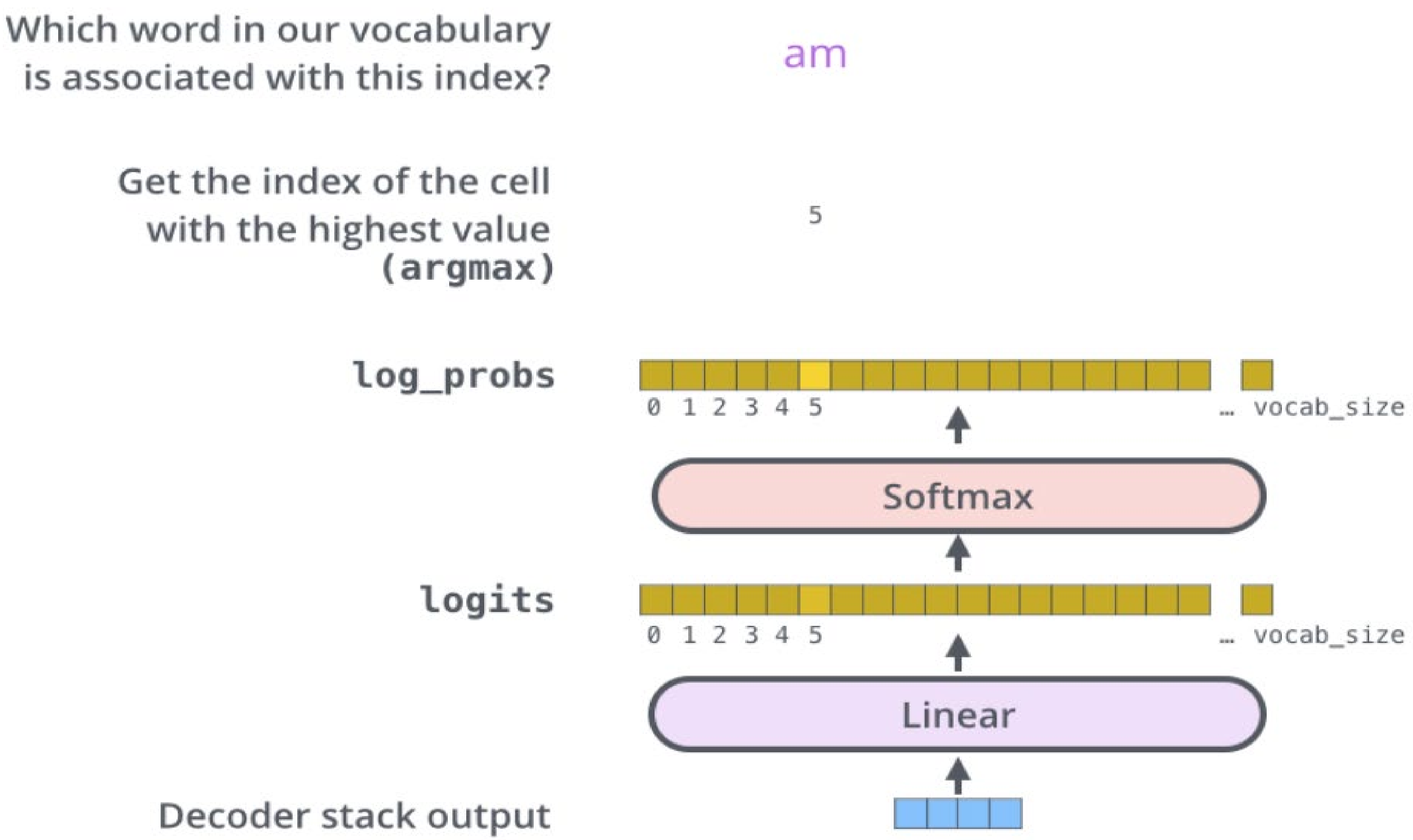


Residuals summing



There is also some residuals that directly 'by-pass' the attention layer.

Decoder stage II (masking)



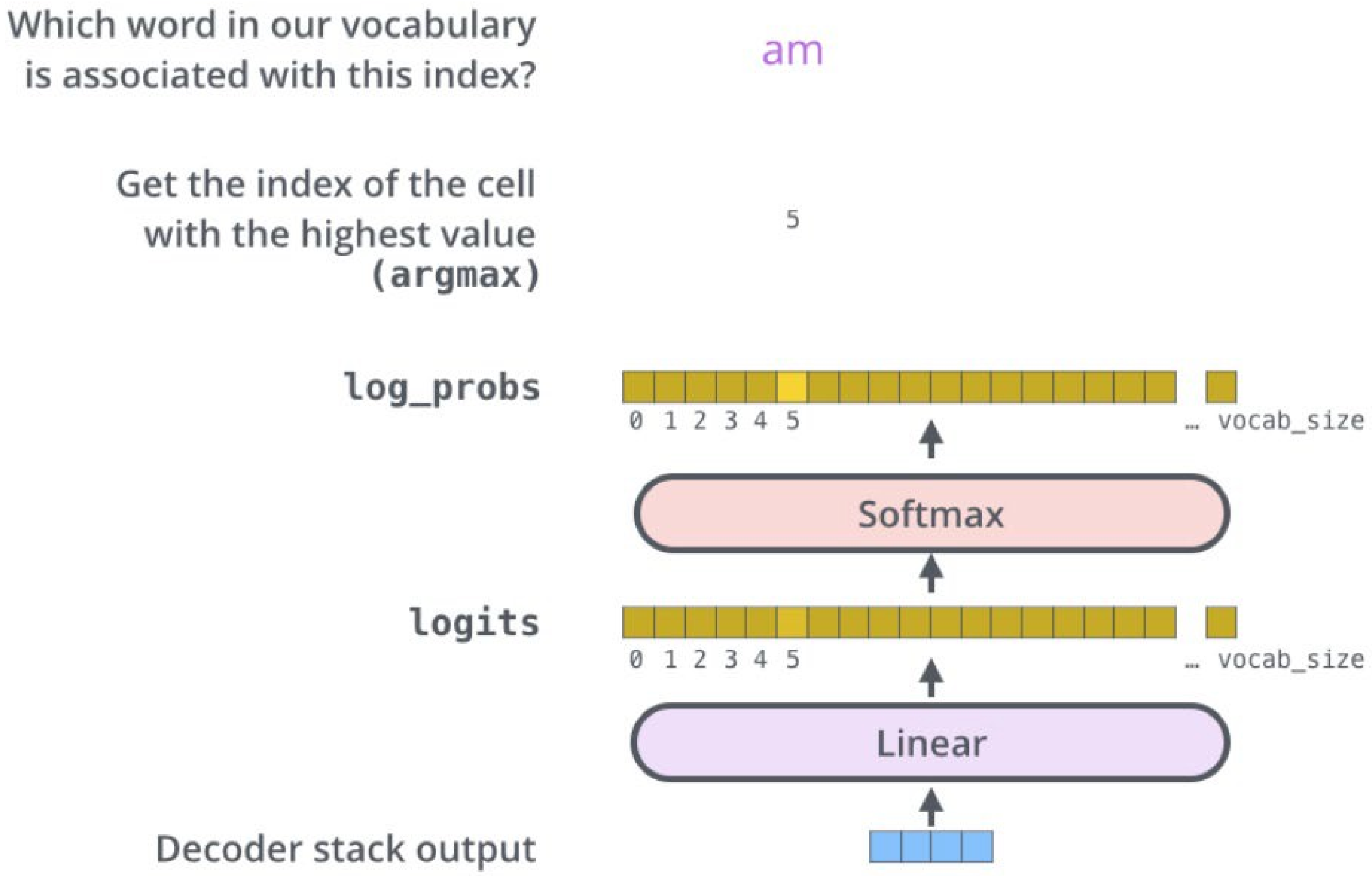
This figure starts from the bottom with the vector produced as the output of the decoder stack. It is then turned into an output word.

Each of the words goes through the decoder and decoded in the process.

In this decoding stage, each word is only affected by the words before it. The future positions are 'masked' by giving them an –inf weight.

Final Layer and Softmax Layer

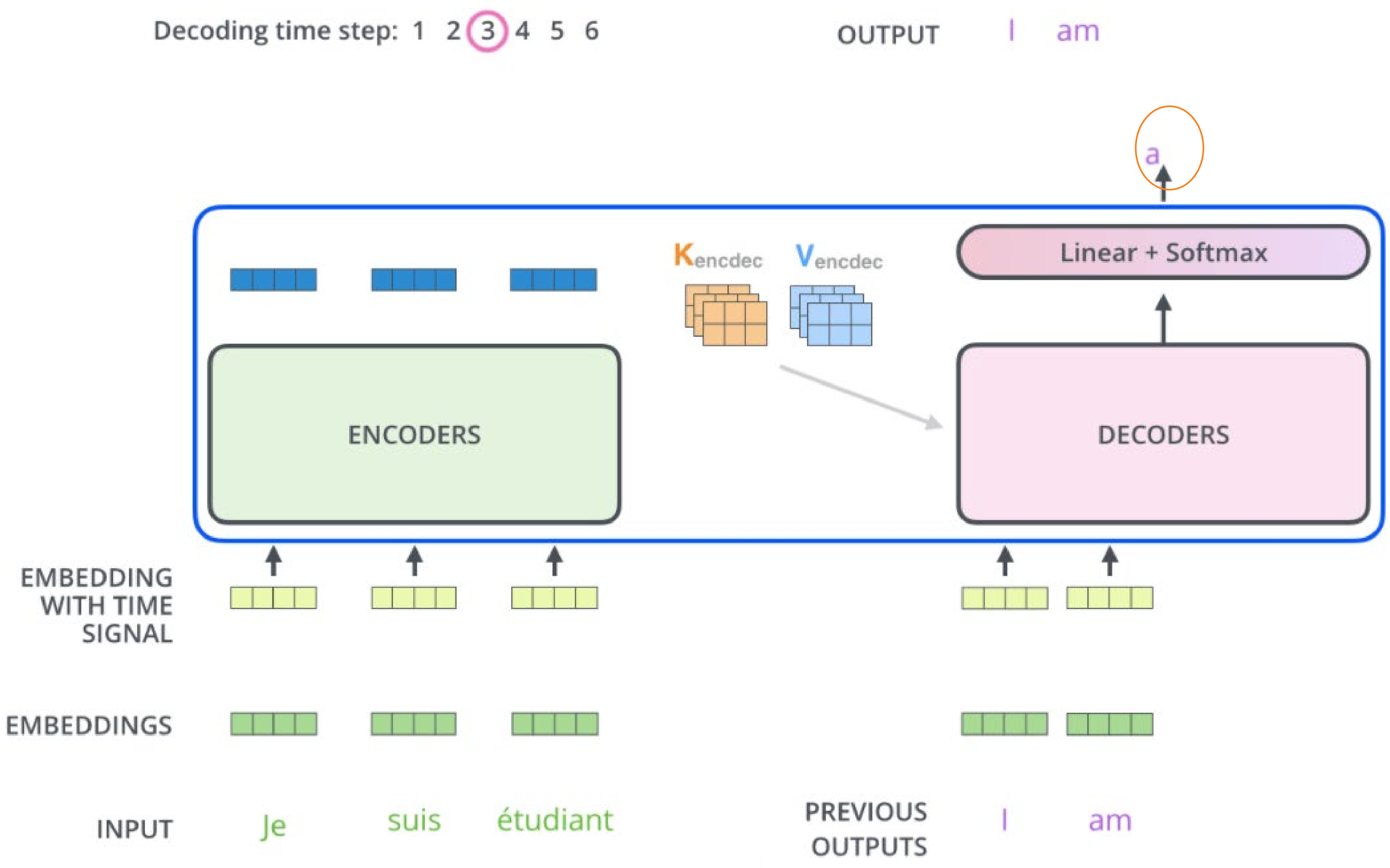
- There is thence a vector for each different word that is output from the decoder, which are turned into a word by the final Linear and Softmax layer.



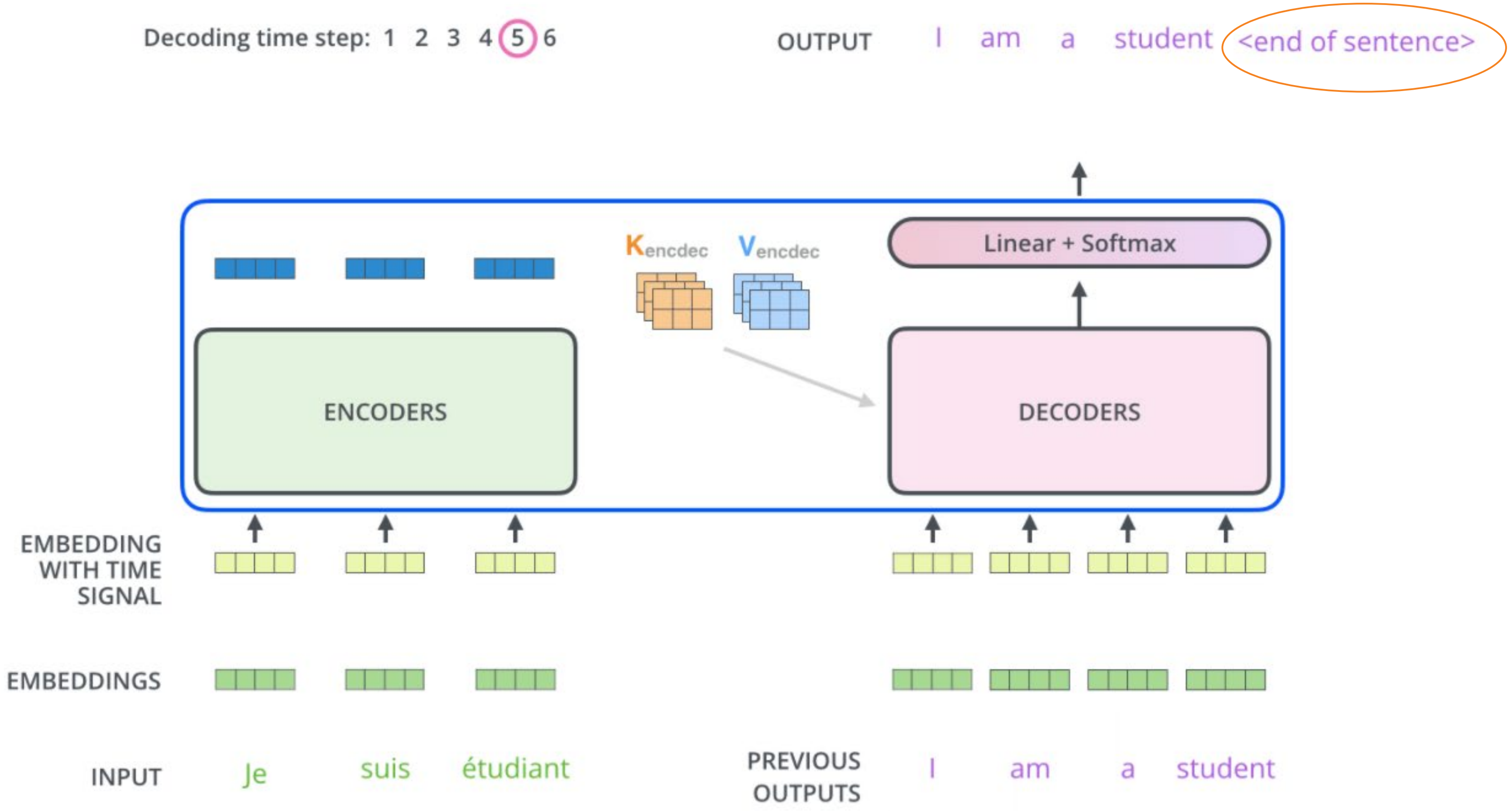
- The Linear layer is a FFN that projects the vector produced by the stack of decoders, into a much larger vector called a logits vector.
- The logits vectors consists a vocabulary of words.
- The word with the highest probabilities is then chosen as the translated word.

This figure starts from the bottom with the vector produced as the output of the decoder stack. It is then turned into an output word.

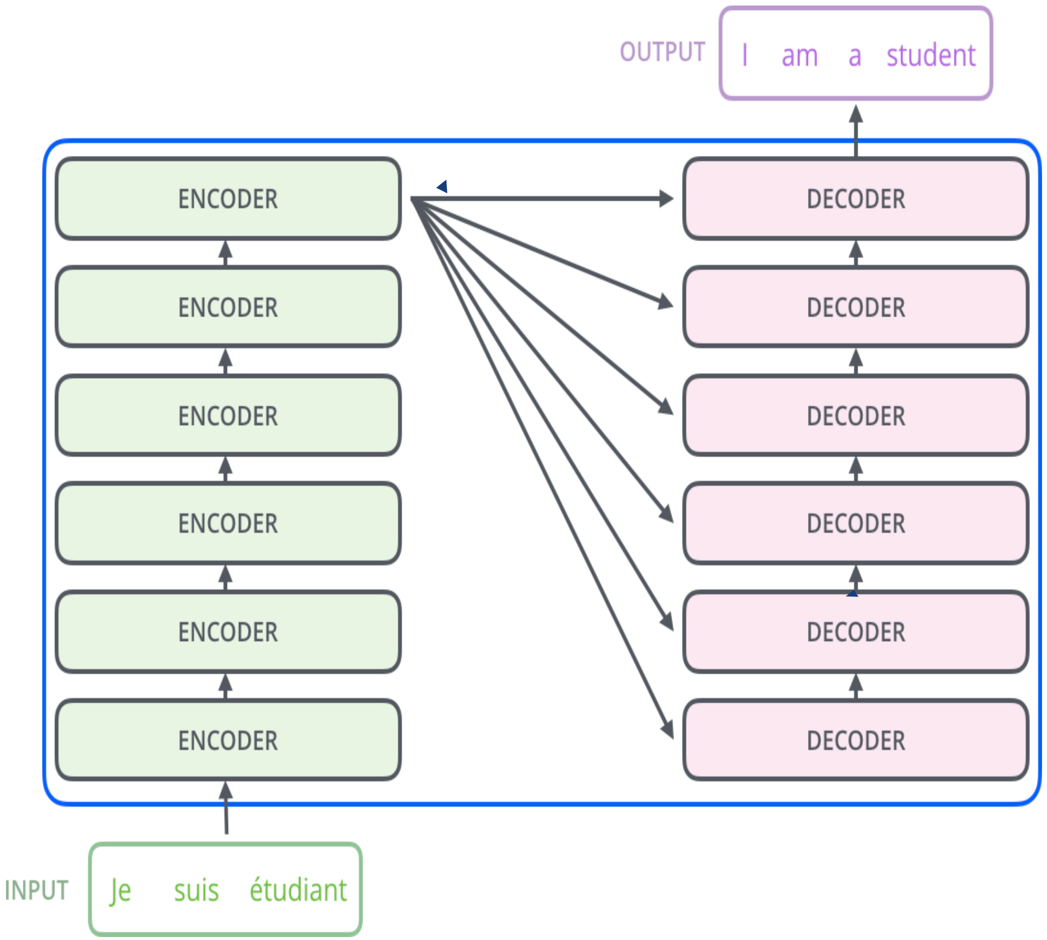
Decoding Progress



Decoding Progress



Fighting the direction of research



paying a complexity tax ?

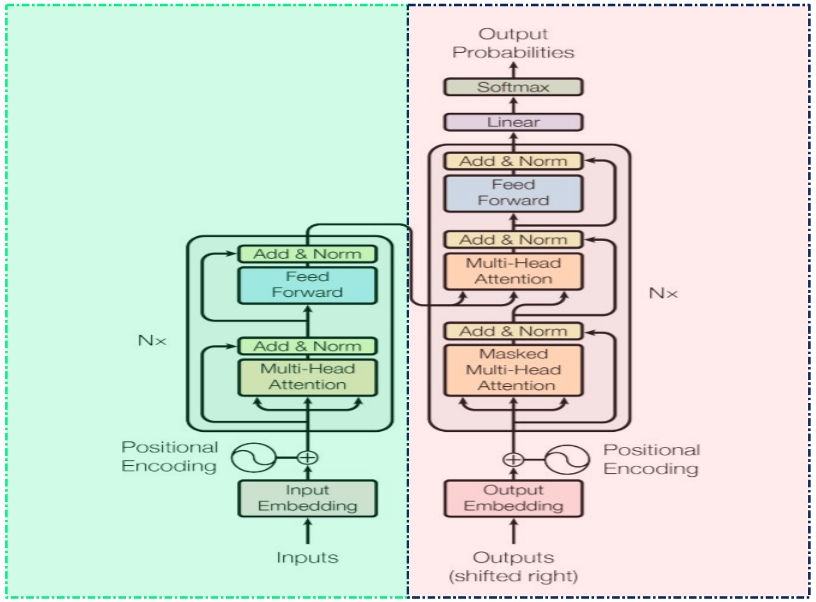
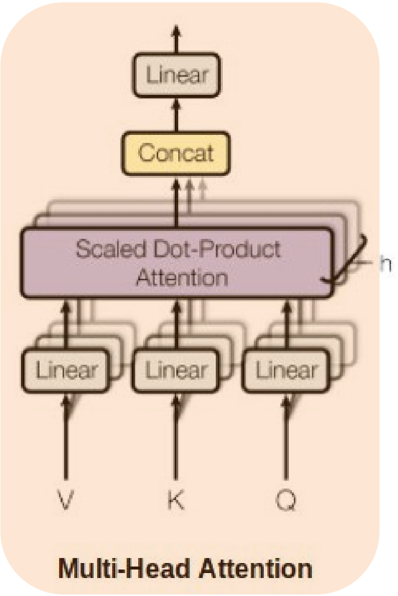


Figure 1: The Transformer - model architecture.



8Head Attention + Transformers
8 GPU + 3.5 day
Google Research, 2017

Fighting the direction of research

Model	Heads	Valid	Test	Params
Large RHN (Zilly et al., 2016)	0	–	1.27	46M
3 layer AWD-LSTM (Merity et al., 2018b)	0	–	1.232	47M
T12 (12 layer) (Al-Rfou et al., 2019)	24	–	1.11	44M
LSTM (Melis et al., 2019)	0	1.182	1.195	48M
Mogrifier LSTM (Melis et al., 2019)	0	1.135	1.146	48M
4 layer SHA-LSTM ($h = 1024$, no attention head)	0	1.312	1.330	51M
4 layer SHA-LSTM ($h = 1024$, single attention head)	1	1.100	1.076	52M
4 layer SHA-LSTM ($h = 1024$, attention head per layer)	4	1.096	1.068	54M
T64 (64 layer) (Al-Rfou et al., 2019)	128	–	1.06	235M
Transformer-XL (12 layer) (Dai et al., 2019)	160	–	1.06	41M
Transformer-XL (18 layer) (Dai et al., 2019)	160	–	1.03	88M
Adaptive Transformer (12 layer) (Sukhbaatar et al., 2019)	96	1.04	1.02	39M
Sparse Transformer (30 layer) (Child et al., 2019)	240	–	0.99	95M

Single Head Attention + LSTMs
Single GPU + 1 day

8Head Attention + Transformers
8 GPU + 3.5 day

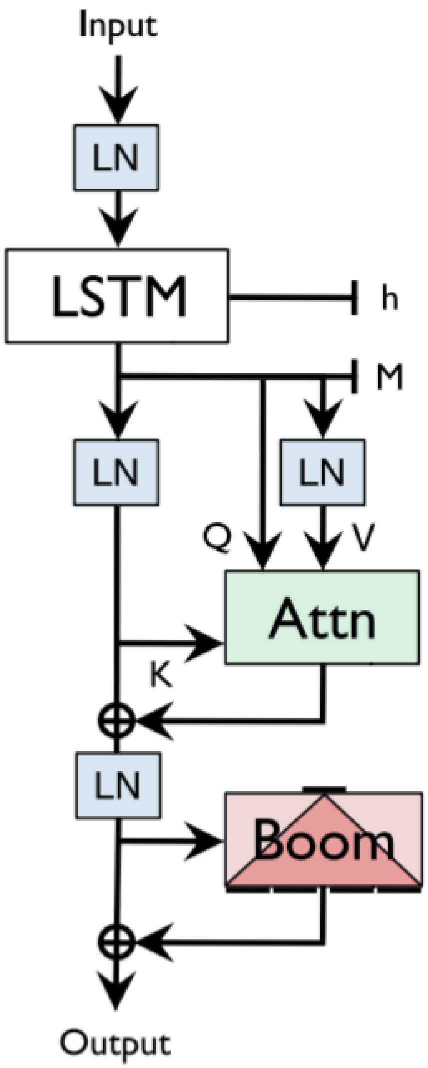
Stephen Merity, 2019

Google Research, 2017

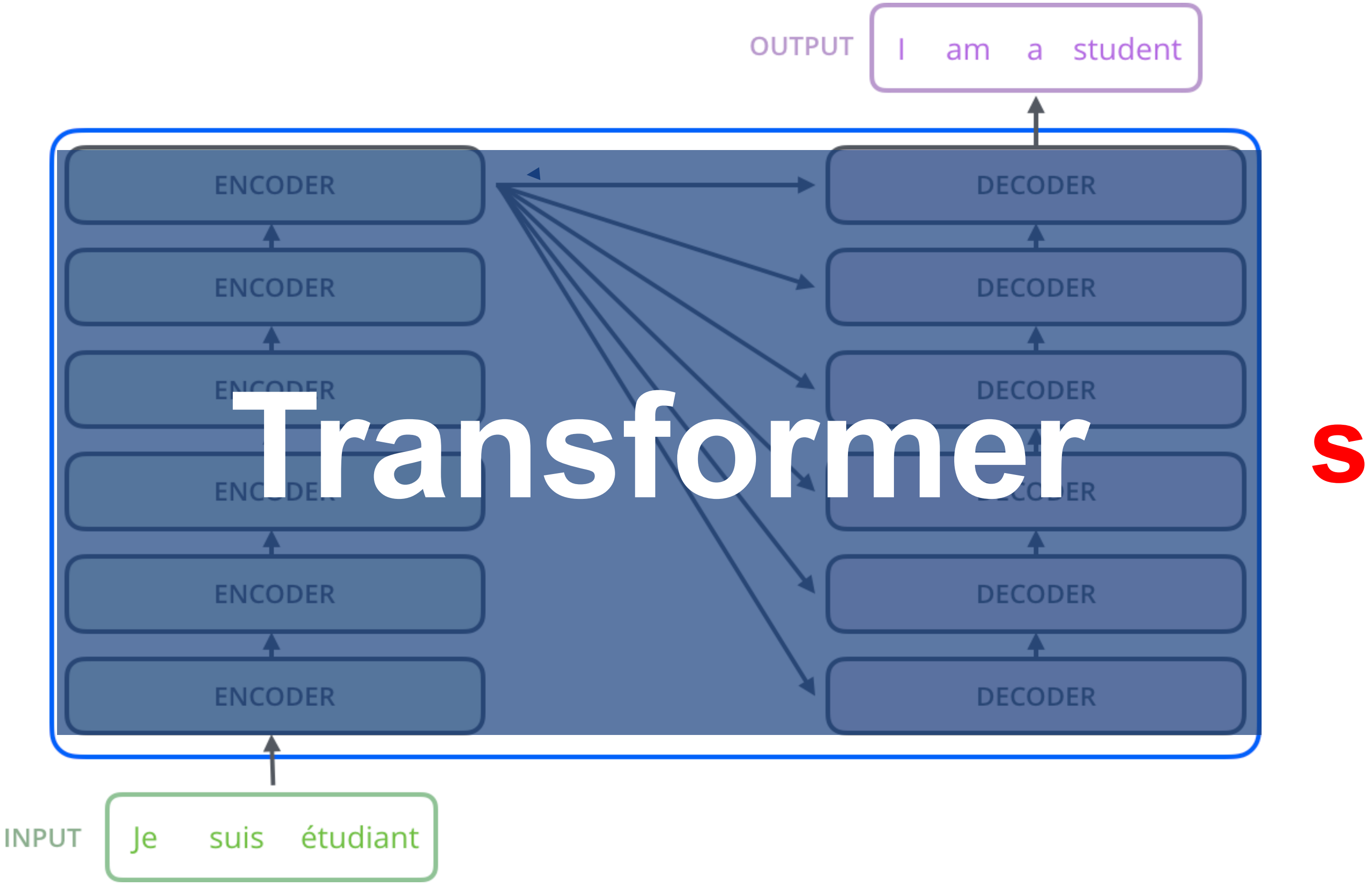
Fighting the direction of research

"Perhaps we were too quick to throw away the past era of models simply due to a new flurry of progress."

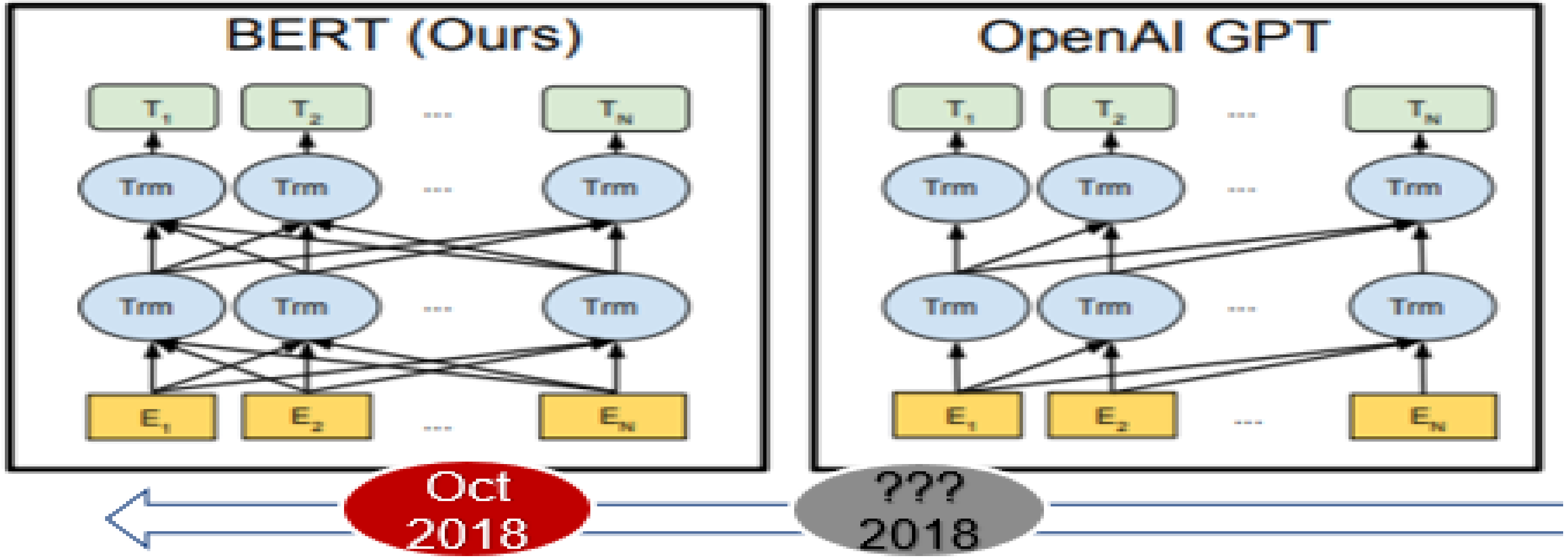
"Perhaps we're too committed to our existing stepping stones to backtrack and instead find ourselves locked to a given path."



Continue with Transformers



Continue with Transformers



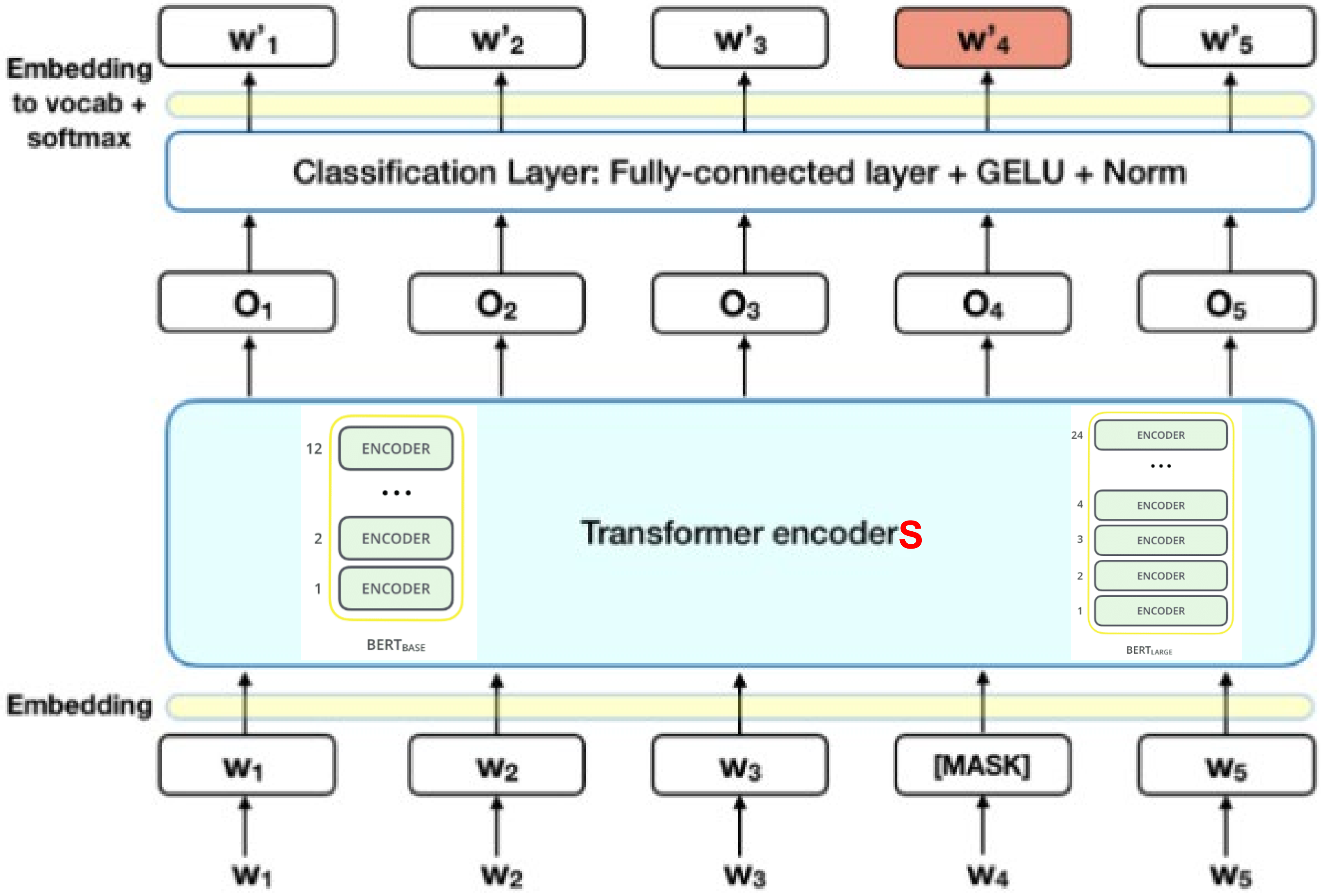
24 Transformer blocks
1024 hidden layers, and
340M parameters
16 TPU pods for
training

12 blocks
37-layer
1 month on
8 GPUs

BERT

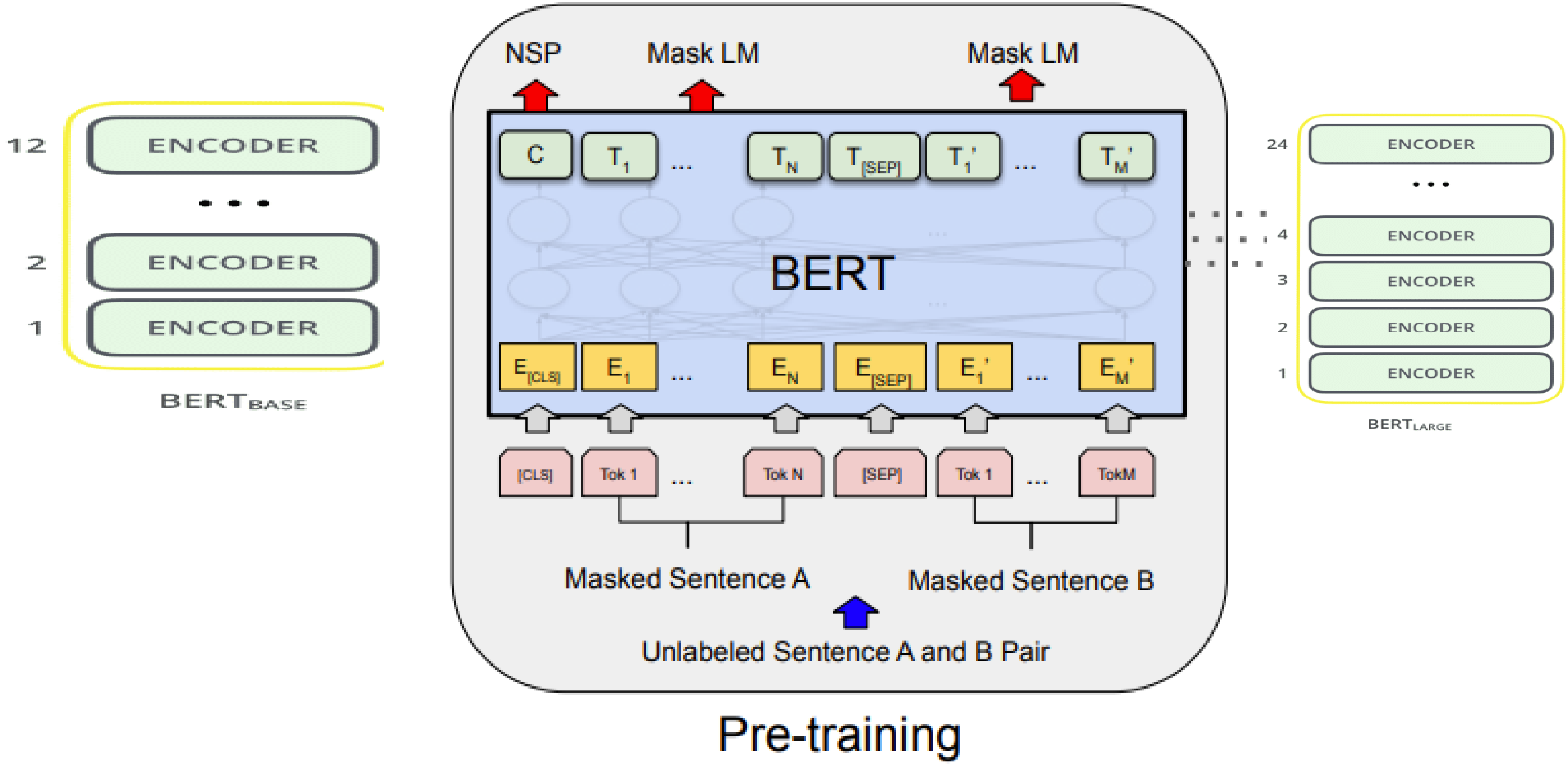
MASK LM

15% of the words in each sequence are replaced with a [MASK] token.

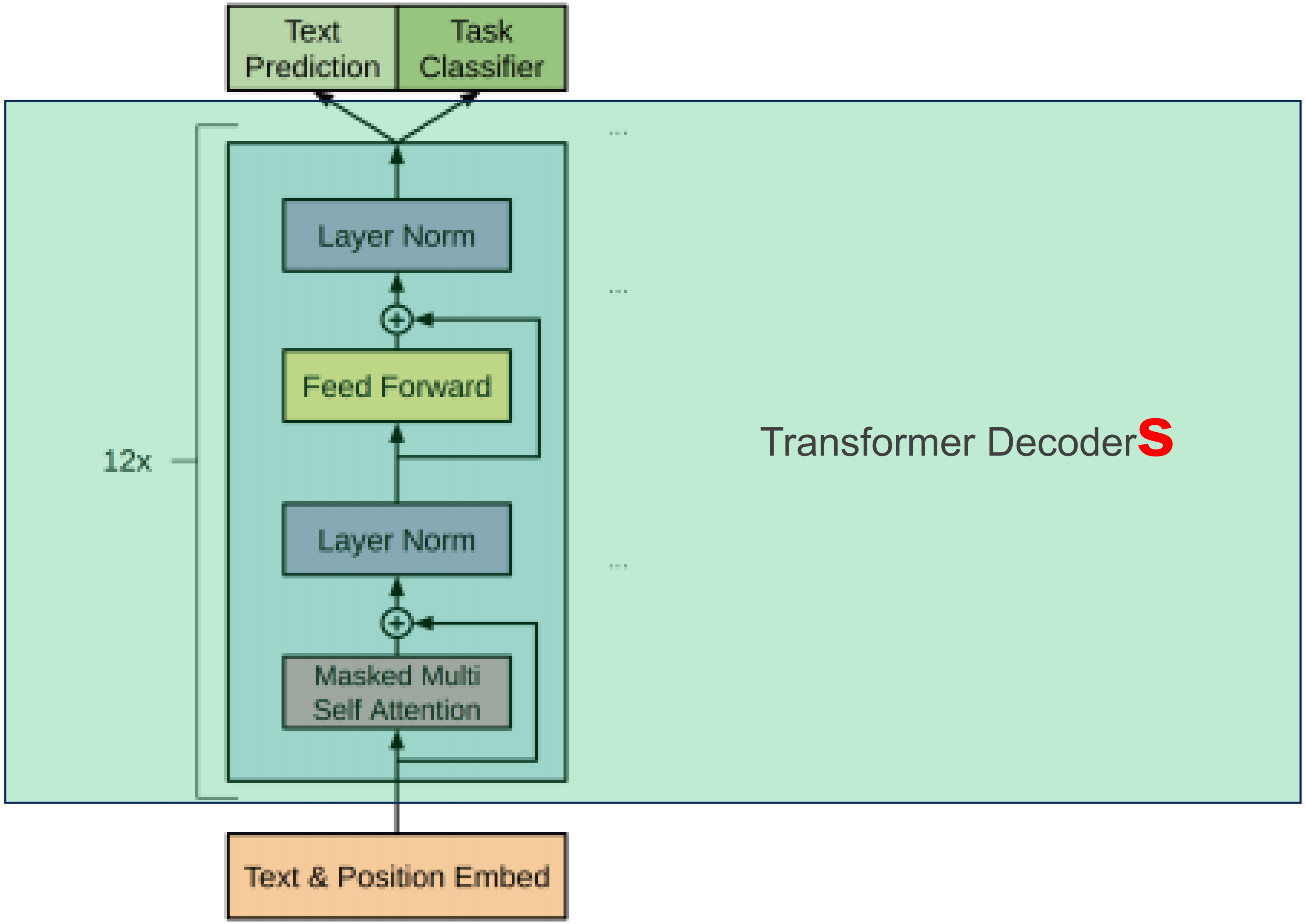


BERT

MASK LM & Next Sentence Prediction



GPT 2/3


Text Prediction
Task Classifier
Layer Norm
Feed Forward
12x
Layer Norm
Masked Multi Self Attention
Text & Position Embed
Transformer DecoderS

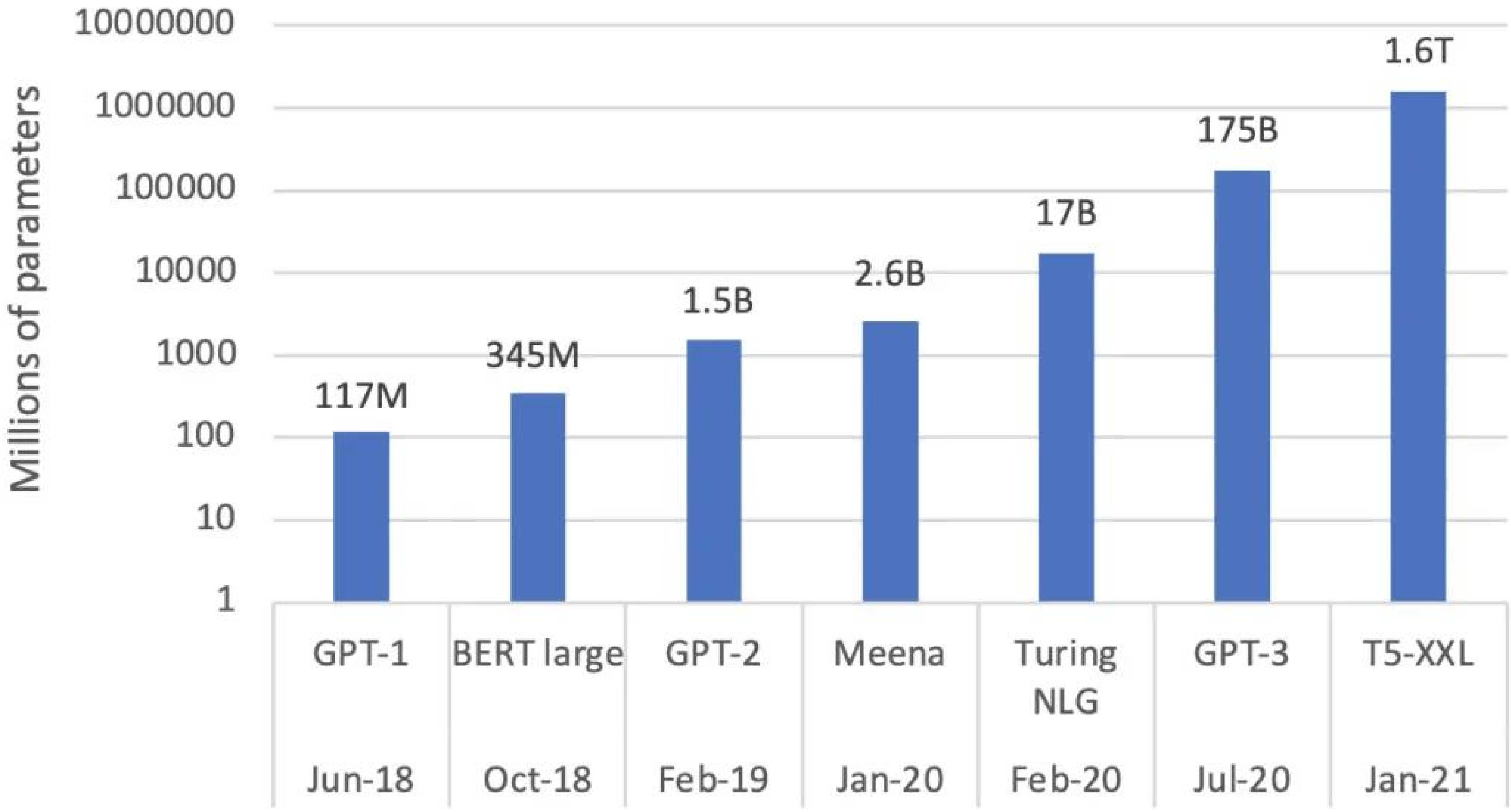
No Silver Bullet (2020)

Table 2: The results of political perspective detection with different models.

	Model	Israeli perspective recall	Palestinian perspective recall	AUC
Baselines	SVM	0.833	0.586	0.710
	Logistic Regression	0.963	0.845	0.964
	LSTM	0.966	0.879	0.966
Off-the-shelf models	BERT pre-trained fine-tuned	0.778	0.638	0.797
	Swivel news matrix factorization	0.740	0.766	0.840

https://arxiv.org/pdf/2009.07238v2.pdf

Size of Language Models Over Time



Reference

1. East to read blogs on transformer/ attention

https://towardsdatascience.com/sequence-2-sequence-model-with-attention-mechanism-9e9ca2a613a

http://jalammar.github.io/illustrated-transformer/

2. Original seminal paper on transformer/ attention

https://arxiv.org/abs/1706.03762

3. Useful article explaining positional encoding using sinusoidal function

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

4. Youtube video on transformer

https://www.youtube.com/watch?v=rBCqOTEfxvg

5. Single Headed Attention RNN: Stop Thinking With Your Head

https://arxiv.org/pdf/1911.11423.pdf

Agenda

- Day 3 Advanced DNN systems
  - Attention
  - Transformer
  - Sentence / Document representation
  - Workshop (PM)

Doc2vec Sentence Representation

- **Doc2Vec** is an extension of **Word2vec**
- The documents here can refer to paragraphs, articles or whole documents.
- **Doc2Vec** vectors represent the theme or overall meaning of a document

Applications of document embeddings

- Document comparison can be done by a similarity measure – and used to retrieve most similar document texts

- In our workshop, we use the gensim (also in Tensorflow). But genism seems to have better accuracy than TF as noted in industry.

Document Similarity Word Movers Distance

- Based on WordVectors such as GLOVE, W2V

- Allows to assess the “distance” between *two documents* in a meaningful way, even when they have no words in common.

- Uses **Euclidean distance** and a ‘transport matrix’ –T (that is trained) that determines how many of such word vectors to ‘transport/move’ from one document to another for them to be similar.

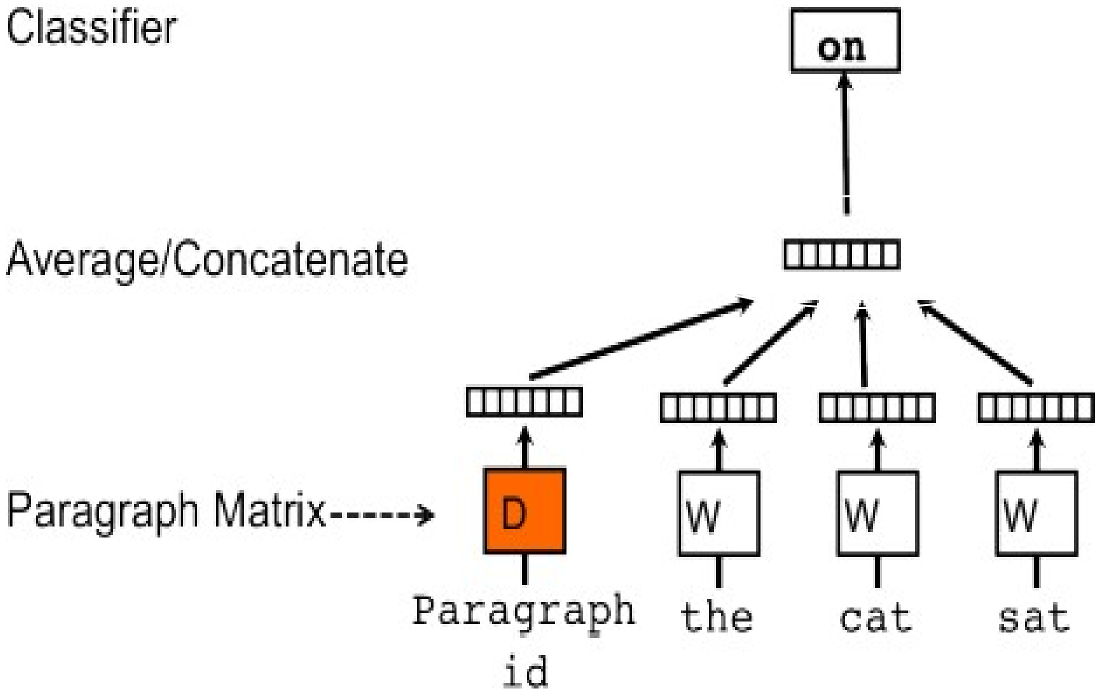
$$D(\mathbf{x}_i, \mathbf{x}_j) = \min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|_2^p, \quad \text{subject to,} \quad \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i^a, \quad \sum_{i=1}^{n} \mathbf{T}_{ij} = d_j^b \quad \forall i, j, \quad (1)$$

Paragraph To Vectors

- Two main training methods of these **Paragraph Vectors** (PV)
  - Distributed Memory Model (PV-DM)
  - Distributed Bag Of Words (PVDBOW)

- Both are **Self-supervised** methods, in that from the original paragraphs/ documents, you try to create a representative paragraph/ document vector.

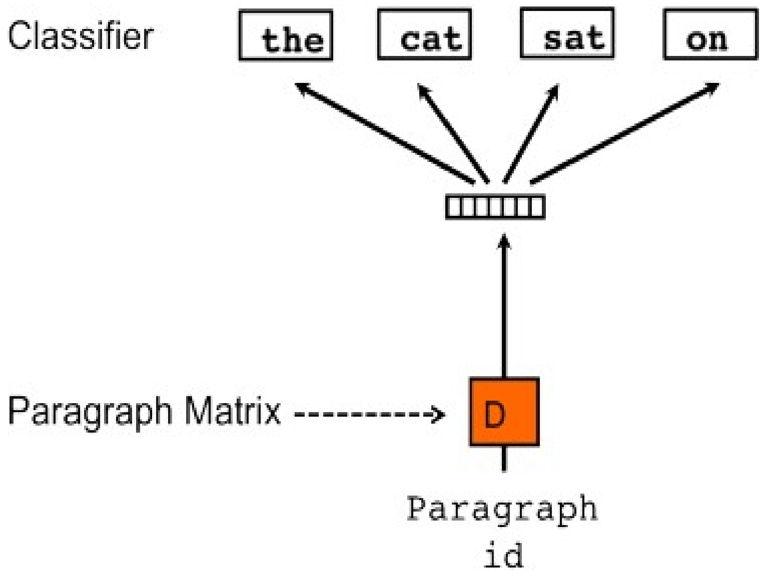
Paragraph Vectors - Distributed Memory Model (PV-DM)



- PVs are obtained by training FFN on the synthetic task of predicting a next word based an average of both context word-vectors and the full document's paragraph vector.

- Similar to CBOW Word2Vec except with a new paragraph vector that represents the document concept.

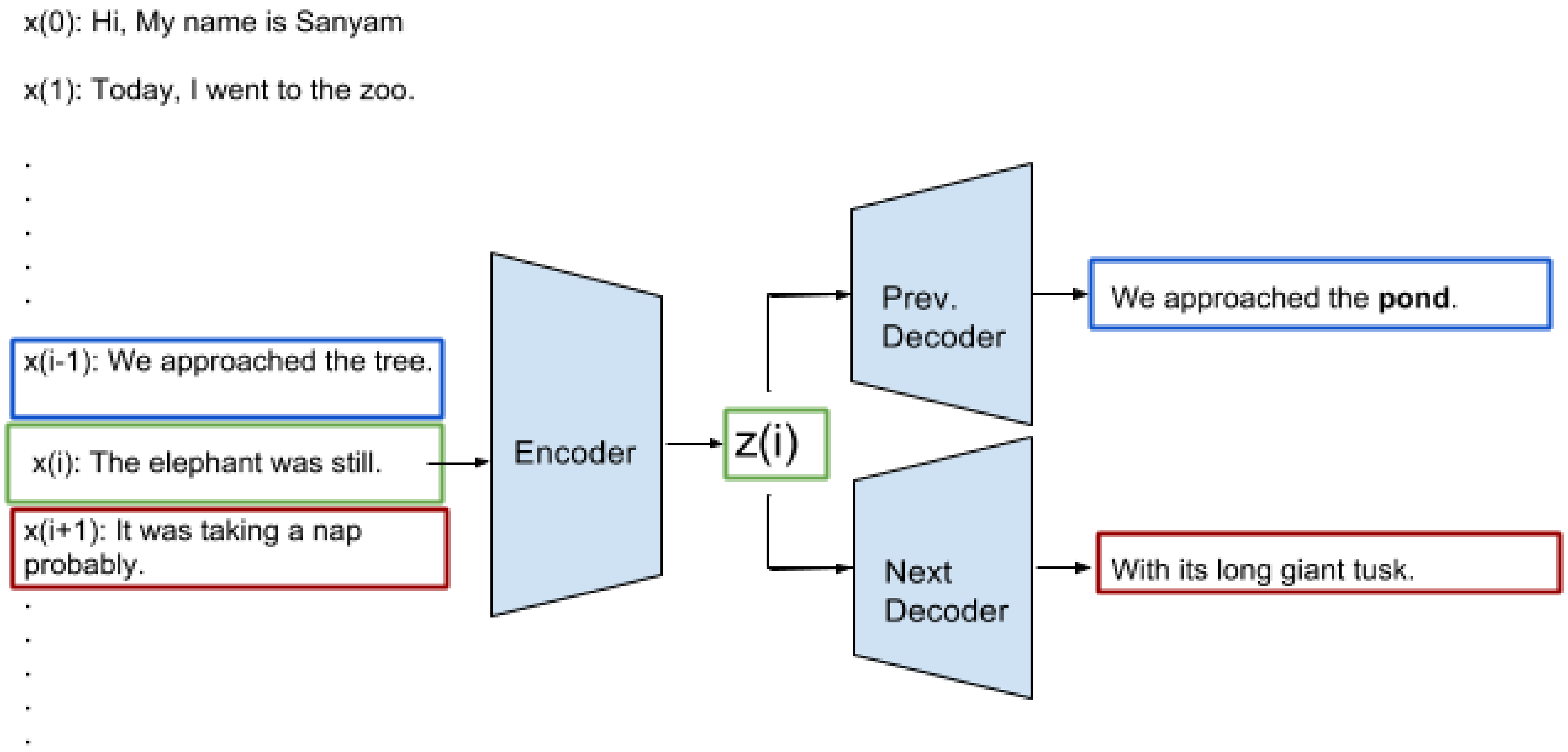
Paragraph Vector Distributed Bag Of Words (PVDBOW)



- PVs obtained by training a neural network on the synthetic task of predicting a target word just from the paragraph vector.
- Faster but may not be as accurate as the PV-DM.
- Similar to word2vec skipgram

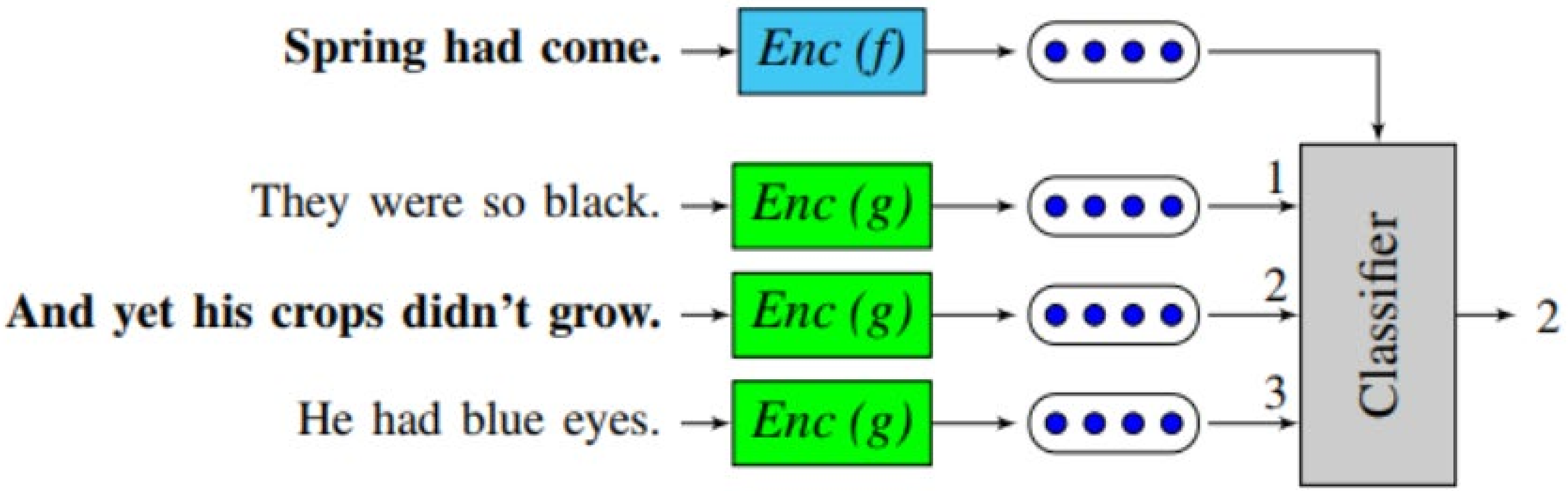
Sentence to Vectors

- **Skip thoughts** to generate the **previous** and **next** sentences.



Sentence to Vectors

Quick thoughts to predict/classify the **next** sentences.



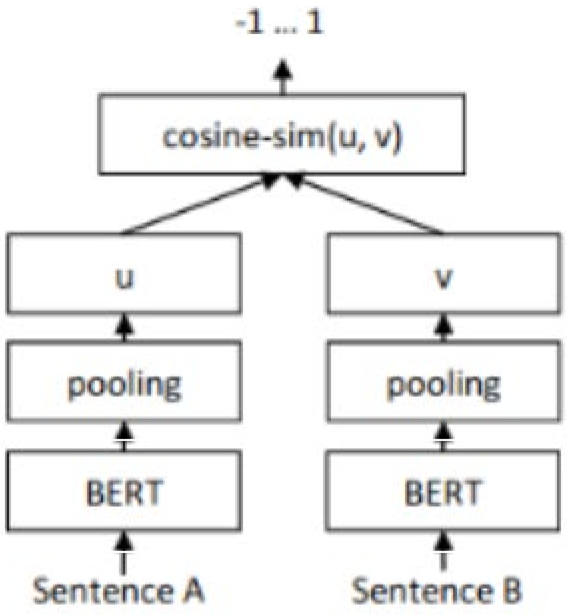
Quick thoughts

- Quick-thoughts generally to create document vectors

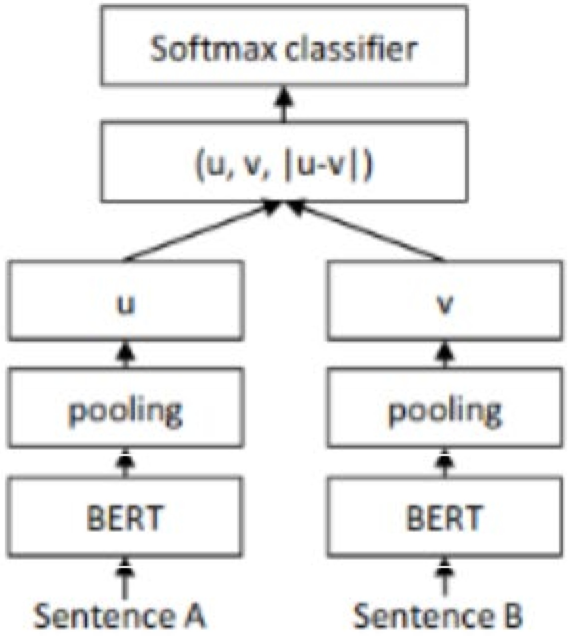
- skip thoughts more for word/sentence vectors.

Sentence to Vectors

Sentence-Bert 2018



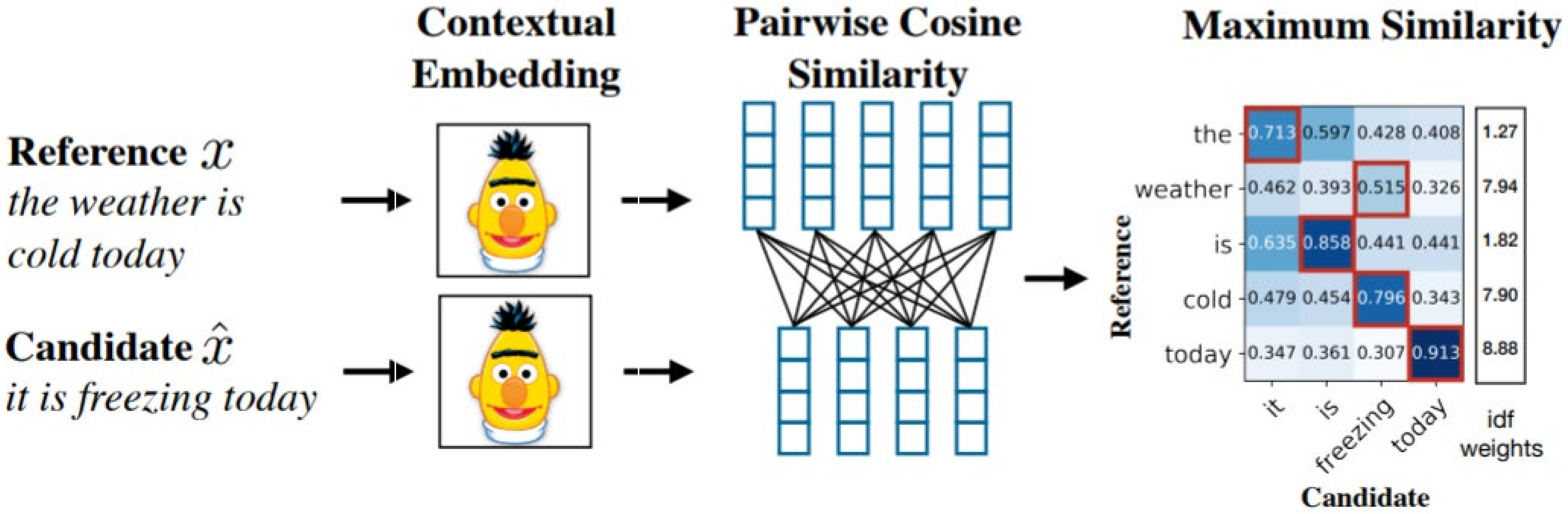
Sentence-BERT for sentence similarity(Regression Task)



Sentence-BERT for Classification task

Sentence to Vectors

BertScore 2020



Conclusion - selecting the best document embedding

- Averaging word vectors – strong benchmark. The word vectors can be generated from word2vec, GLOVE, BERT, ELMO etc.

- Performance also a key consideration. Eg Fast2Sent simplified version of Skipthought

- No clear task-specific leaders → BERT (maxLen 512)

Reference

https://towardsdatascience.com/word-embeddings-and-document-vectors-part-2-order-reduction-2d11c3b5139c

https://www.aclweb.org/anthology/R13-1072.pdf

https://towardsdatascience.com/building-sentence-embeddings-via-quick-thoughts-945484cae273

https://papers.nips.cc/paper/6139-supervised-word-movers-distance.pdf

https://arxiv.org/pdf/1602.03483.pdf

https://arxiv.org/pdf/1904.09675.pdf

https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d

THANK YOU!