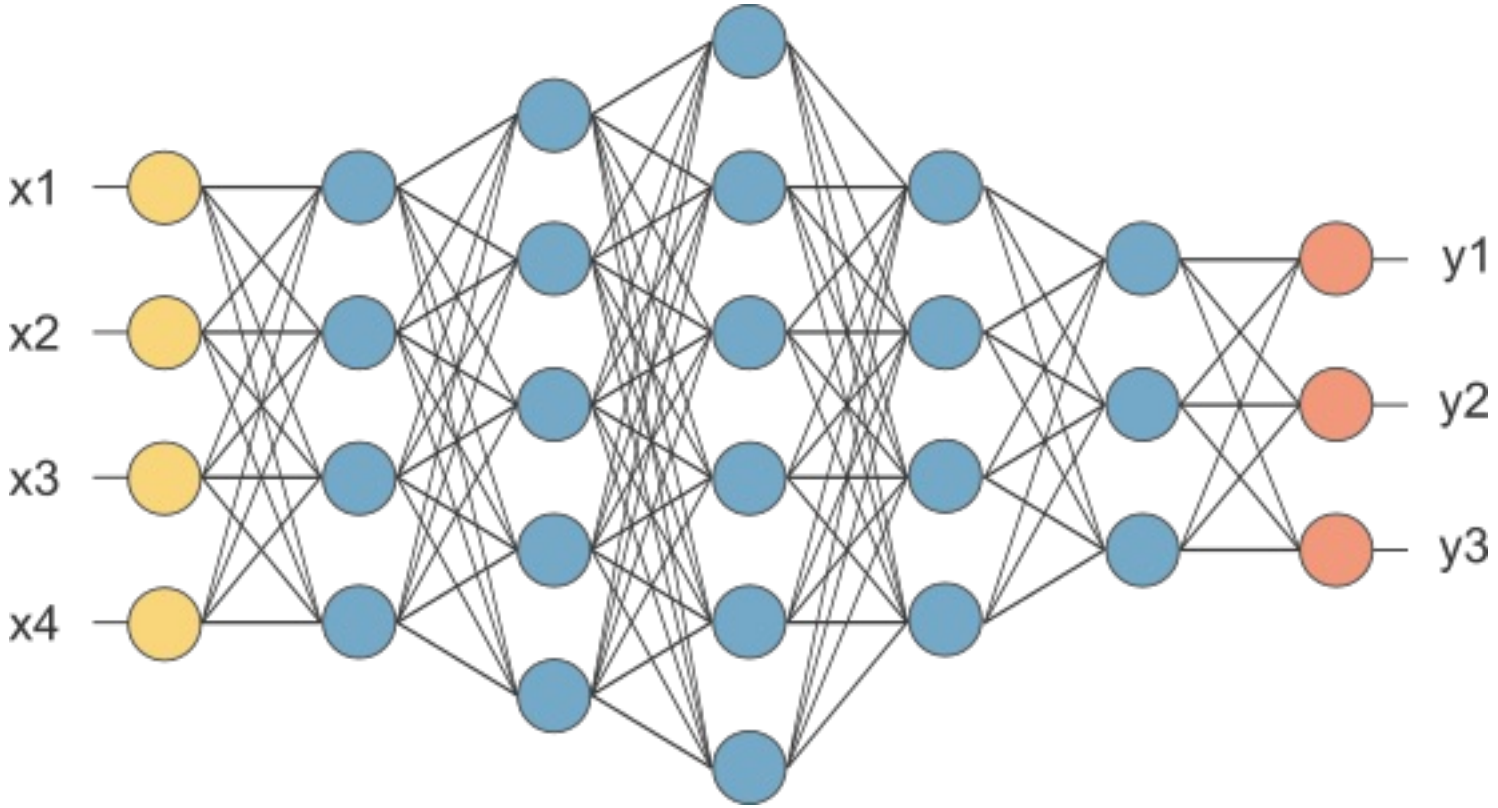
Attention and Transformers

Attention and Transformers

Plan: We trace back history to see how attention and transformers have emerged

1. Basic models, related to transduction models and attention.
2. Encoder-Decoder model, using recurrent networks such as LSTM.
3. Transformer models are general models sufficient for almost all biotech applications (graph models may be treated to be special cases too).
4. For example, DeepMind AlphaFold2 uses depends on a transformer architecture to train an end-to-end model.
5. The transformer model also makes it easy for large scale biological data (pre)training.

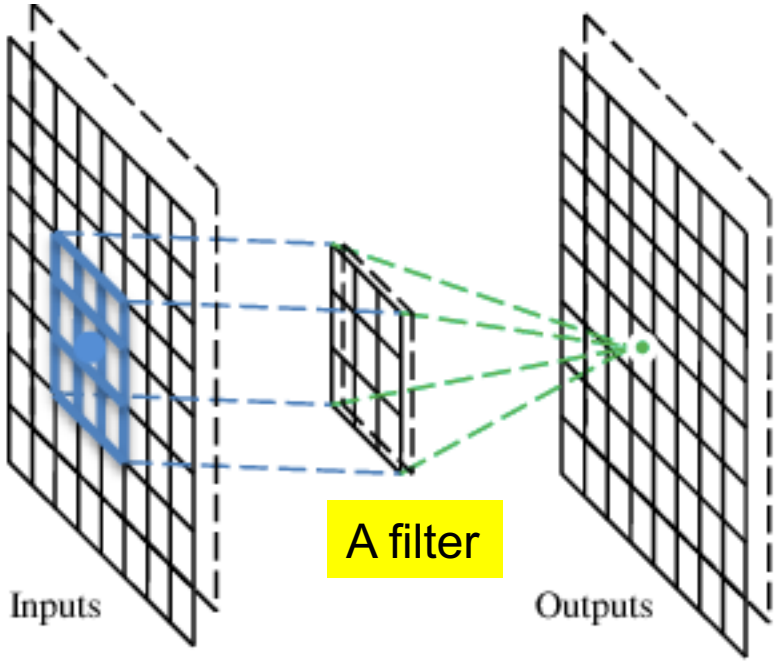
1. Fully connected network, feedforward network



To learn the weights on the edges

2. CNN

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that do convolutional operation.



Convolutional layer

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input

These are the network parameters to be learned.

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution Operation

1	-1	-1
-1	1	-1
-1	-1	1

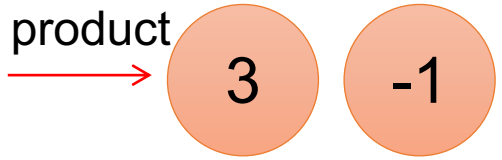
Filter 1

stride=1

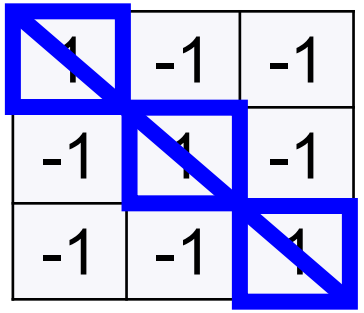
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Input

Dot product

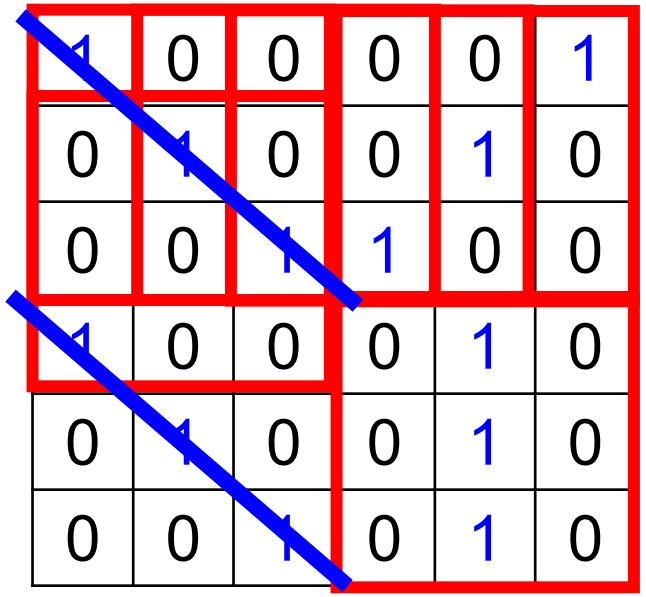


Convolution

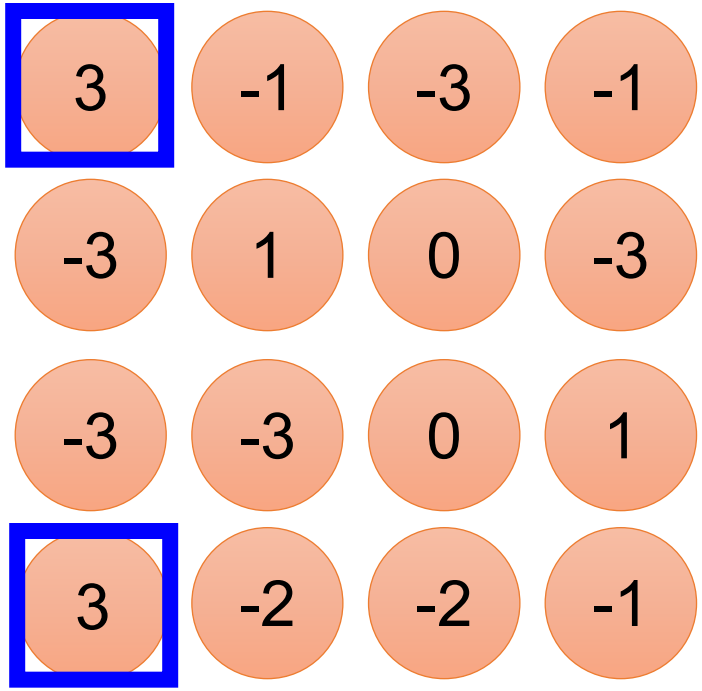


Filter 1

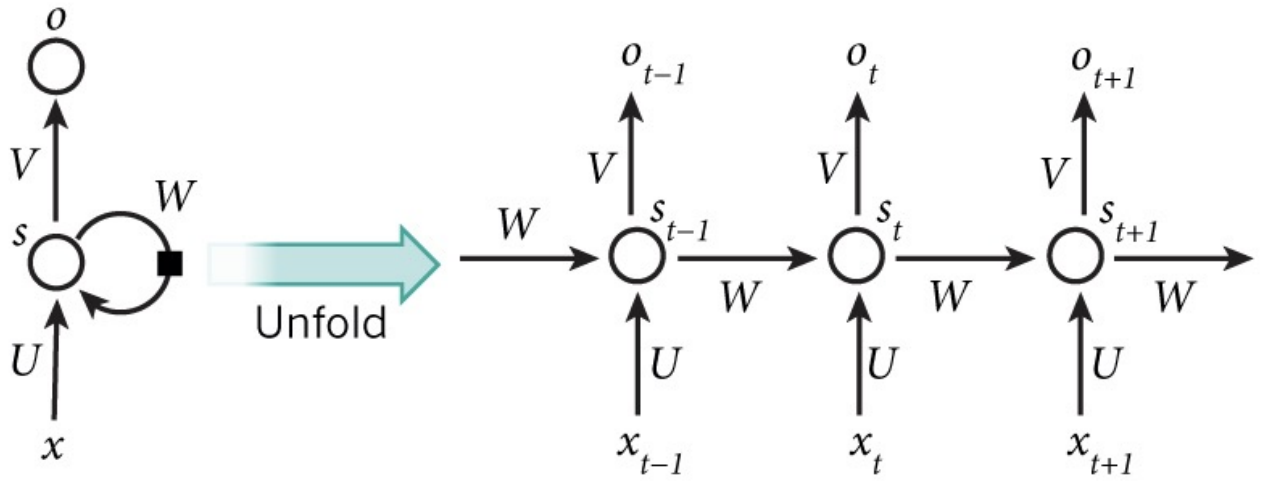
stride=1



Input

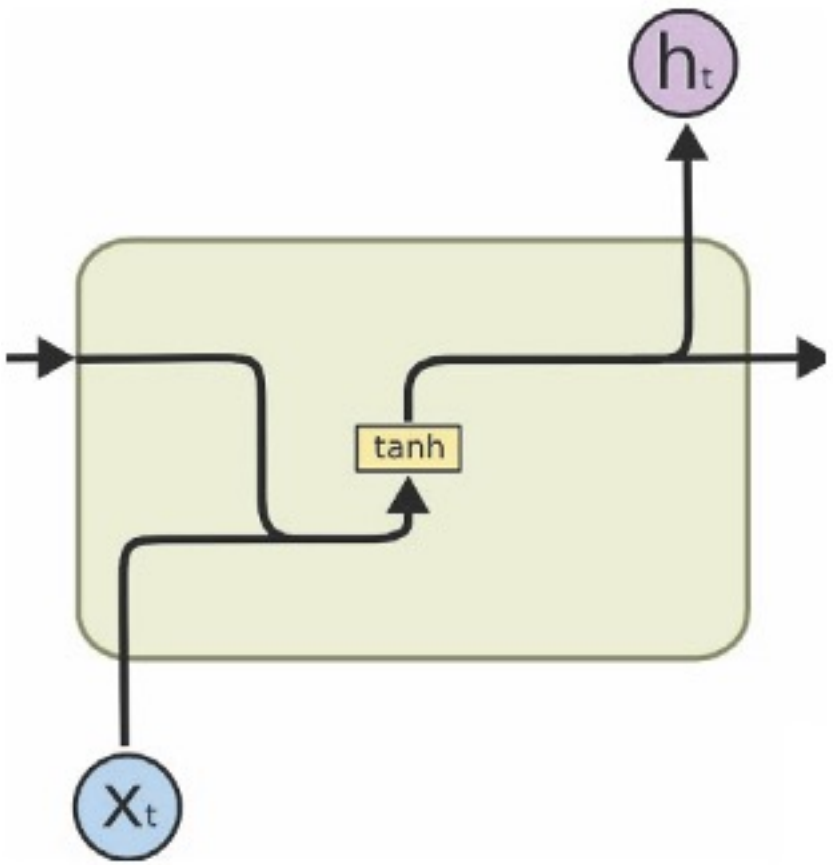


3. RNN

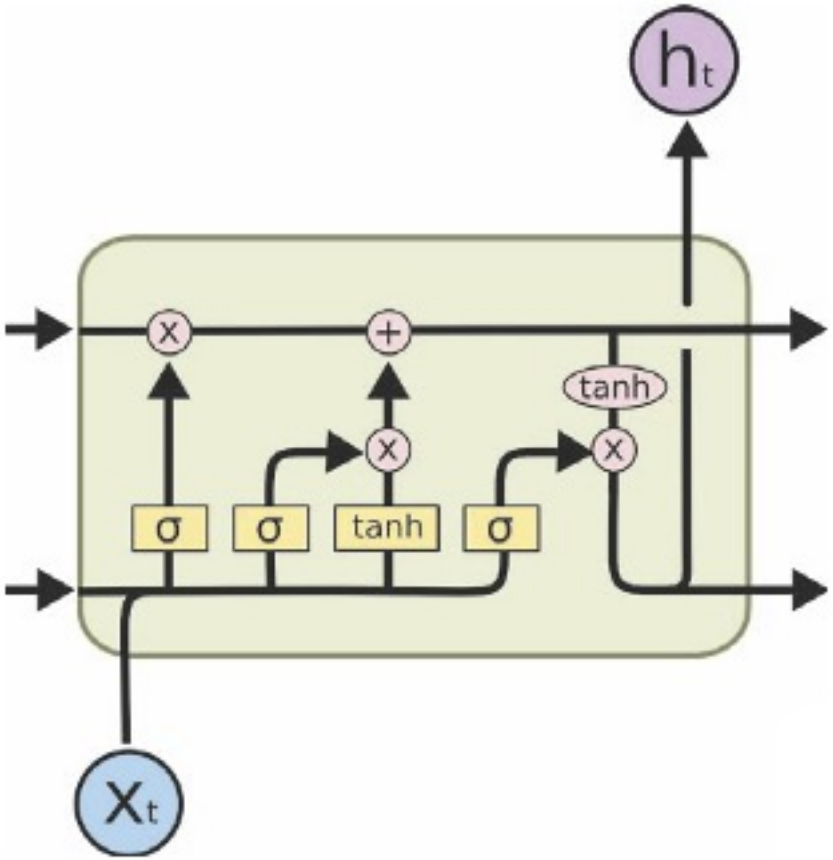


Parameters to be learned:
U, V, W

Simple RNN vs LSTM



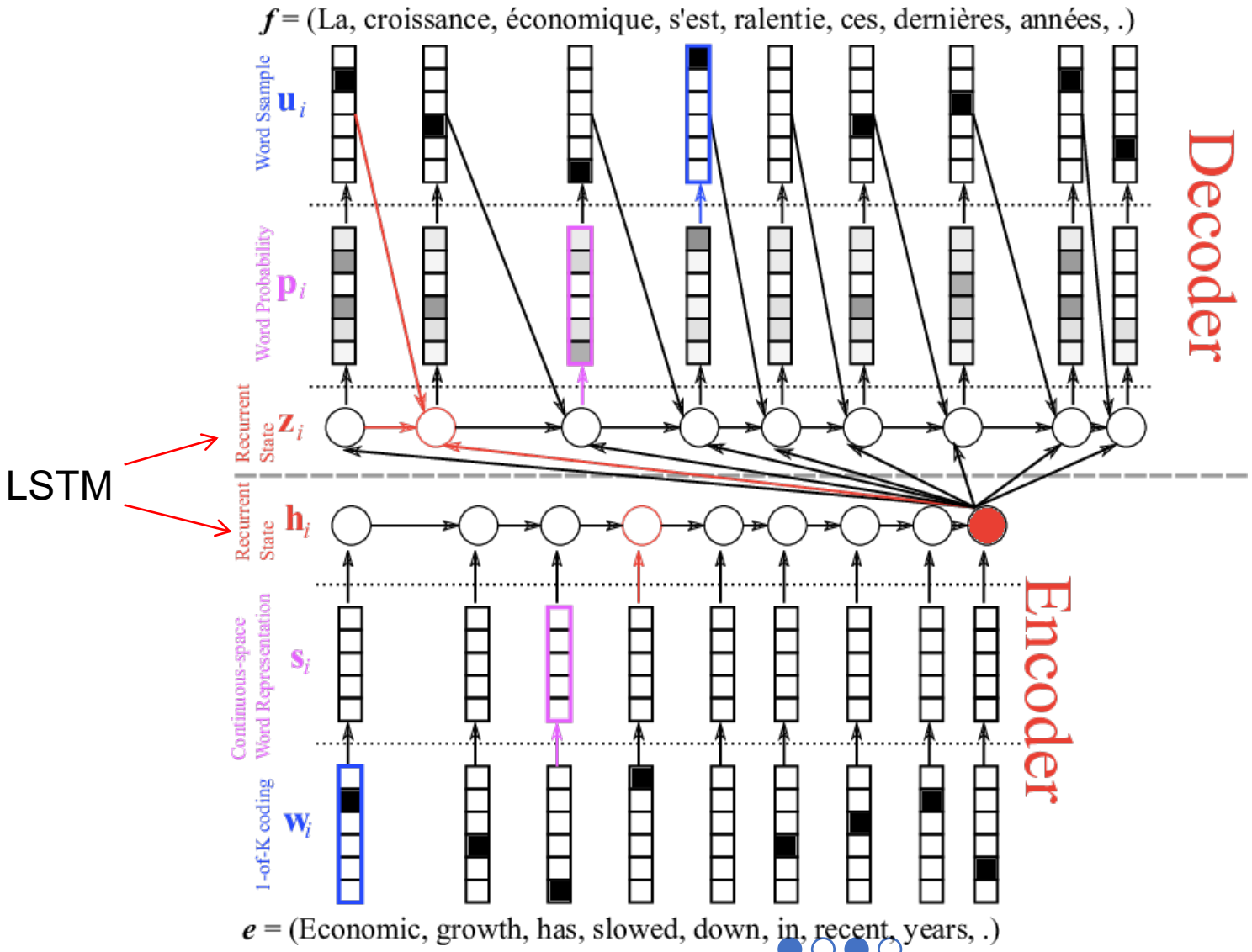
(a) RNN



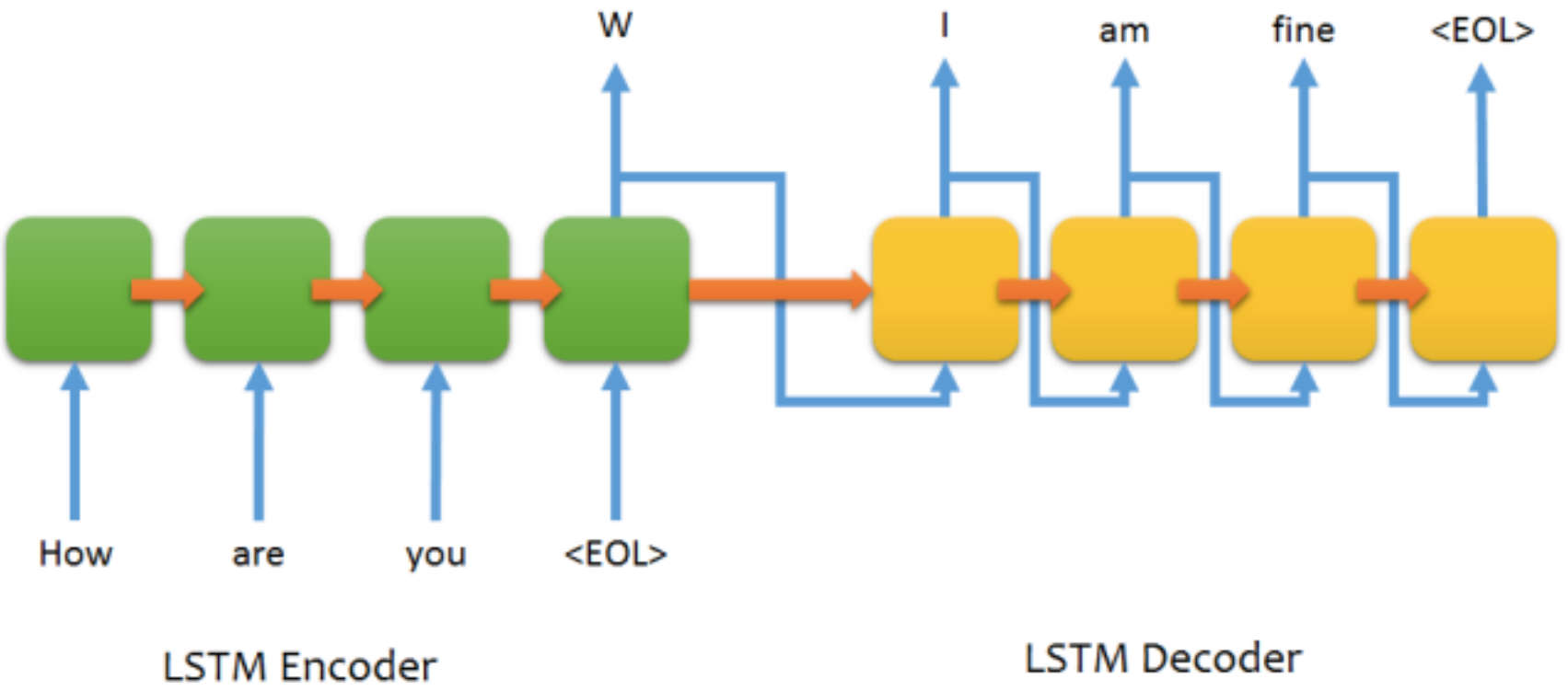
(b) LSTM

Attention and Transformers

Encoder-Decoder machine translation



Encoder-Decoder LSTM structure for chatting



Attention

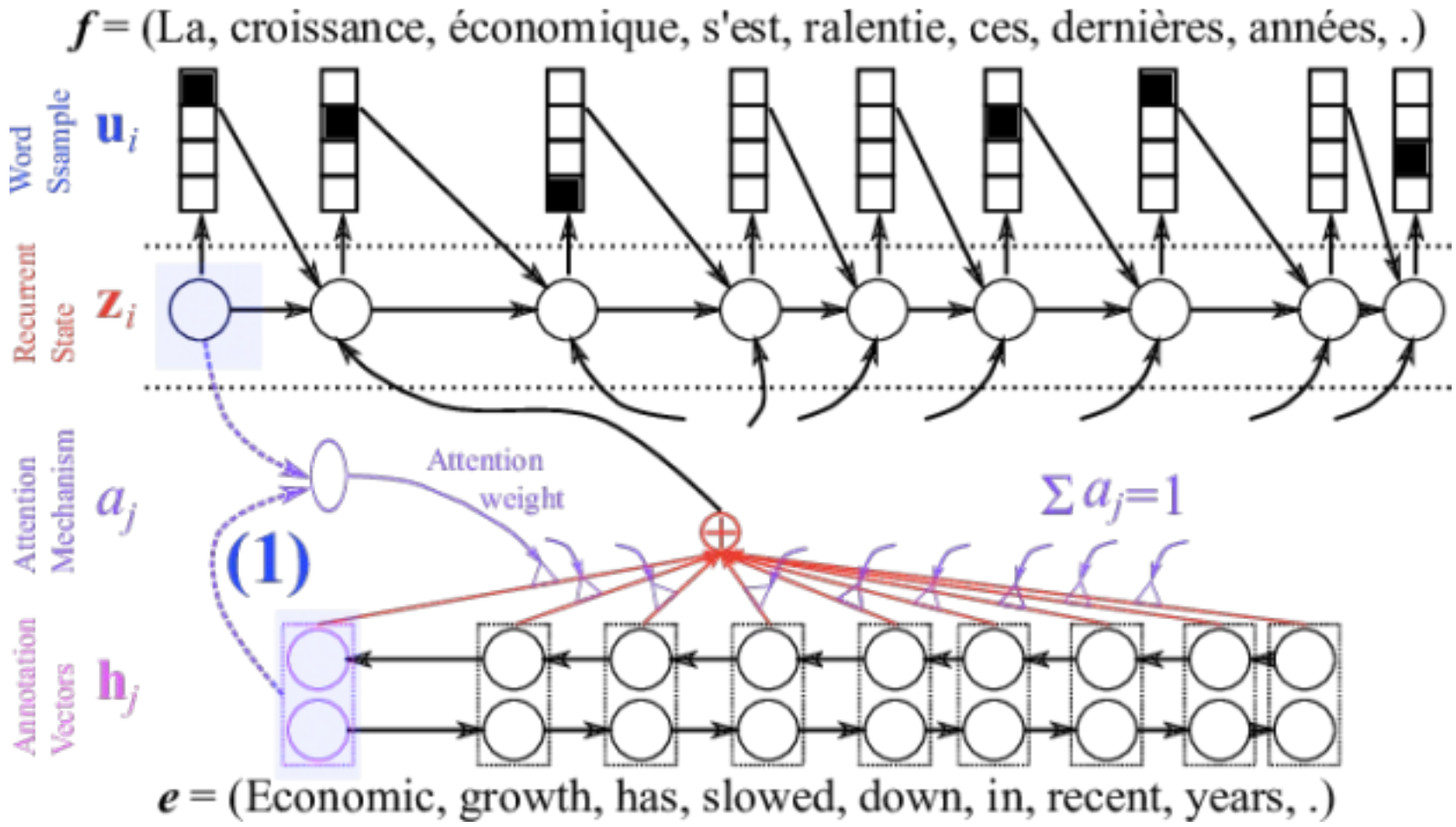
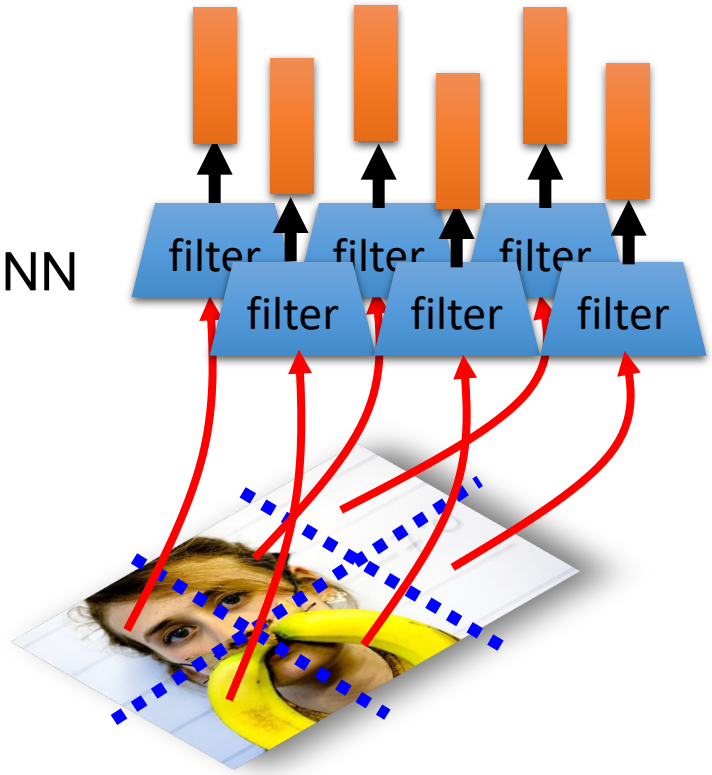

f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)
Word Ssample
u_i
Recurrent State
z_i
Attention Mechanism
a_j
Attention weight
$\sum a_j=1$
(1)
Annotation Vectors
h_j
e = (Economic, growth, has, slowed, down, in, recent, years, .)

Image caption generation using attention

A vector for each region

CNN



z^0 is initial parameter, it is also learned

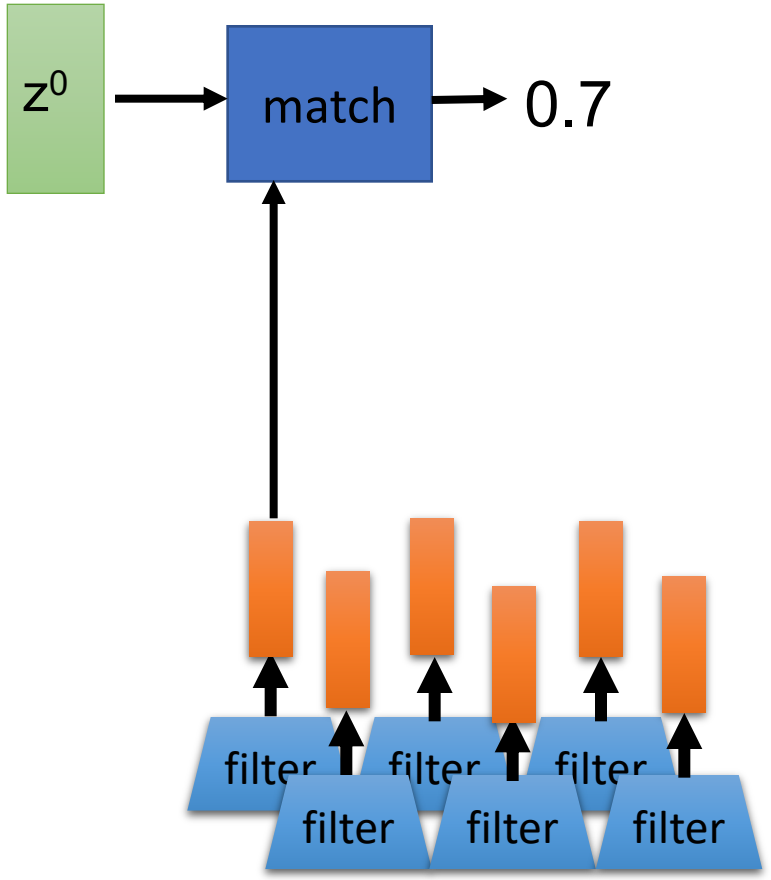


Image caption generation using attention

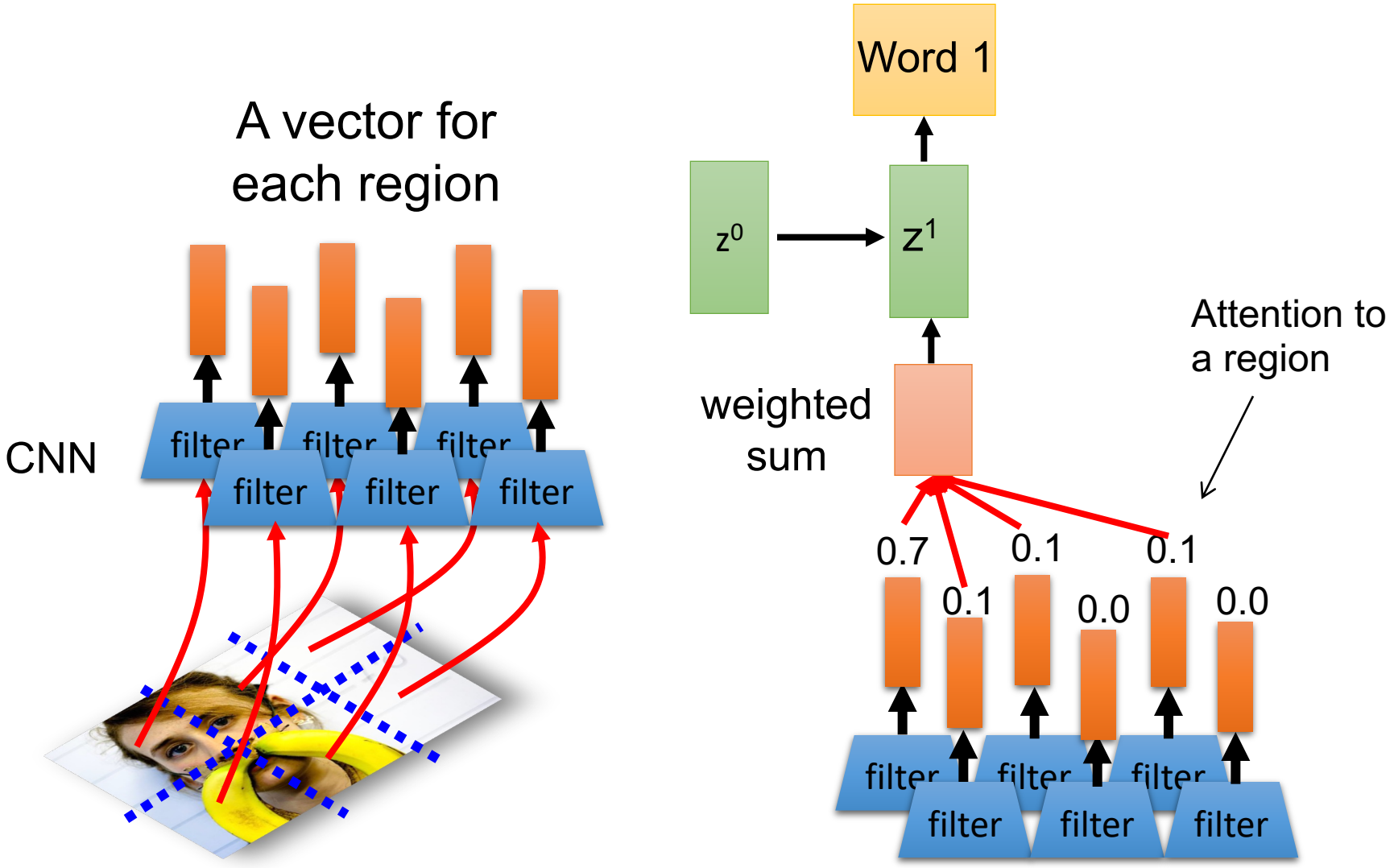


Image caption generation using attention

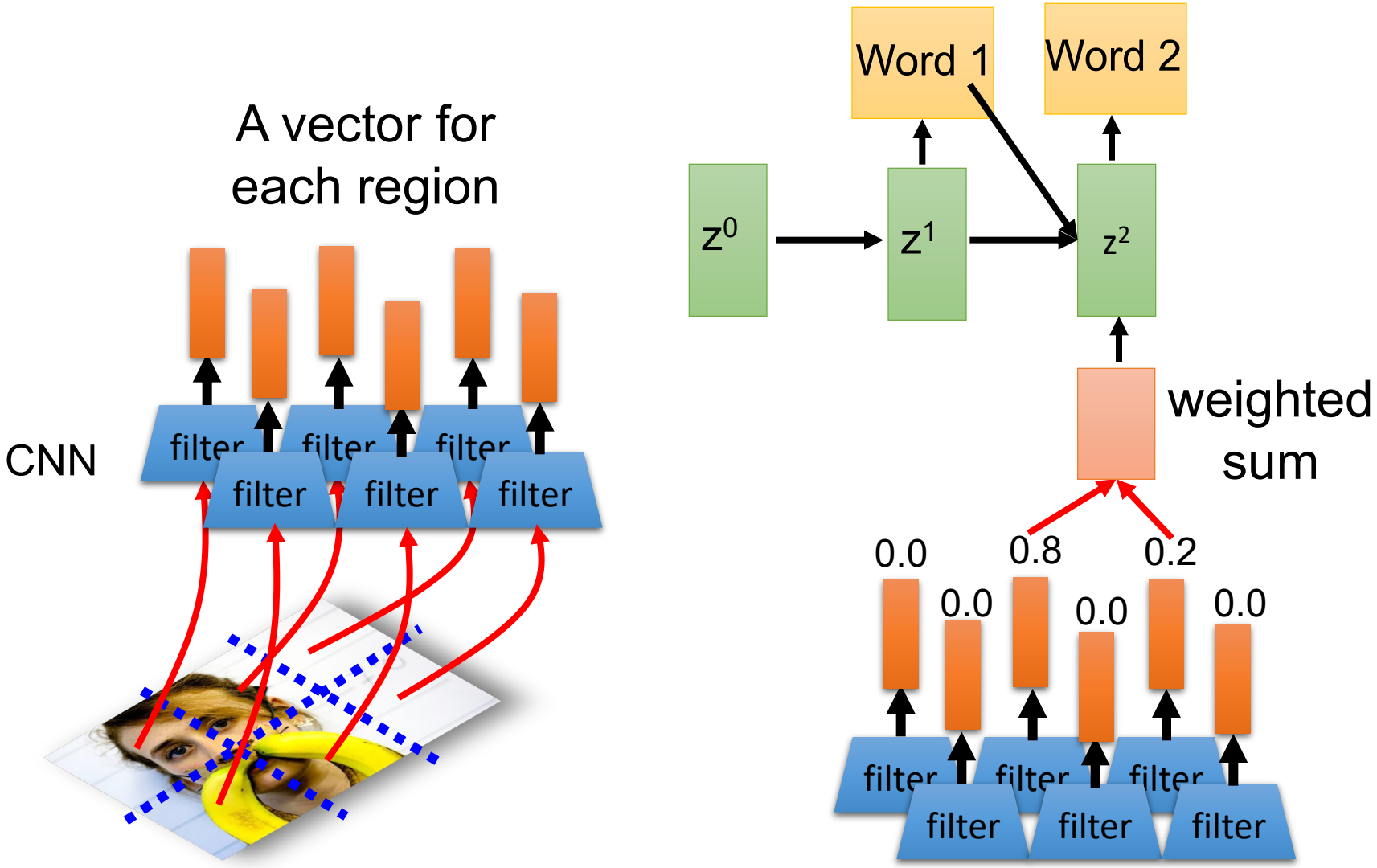


Image caption generation using attention

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.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

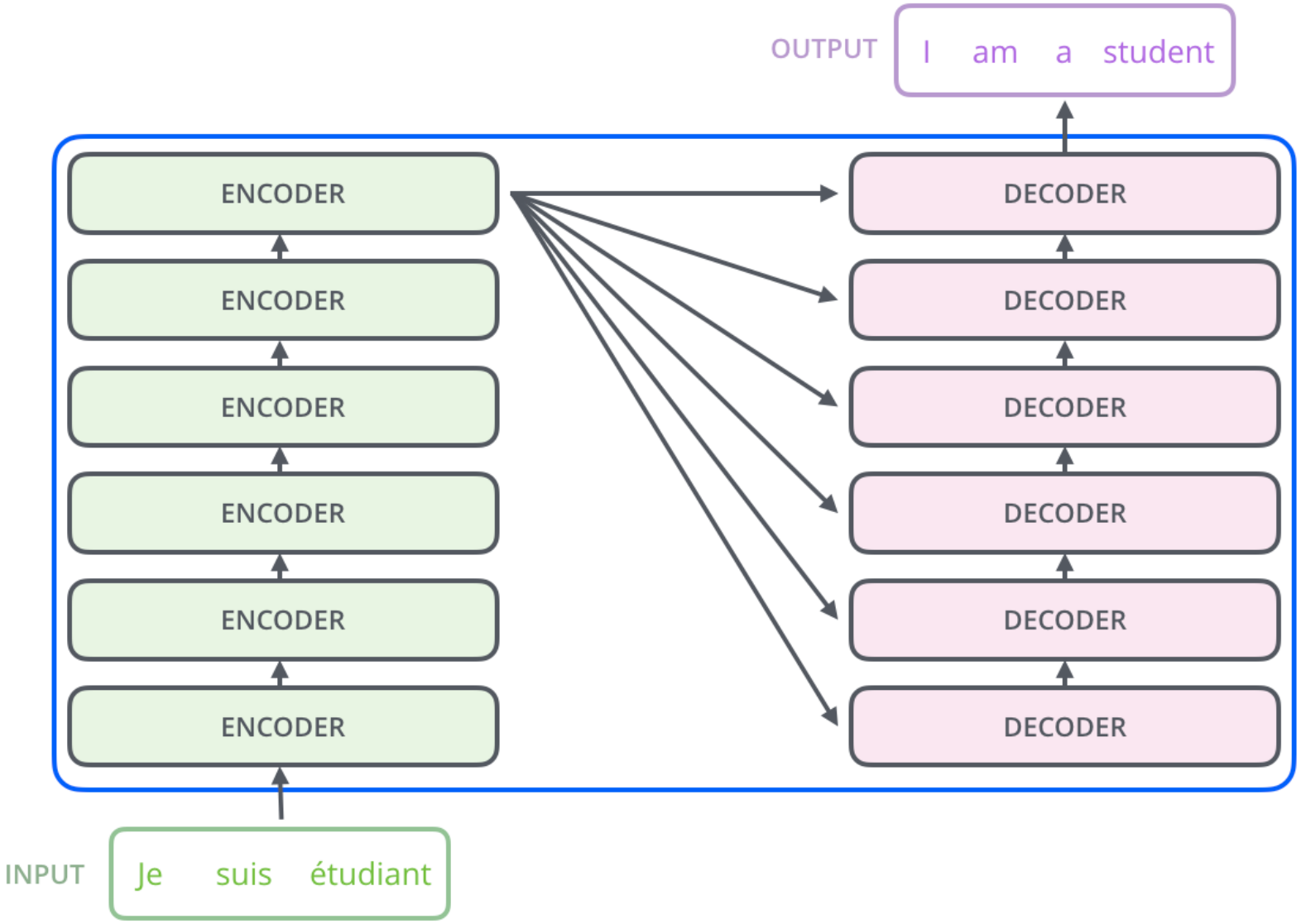
A giraffe standing in a forest with trees in the background.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML, 2015

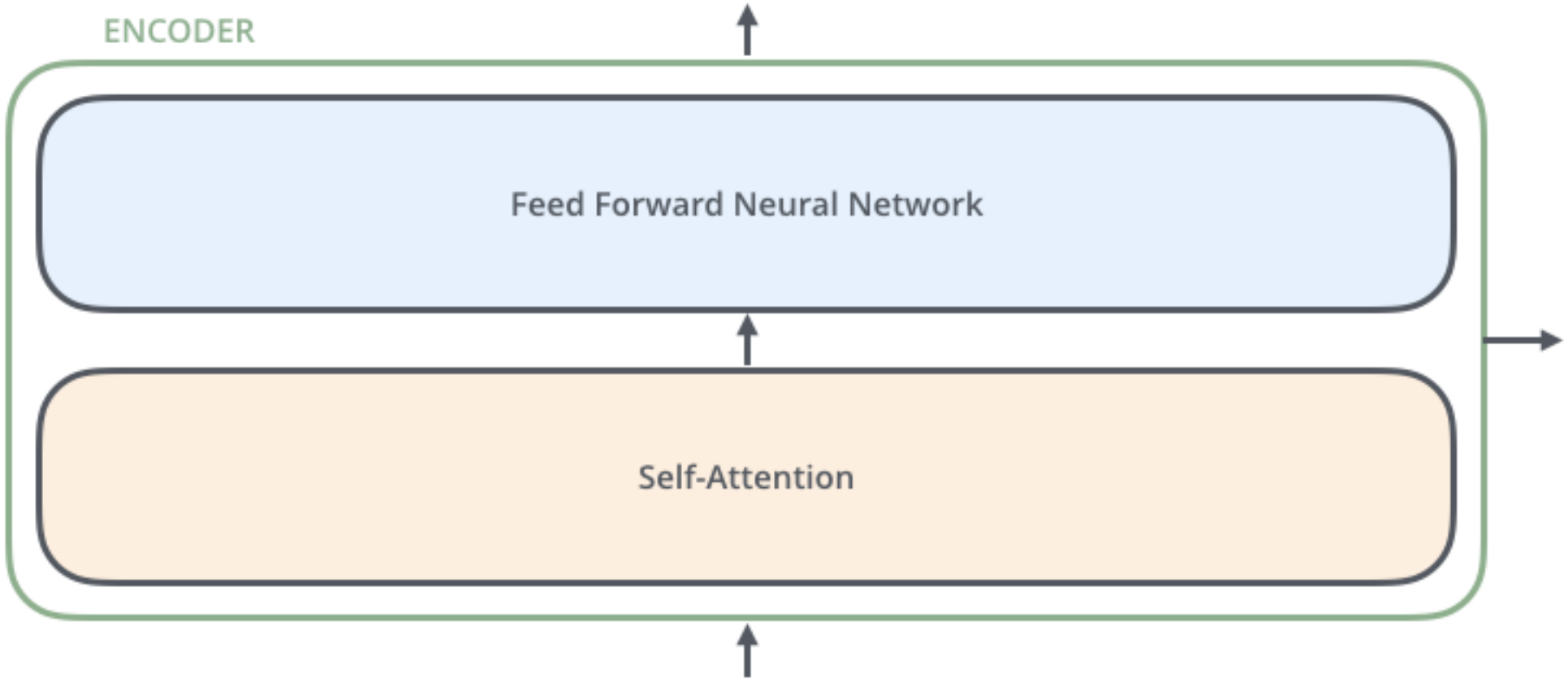
More new ideas:

1. ULM-FiT, pre-training, transfer learning in NLP
2. Recurrent models require linear sequential computation, hard to parallelize. ELMo, bidirectional LSTM.
3. In order to reduce such sequential computation, several models based on CNN are introduced, such as ConvS2S and ByteNet. Dependency for ConvS2S needs linear depth, and ByteNet logarithmic.
4. The transformer is the first transduction model relying entirely on self-attention to compute the representations of its input and output without using RNN or CNN.

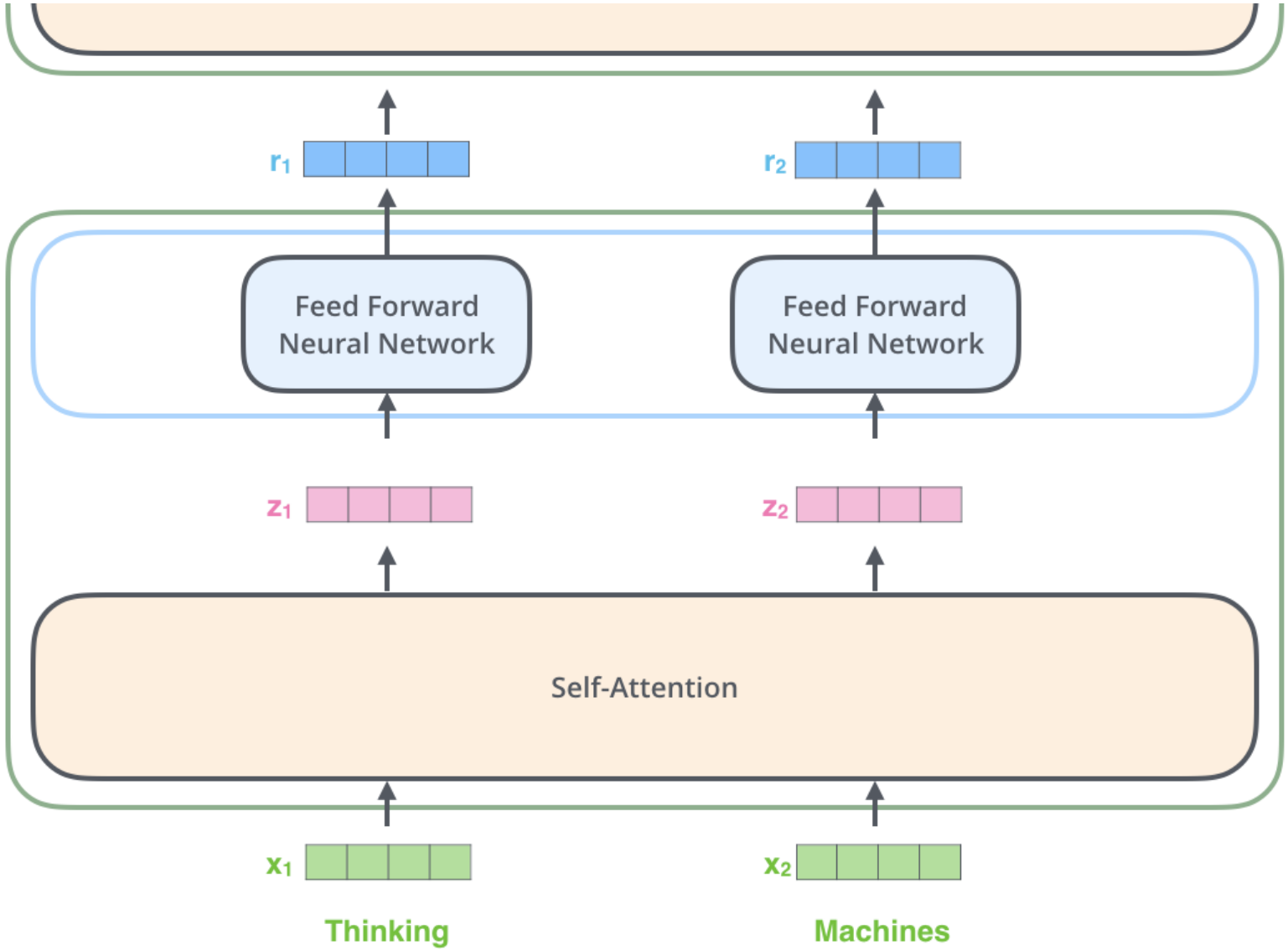
Transformer



An Encoder Block: same structure, different parameters

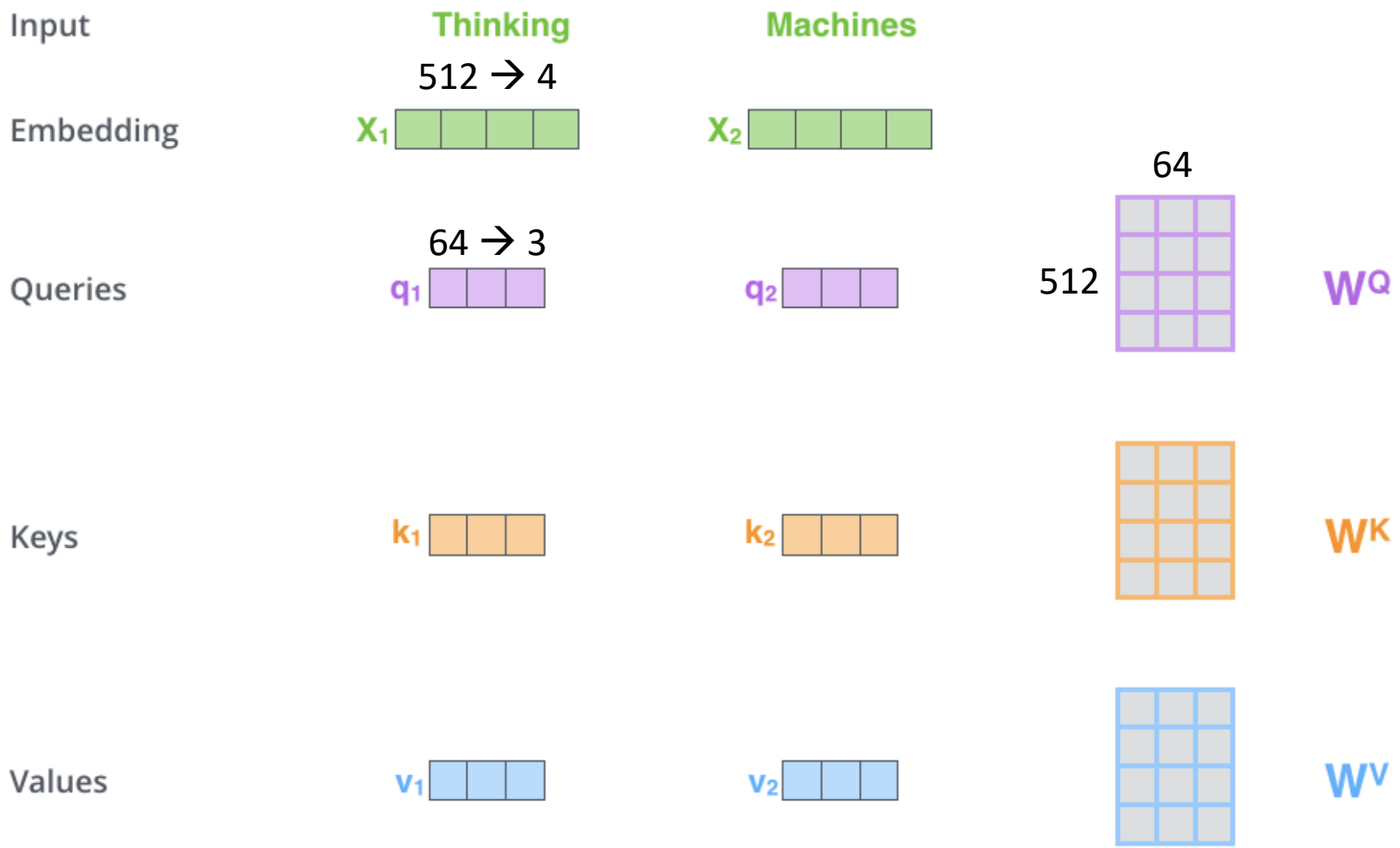


Encoder



Note: The ffnn is independent for each word.
Hence can be parallelized.

Self Attention

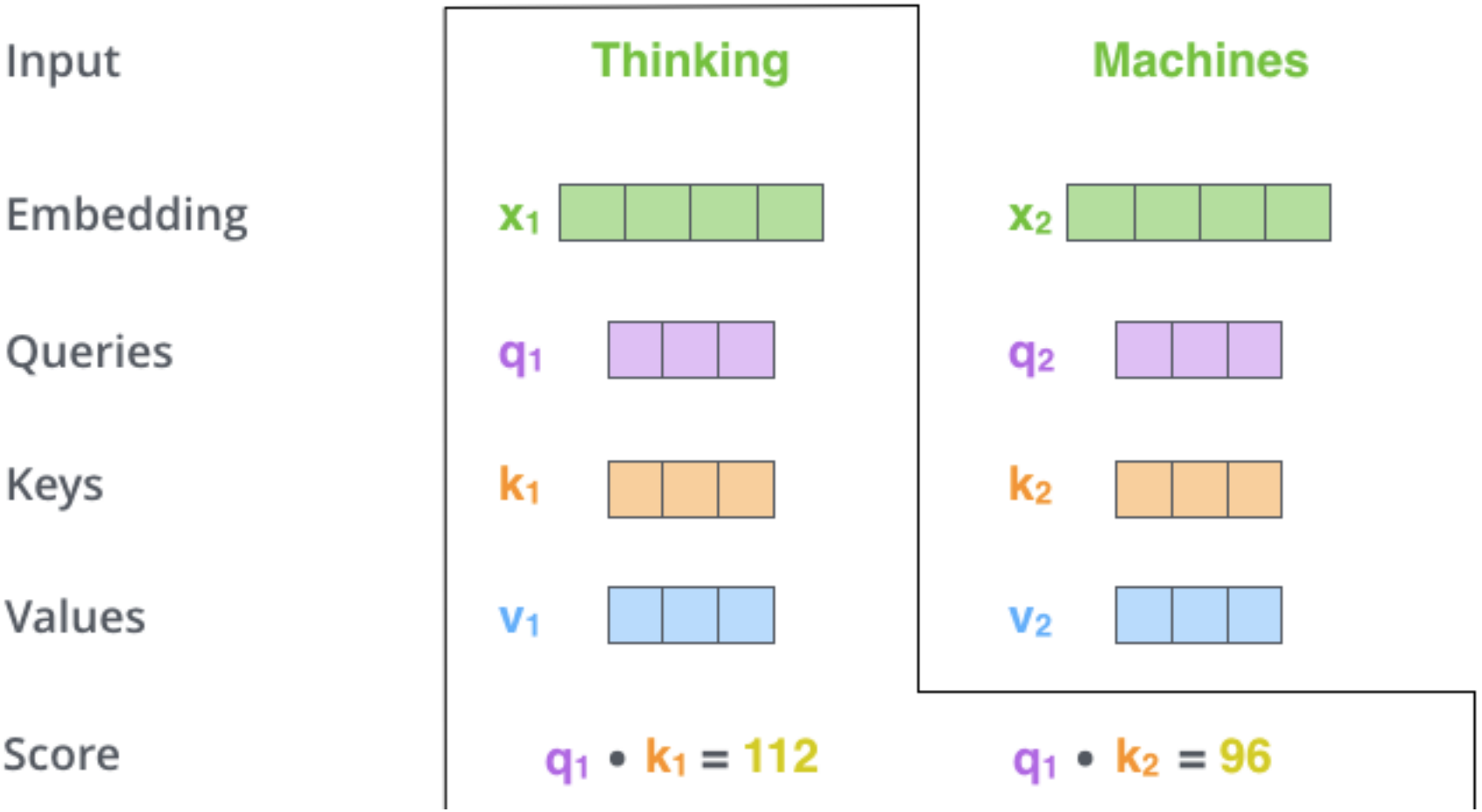


First we create three vectors by multiplying input embedding (1x512)
x_i with three matrices (64x512):

$q_i = x_i W^Q$
$K_i = x_i W^K$
$V_i = x_i W^V$

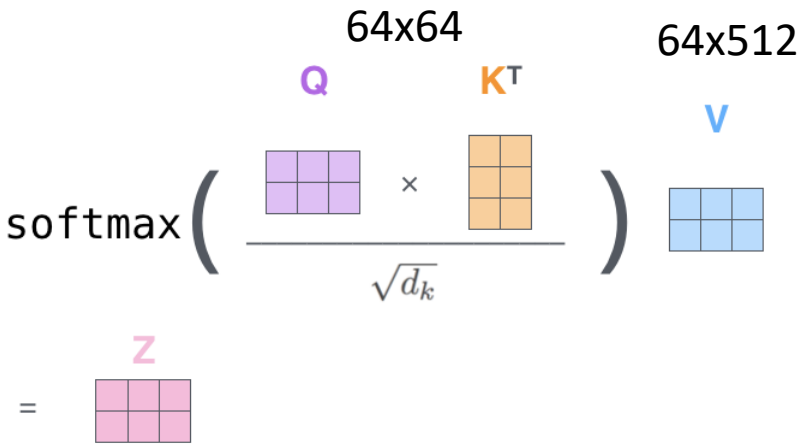
Self Attention

Now we need to calculate a score to determine how much focus to place on other Parts of the input.

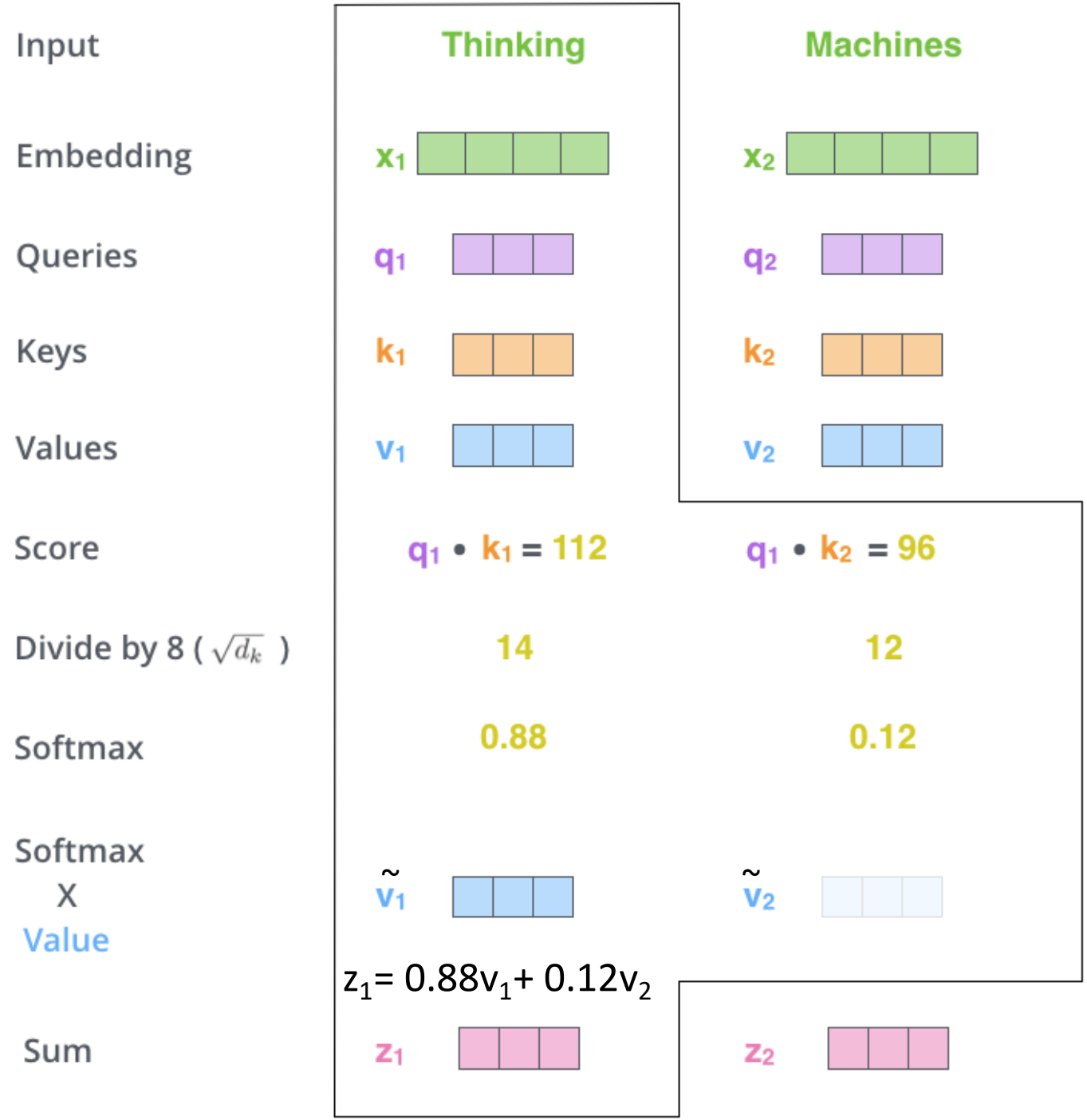


Self Attention

Formula

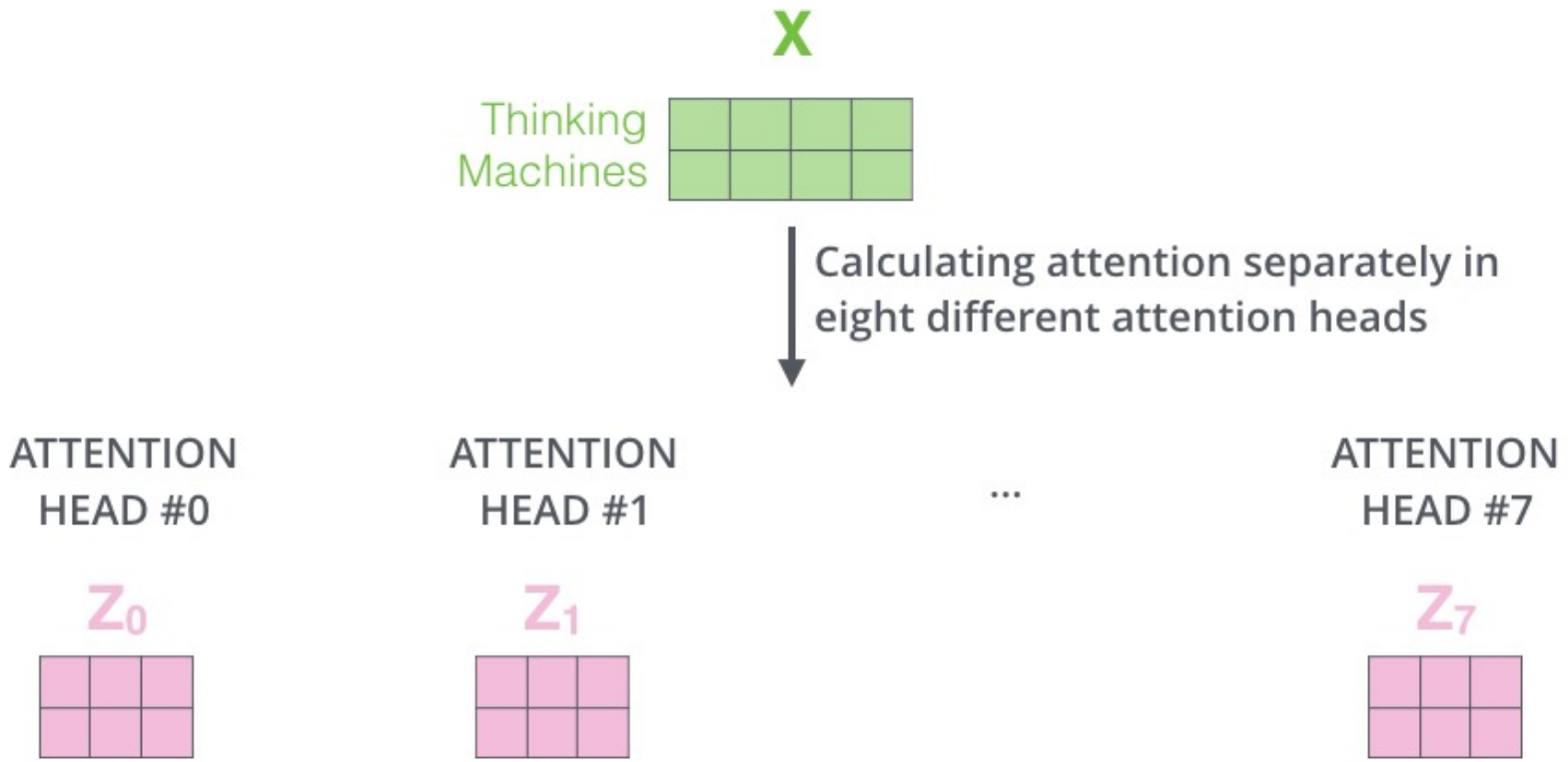


d_k=64 is dimension of key vector



Multiple heads

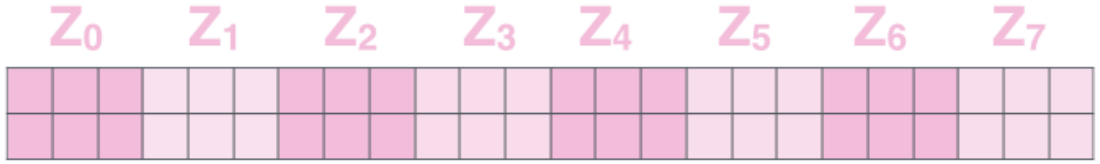
1. It expands the model's ability to focus on different positions.
2. It gives the attention layer multiple "representation subspaces"



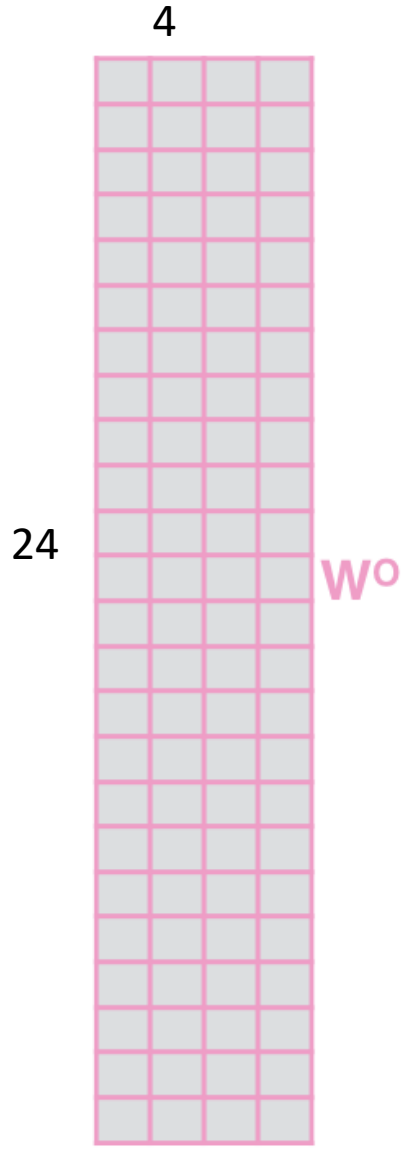
Attention and Transformers

The output is expecting only a 2x4 matrix, hence,

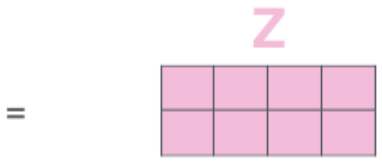
1) Concatenate all the attention heads



2) Multiply with a weight matrix W^O that was trained jointly with the model



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

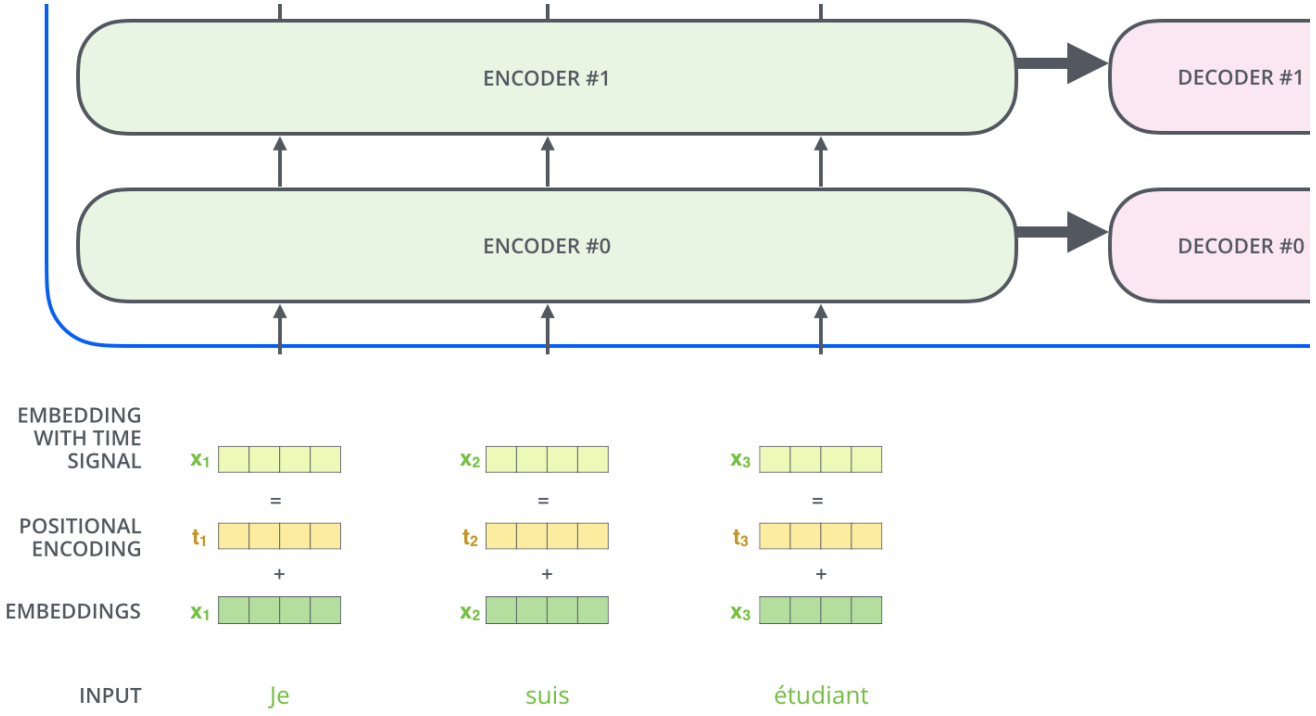


If you want some more intuition on attention: watch https://www.youtube.com/watch?v=-9vVhYEXeyQ

Representing the input order (positional encoding)

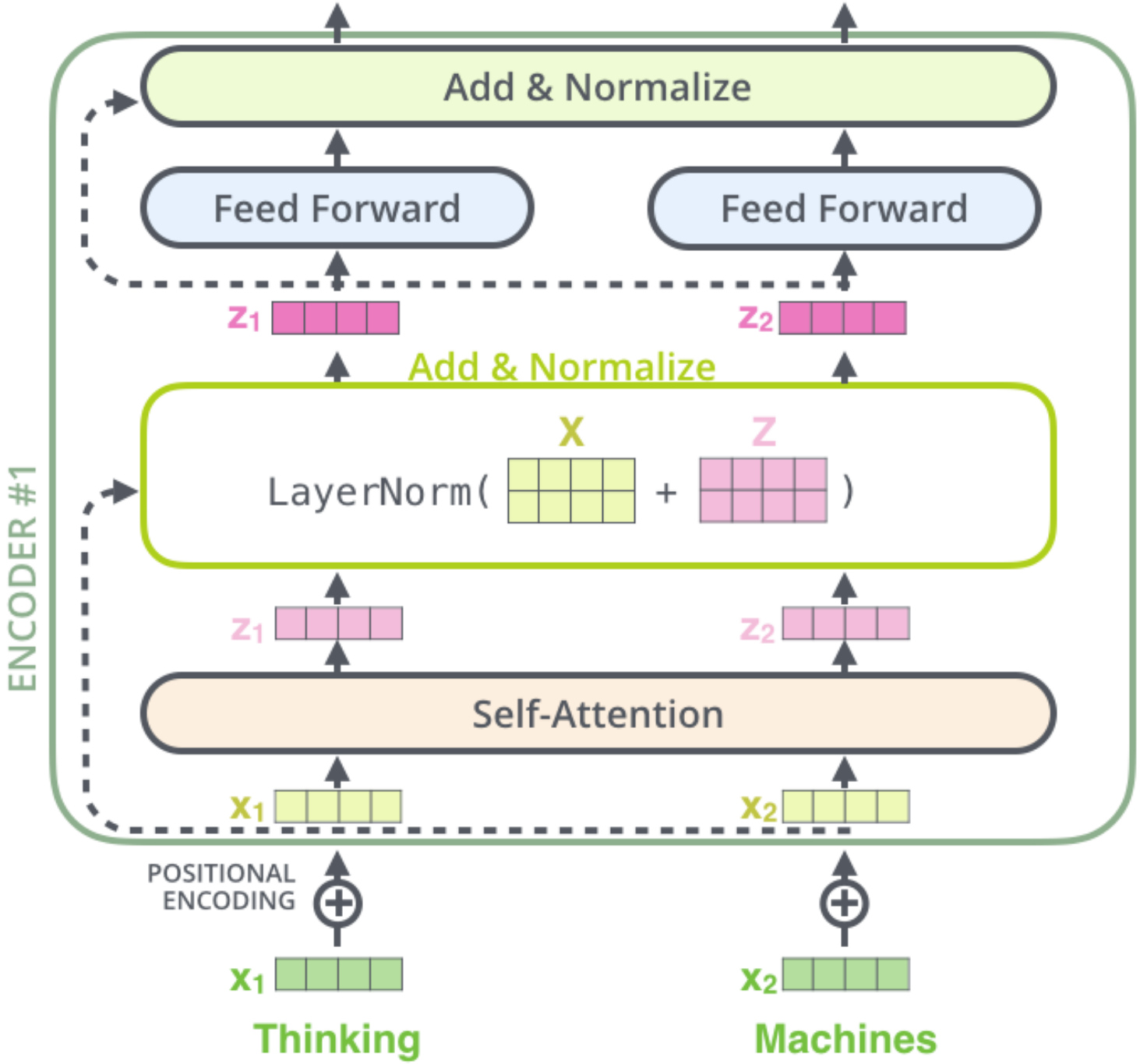
The transformer adds a vector to each input embedding. These vectors follow a specific pattern that the model learns, which helps it determine the position of each word, or the distance between different words in the sequence. The intuition here is that adding these values to the embedings provides meaningful distances between the embedding vectors once they're projected into Q/K/V vectors and during dot-product attention.

More on positional encoding:
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/



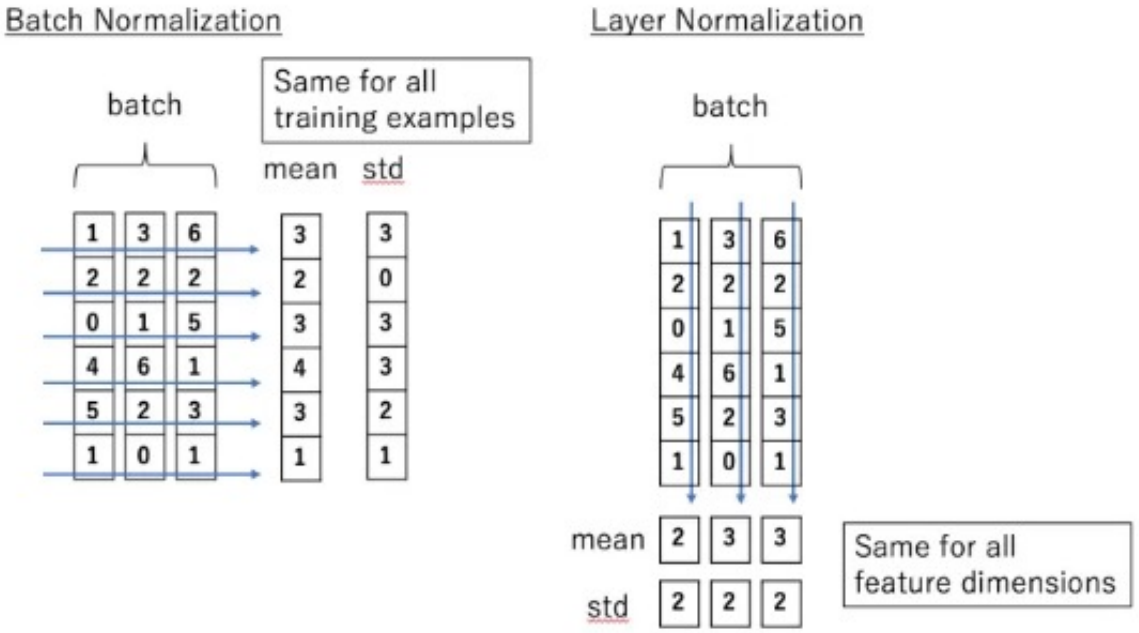
Add and Normalize

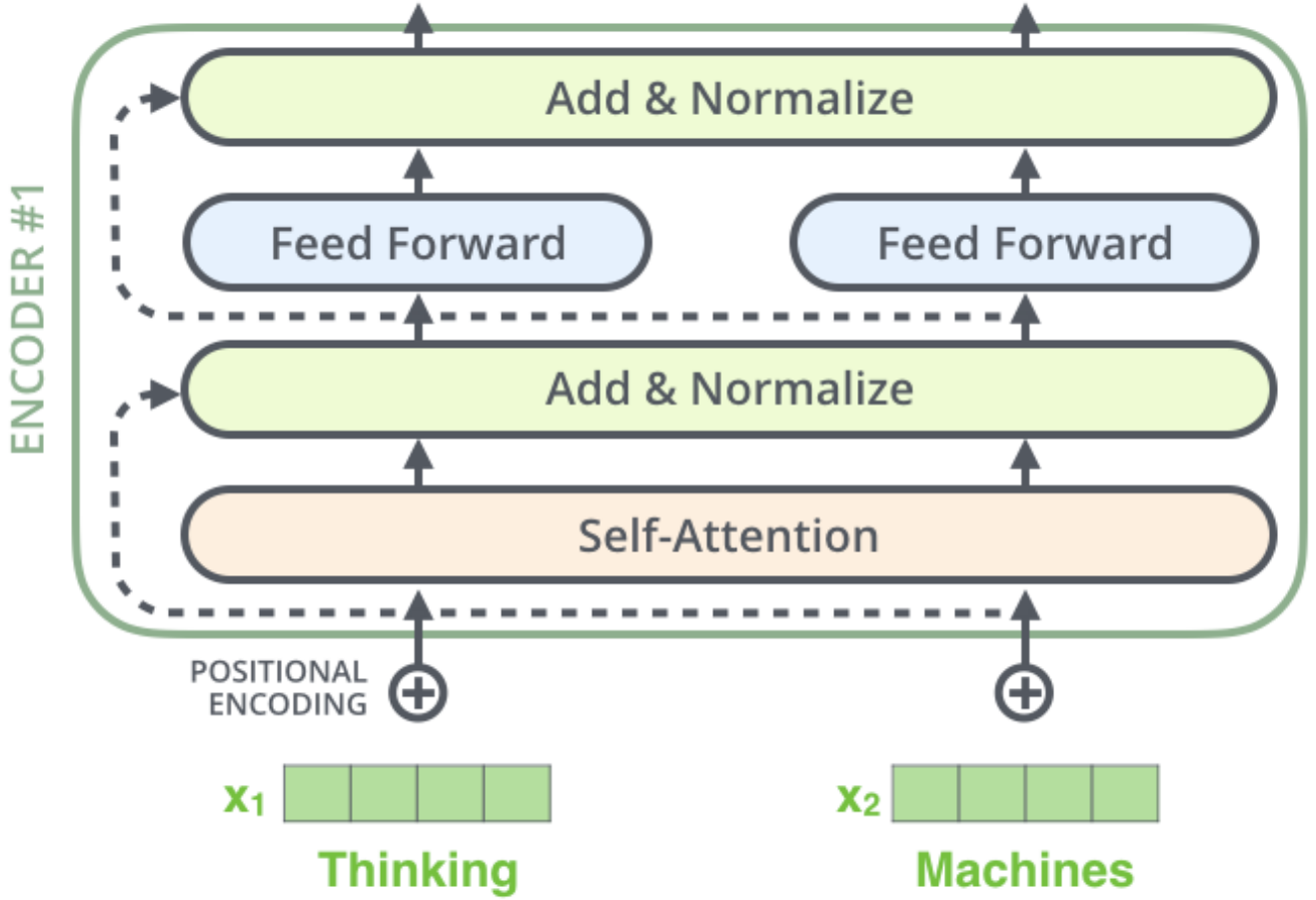
In order to regulate the computation, this is a normalization layer so that each feature (column) have the same average and deviation.



Layer Normalization (Hinton)

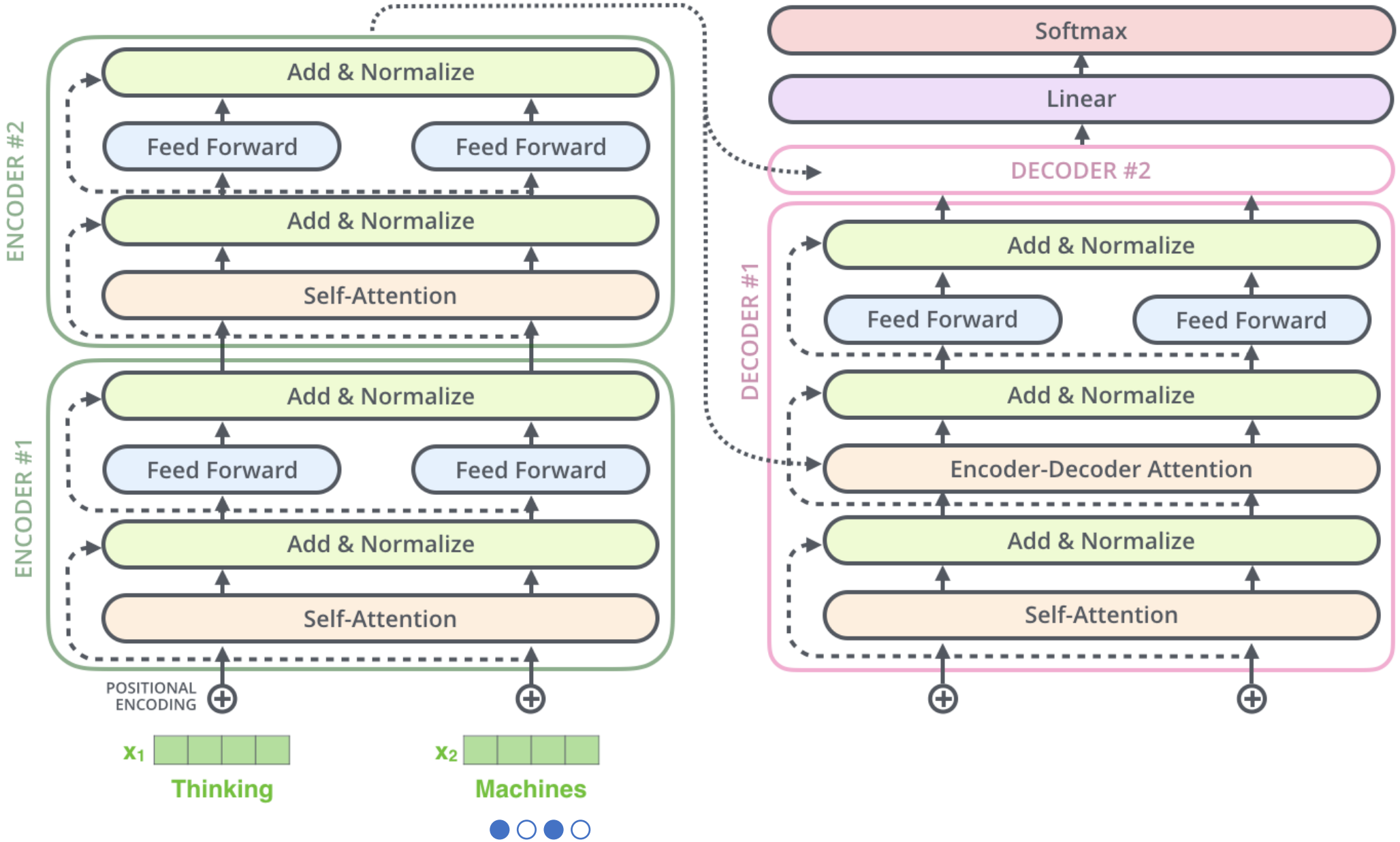
Layer normalization normalizes the inputs across the features.





The complete transformer

The encoder-decoder attention is just like self attention, except it uses K, V from the top of encoder output, and its own Q



Note: In decoder, the input is “incomplete” when calculating self-attention.

The solution is to set future unknown values with “-inf”.

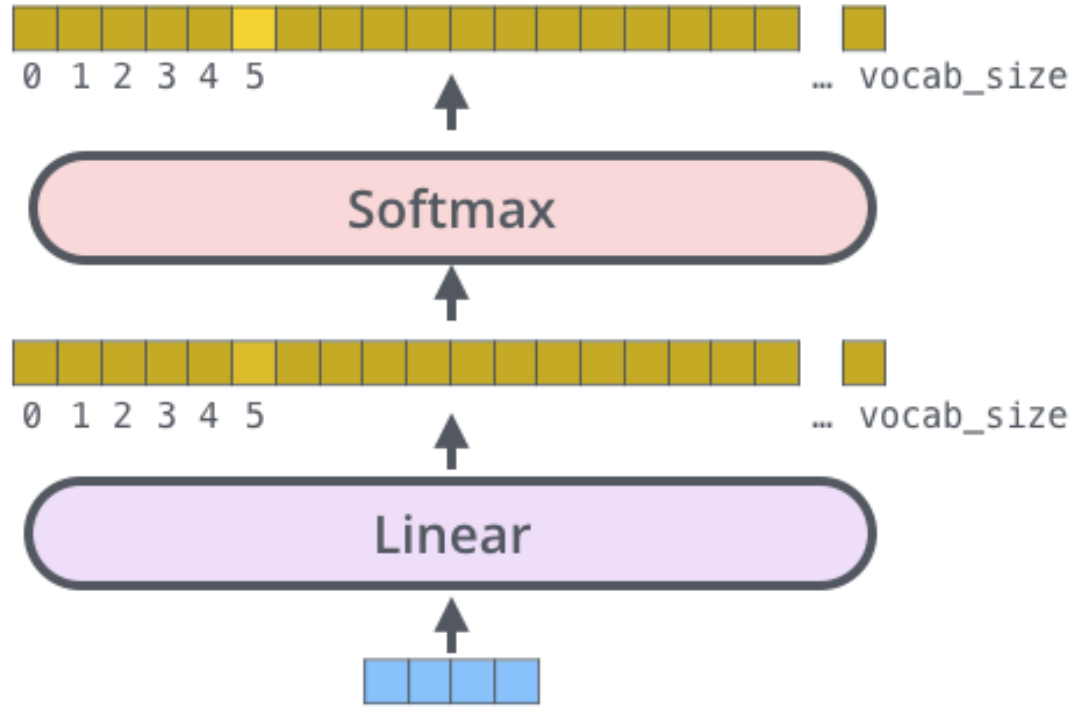
Decoder's Output Linear Layer

Which word in our vocabulary is associated with this index?

am

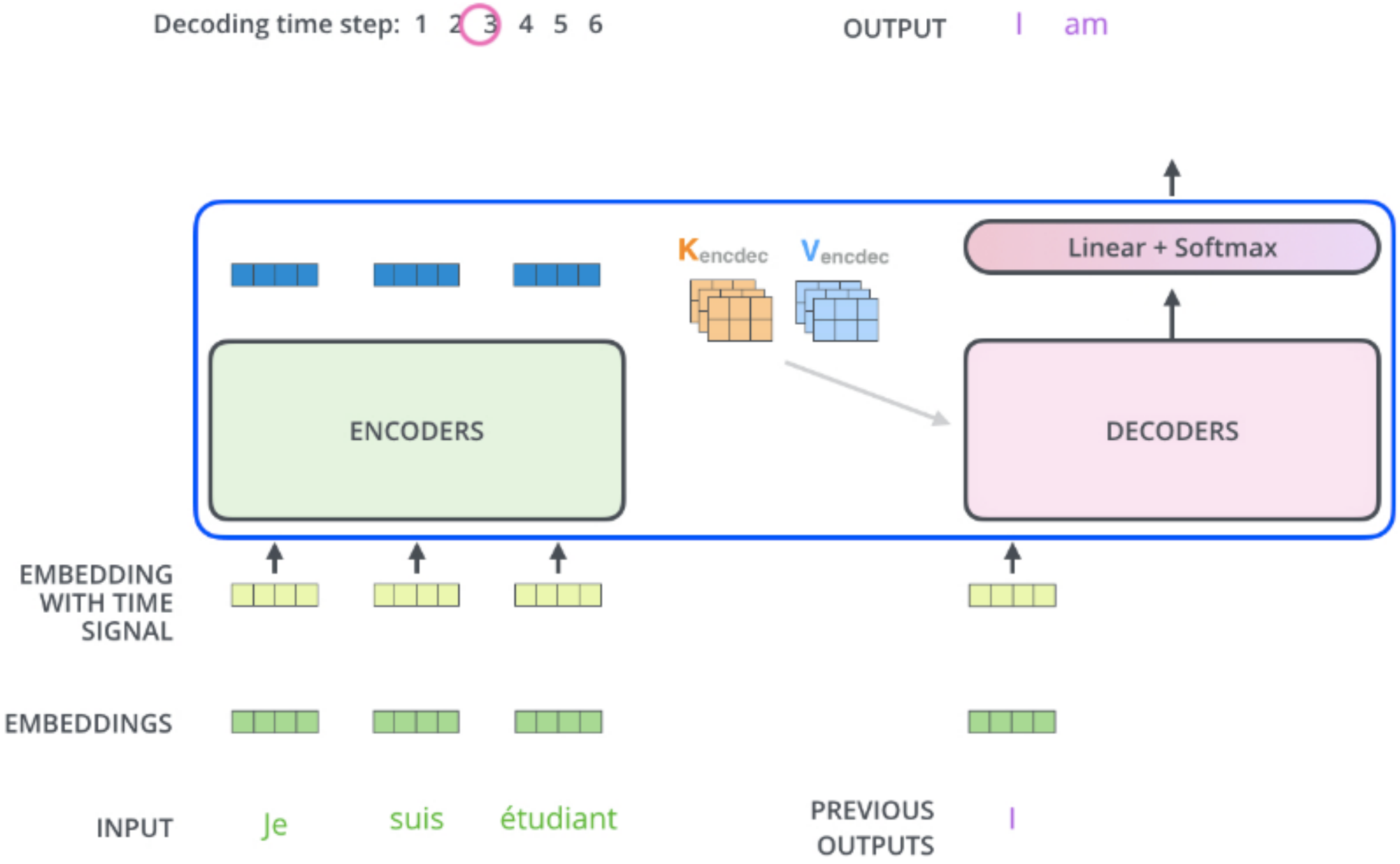
Get the index of the cell with the highest value (**argmax**)

5



Decoder stack output

How it works



But what about Self-attention?

Training and the Loss Function

Untrained Model Output

0.2	0.2	0.1	0.2	0.2	0.1

Correct and desired output

0.0	0.0	0.0	1.0	0.0	0.0
a	am	I	thanks	student	<eos>

We can use cross Entropy.

We can also optimize two words at a time: using BEAM search: keep a few alternatives for the first word.

Cross Entropy and KL (Kullback-Leibler) divergence

- Entropy: $E(P) = -\Sigma_i P(i) \log P(i)$ - expected prefix-free code length (also optimal)
- Cross Entropy: $C(P) = -\Sigma_i P(i) \log Q(i)$ – expected coding length using optimal code for Q
- KL divergence:

  $D_{KL}(P \,||\, Q) = \Sigma_i P(i) \log[P(i)/Q(i)] = \Sigma_i P(i)[\log P(i) - \log Q(i)]$, extra bits to code using Q rather than P

- JSD$(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$, $M = \frac{1}{2}(P+Q)$, symmetric KL

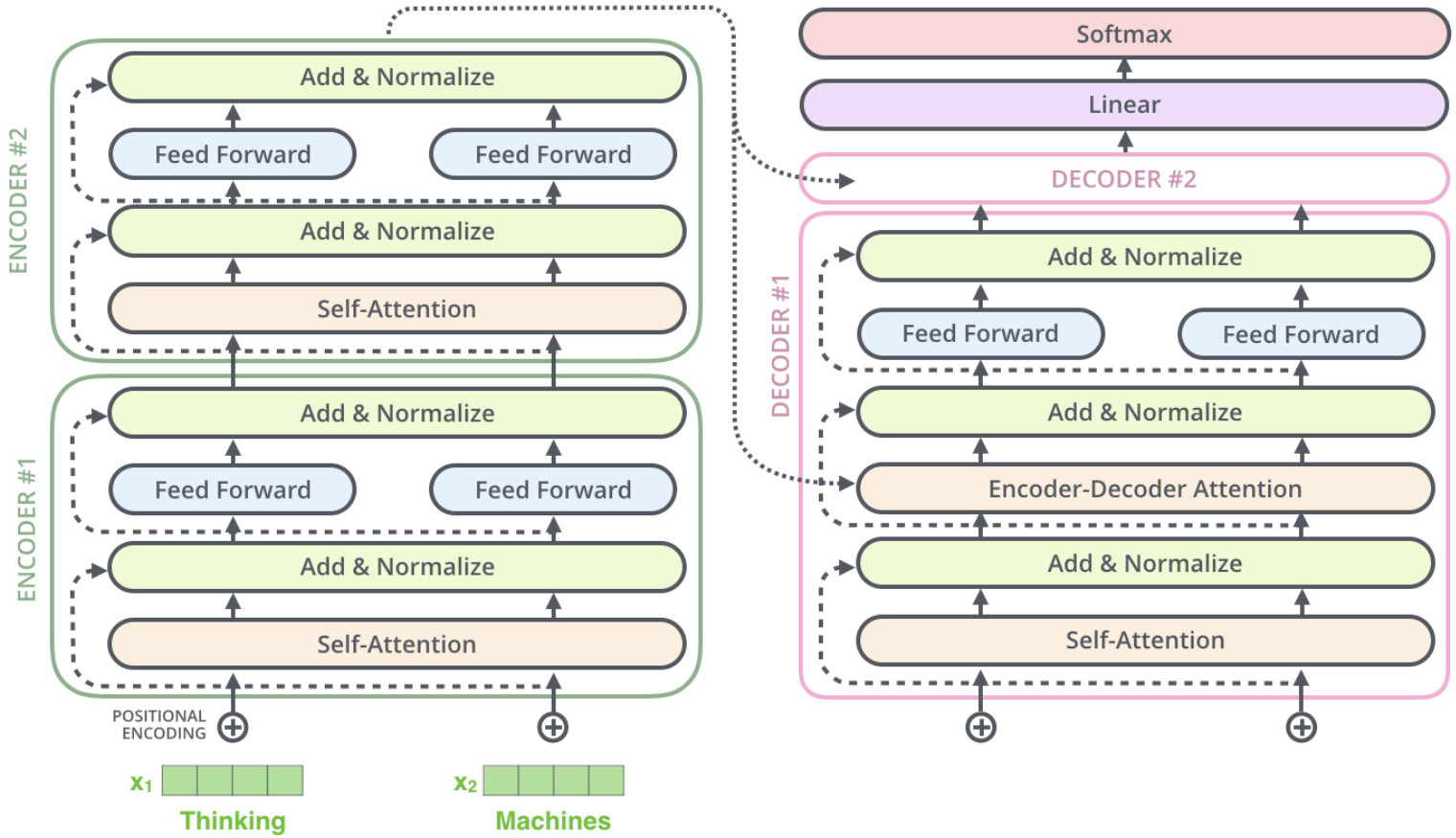
* JSD = Jensen-Shannon Divergency

Transformer Results

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 [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	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 [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	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	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Next Lecture: Pretraining



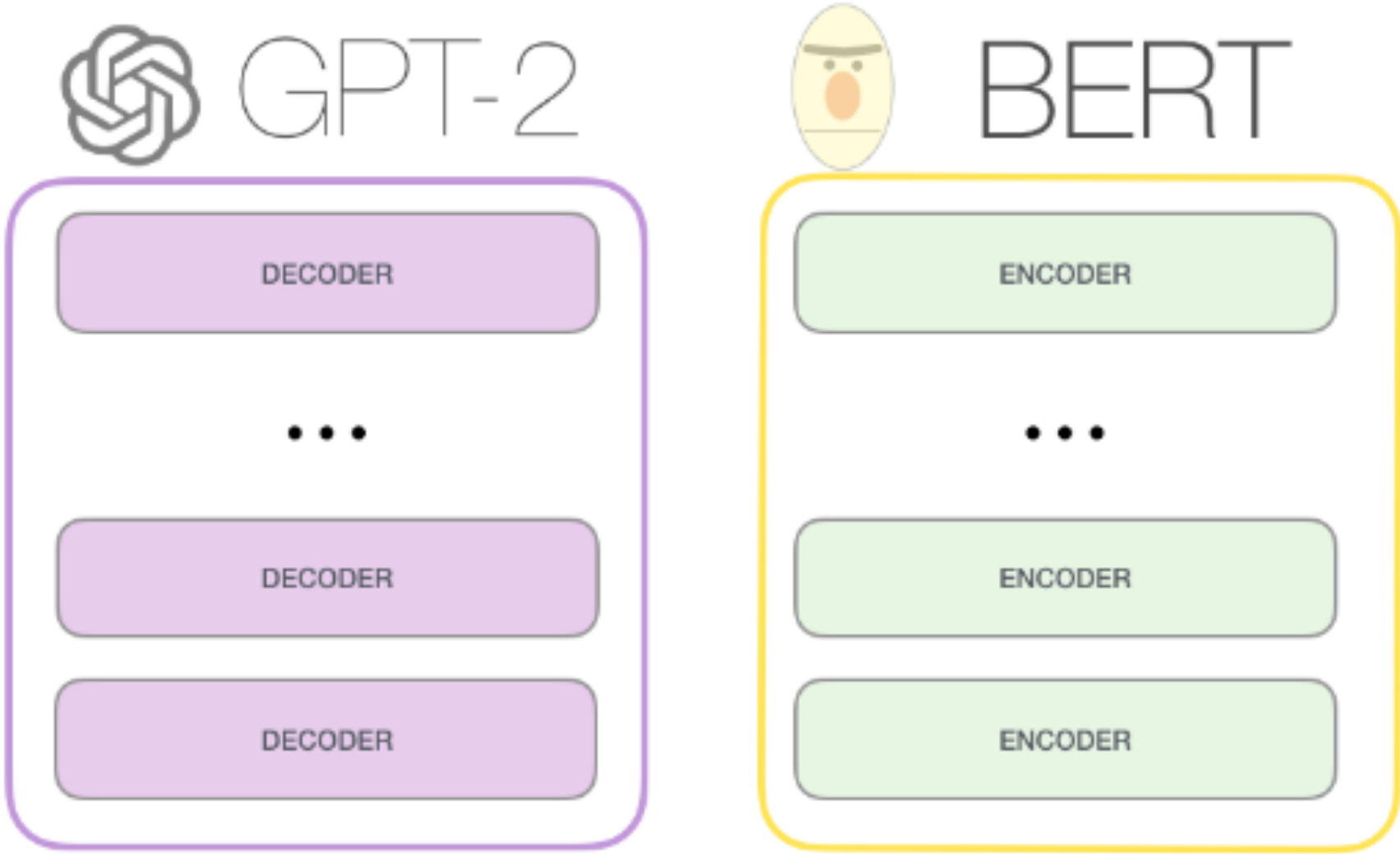
BERT

GPT

Transformers, GPT-2, and BERT

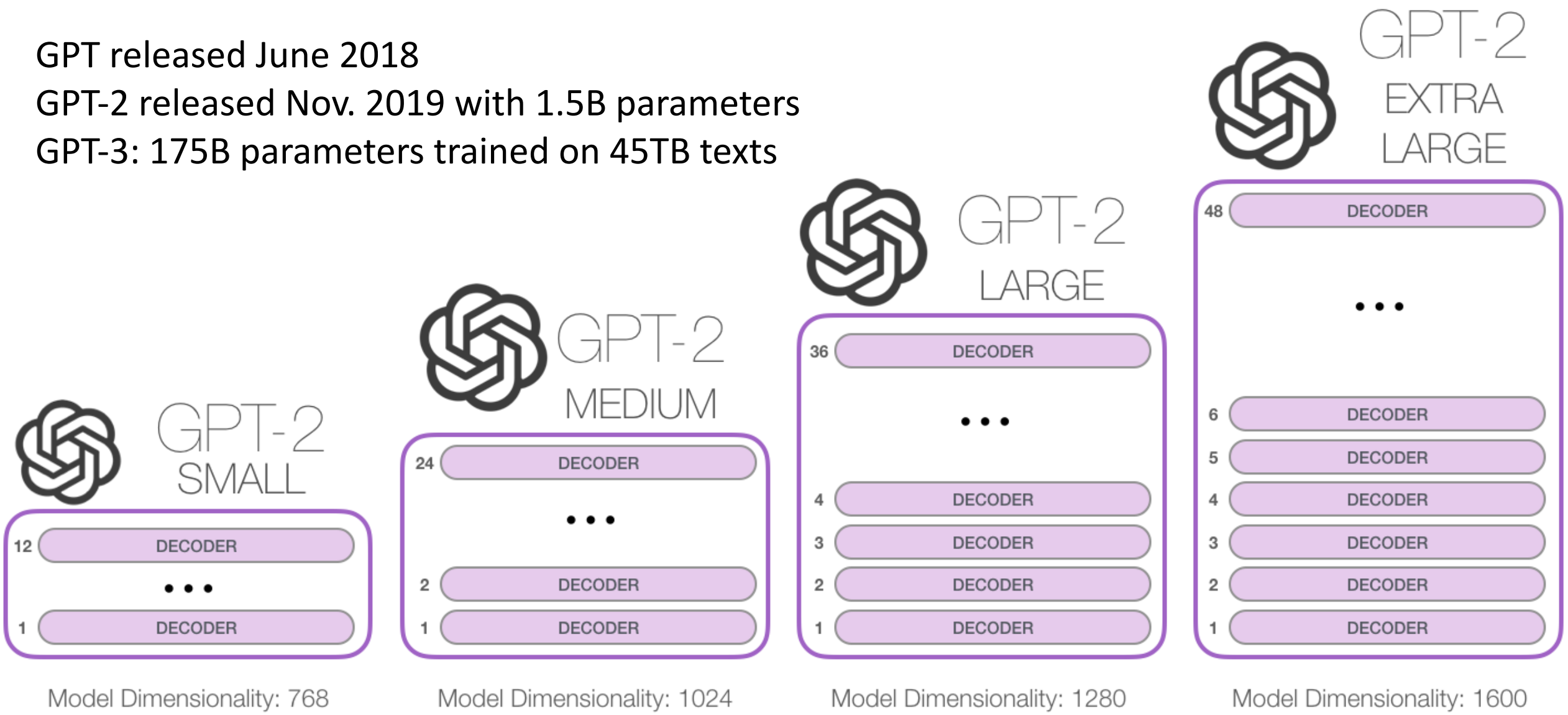
1. A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
2. But if we do not have input, we just want to model the "next word", we can get rid of the Encoder side of a transformer and output "next word" one by one. This gives us GPT.
3. If we are only interested in training a language model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.

GPT-2, BERT

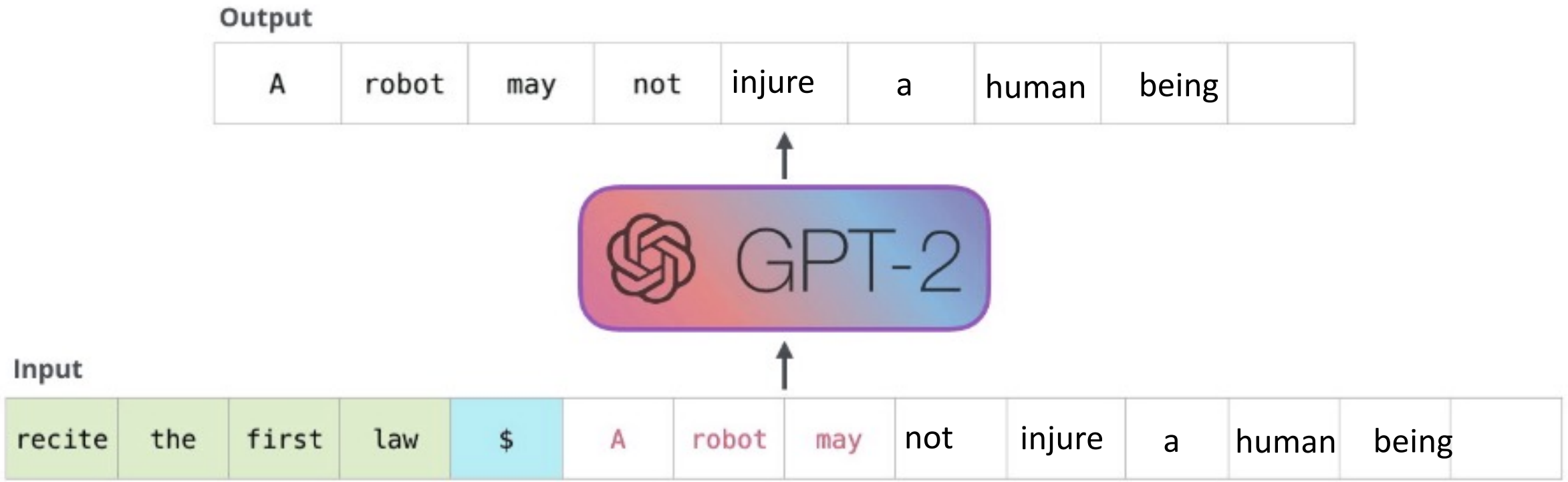

GPT-2
DECODER
DECODER
DECODER
BERT
ENCODER
ENCODER
ENCODER

GPT-2, BERT

GPT released June 2018
GPT-2 released Nov. 2019 with 1.5B parameters
GPT-3: 175B parameters trained on 45TB texts



GPT-2 in action



Byte Pair Encoding (BPE)

Word embedding sometimes is too high level, pure character embedding too low level. For example, if we have learned

old older oldest

We might also wish the computer to infer

smart smarter smartest

But at the whole word level, this might not be so direct. Thus the idea is to break the words up into pieces like er, est, and embed frequent fragments of words.

GPT adapts this BPE scheme.

Byte Pair Encoding (BPE)

GPT uses BPE scheme. The subwords are calculated by:

1. Split word to sequence of characters (add </w> char)
2. Joining the highest frequency pattern.
3. Keep doing step 2, until it hits the pre-defined maximum number of sub-words or iterations.

Example (5, 2, 6, 3 are number of occurrences)

{‘l o w </w>’: 5, ‘l o w e r </w>’: 2, ‘n e w e s t </w>’: 6, ‘w i d e s t </w>’: 3 }

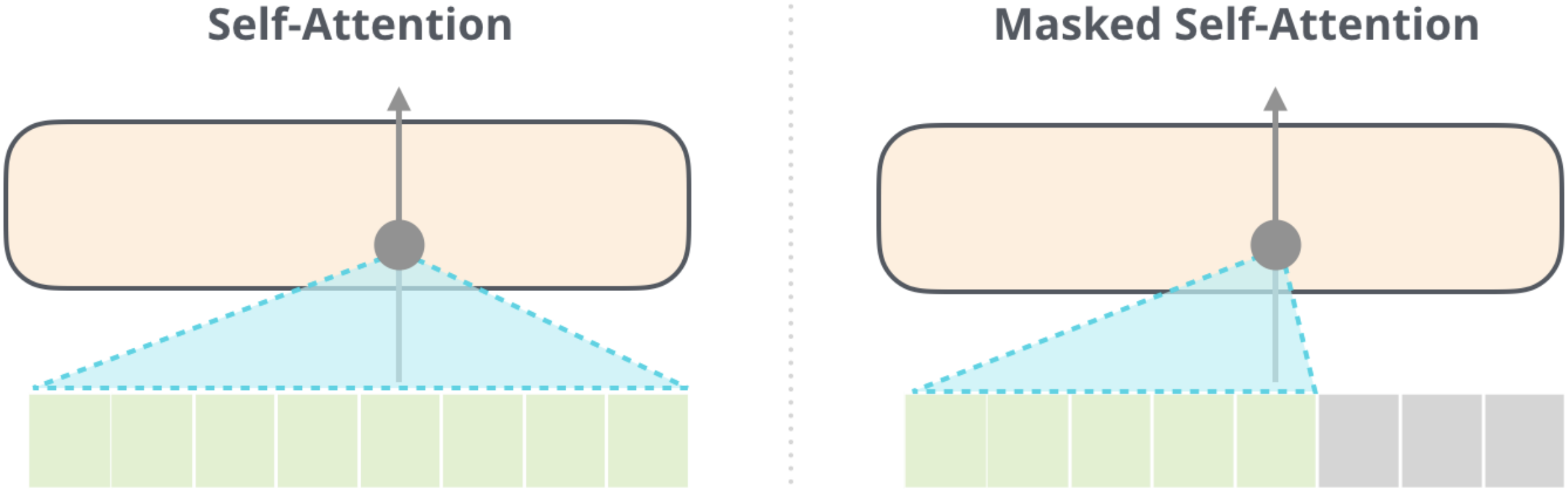
{‘l o w </w>’: 5, ‘l o w e r </w>’: 2, ‘n e w es t </w>’: 6, ‘w i d es t </w>’: 3 }

{‘l o w </w>’: 5, ‘l o w e r </w>’: 2, ‘n e w est </w>’: 6, ‘w i d est </w>’: 3 } (est freq. 9)

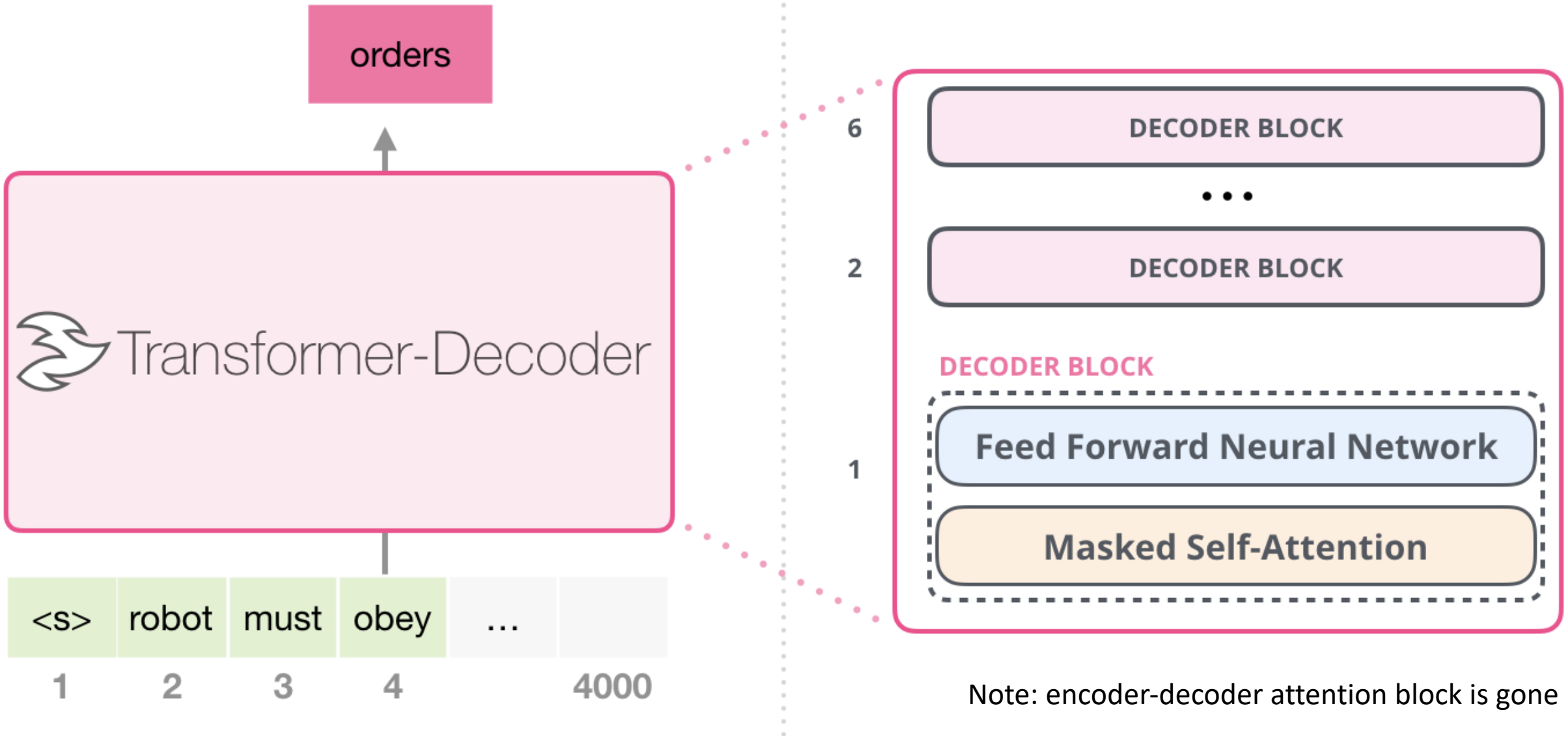
{‘lo w </w>’: 5, ‘lo w e r </w>’: 2, ‘n e w est</w>’: 6, ‘w i d est</w>’: 3 } (lo freq 7)

…..

Masked Self-Attention (to compute more efficiently)



Masked Self-Attention



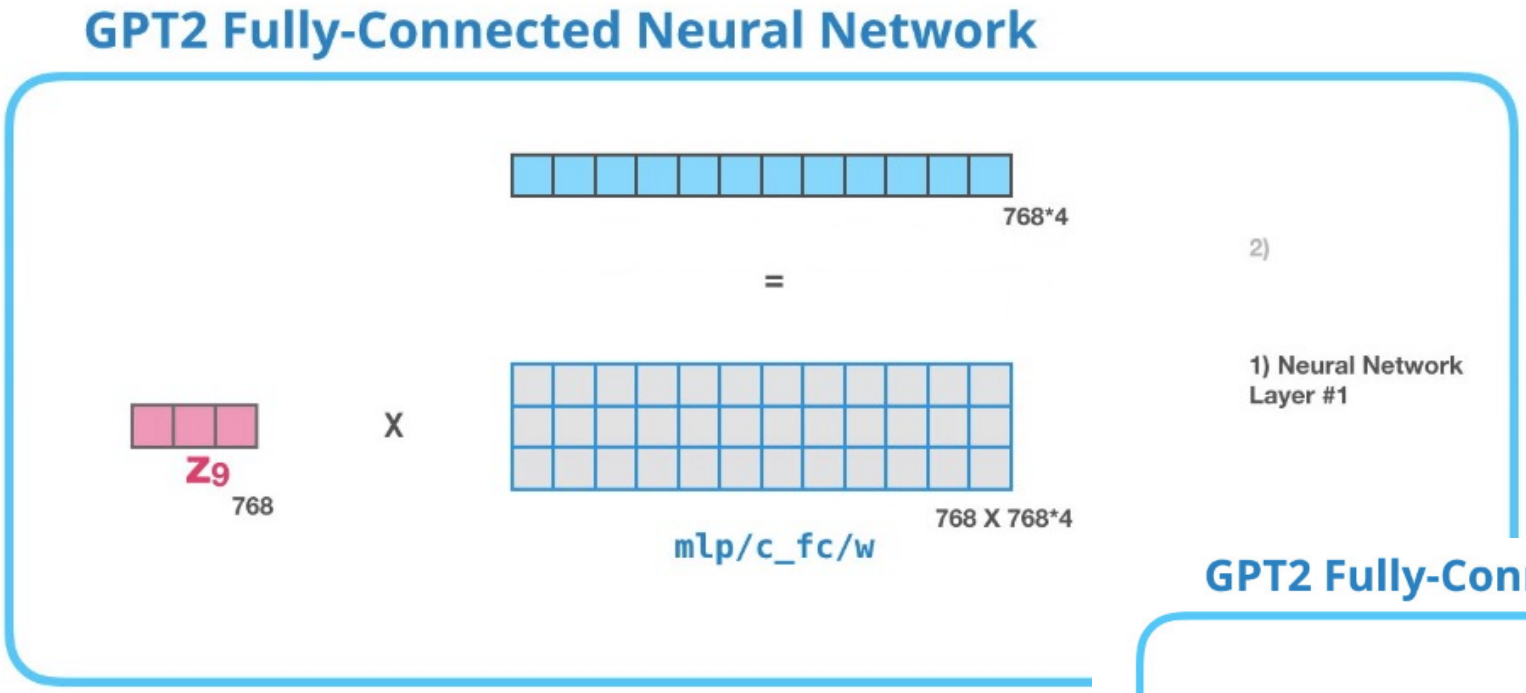
Note: encoder-decoder attention block is gone

Masked Self-Attention Calculation

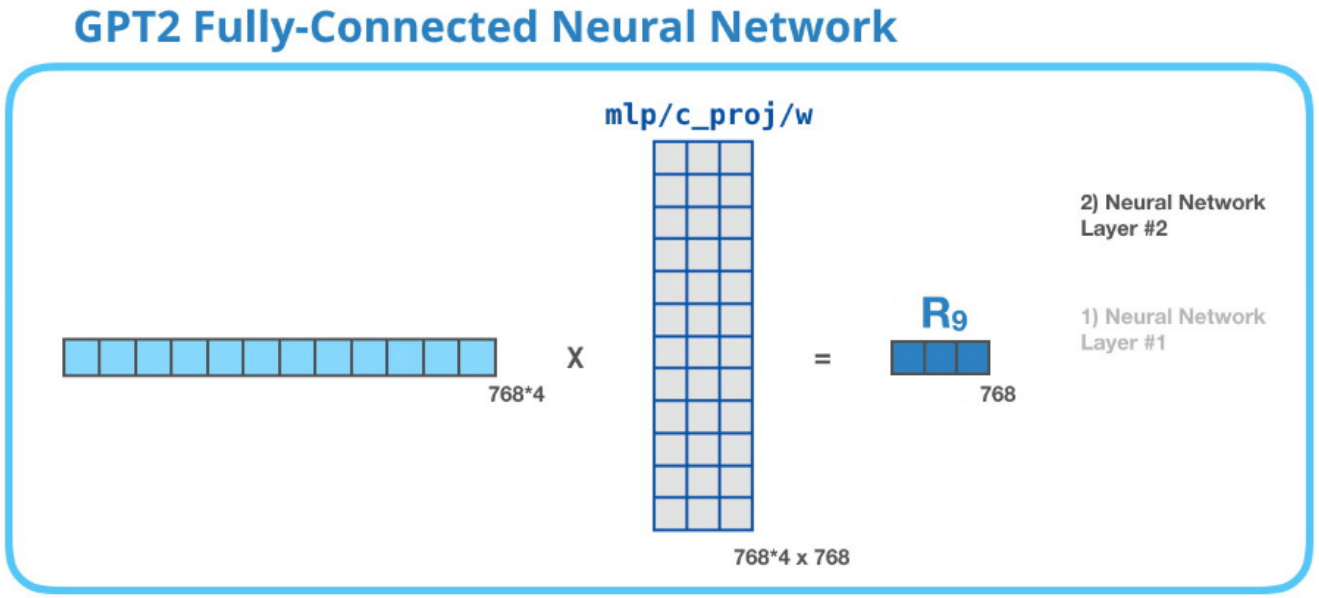
Re-use previous computation results: at any step, only need to results of q, k , v related to the new output word, no need to re-compute the others. Additional computation is linear, instead of quadratic.

Note: encoder-decoder attention block is gone

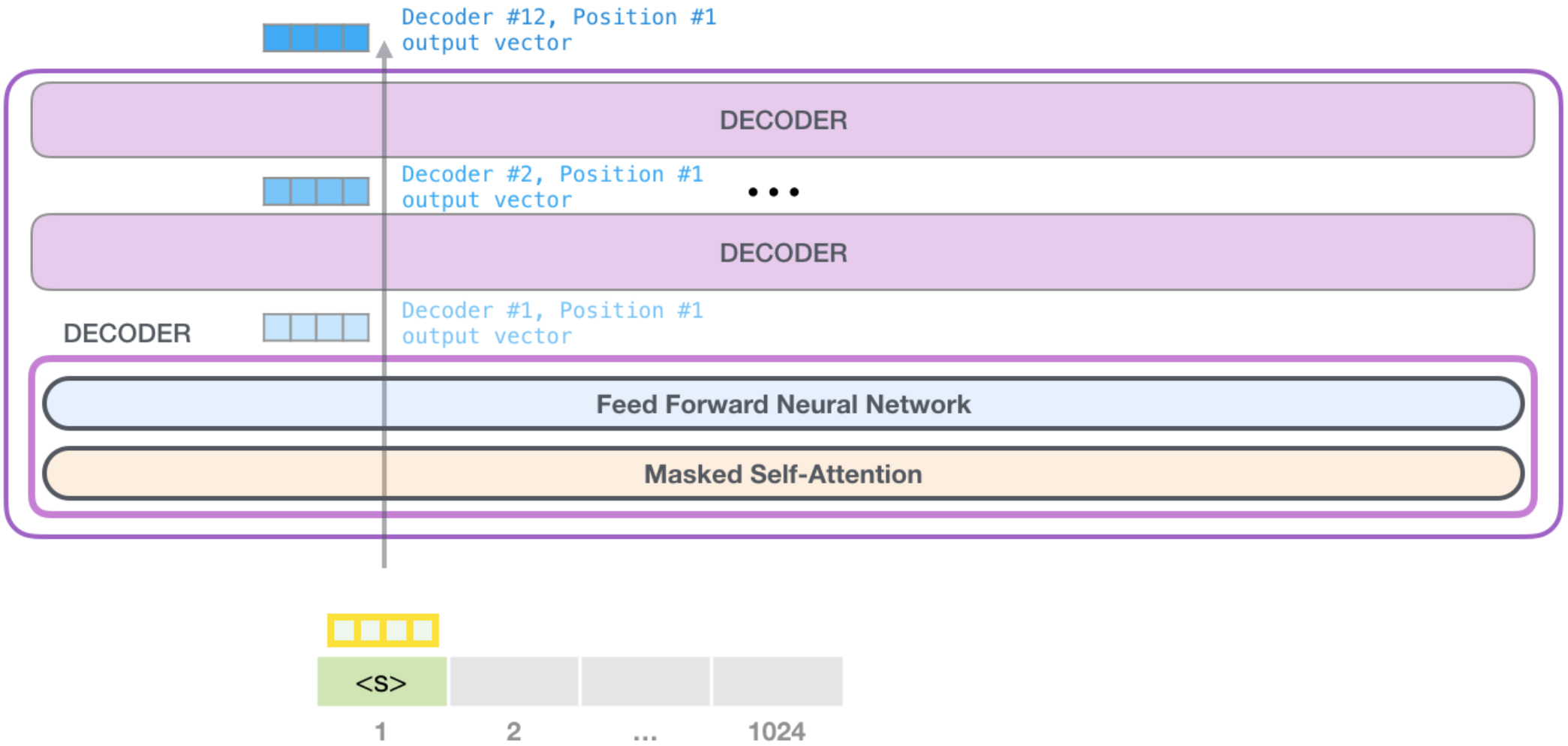
GPT-2 fully connected network has two layers (Example for GPT-2



768 is small model size



GPT-2 has a parameter top-k, so that we sample words from top k (highest probability from softmax) words for each output



This top-k parameter, if k=1, we would have output like:

The first time I saw the new version of the game, I was so excited. I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of

GPT Training

GPT-2 uses unsupervised learning approach to training the language model.

There is no custom training for GPT-2, no separation of pre-training and fine-tuning like BERT.

A story generated by GPT-2

“The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

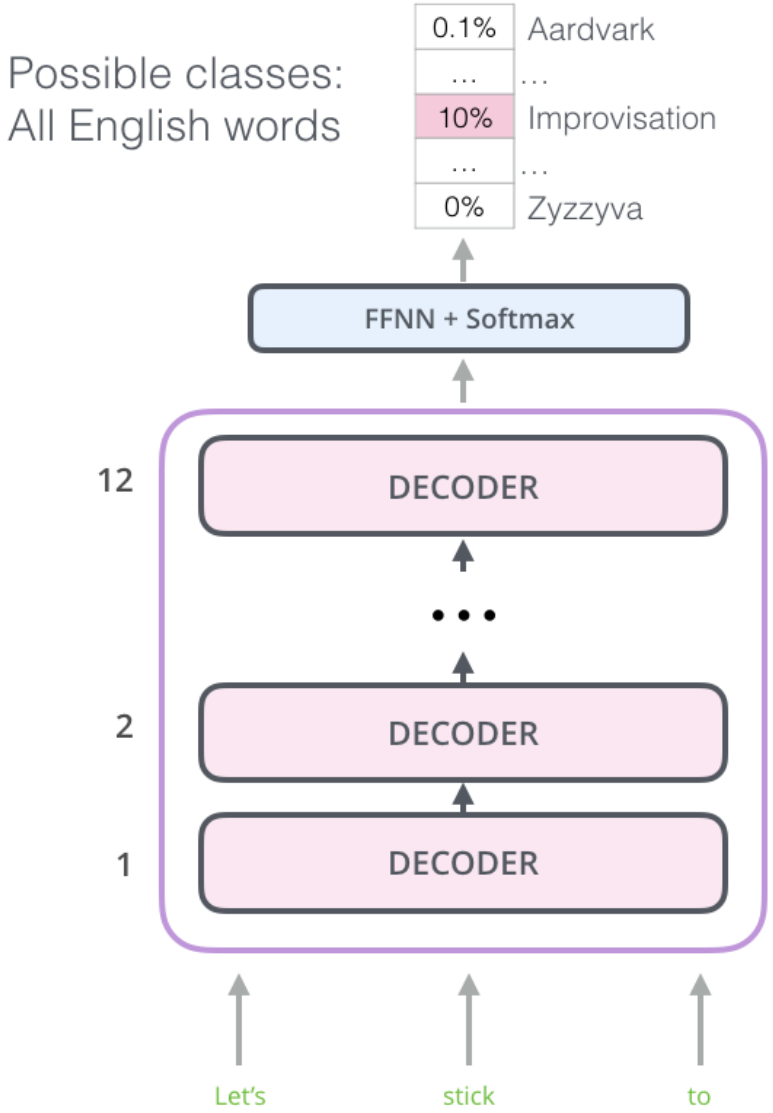
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. `By the time we reached the top of one peak, the water looked blue, with some crystals on top,’ said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns."

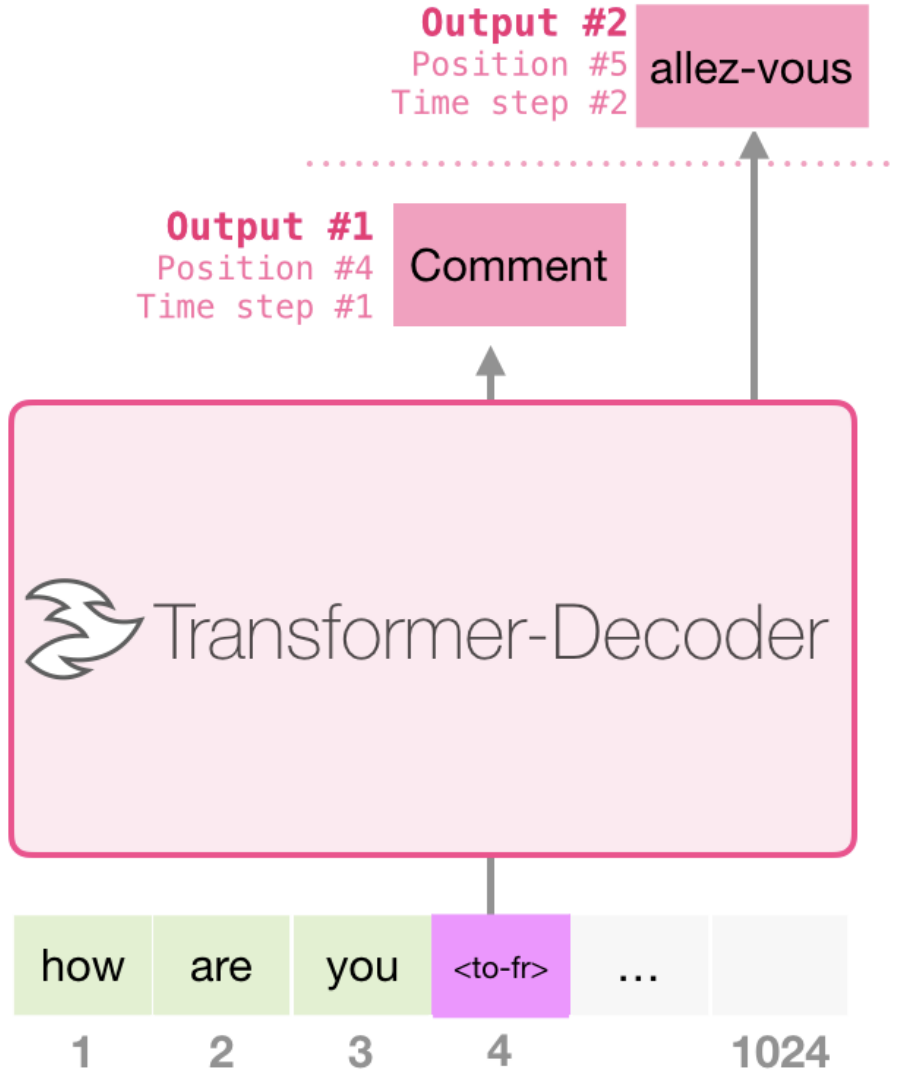
Transformer / GPT prediction



GPT-2 Application: Translation

Training Dataset

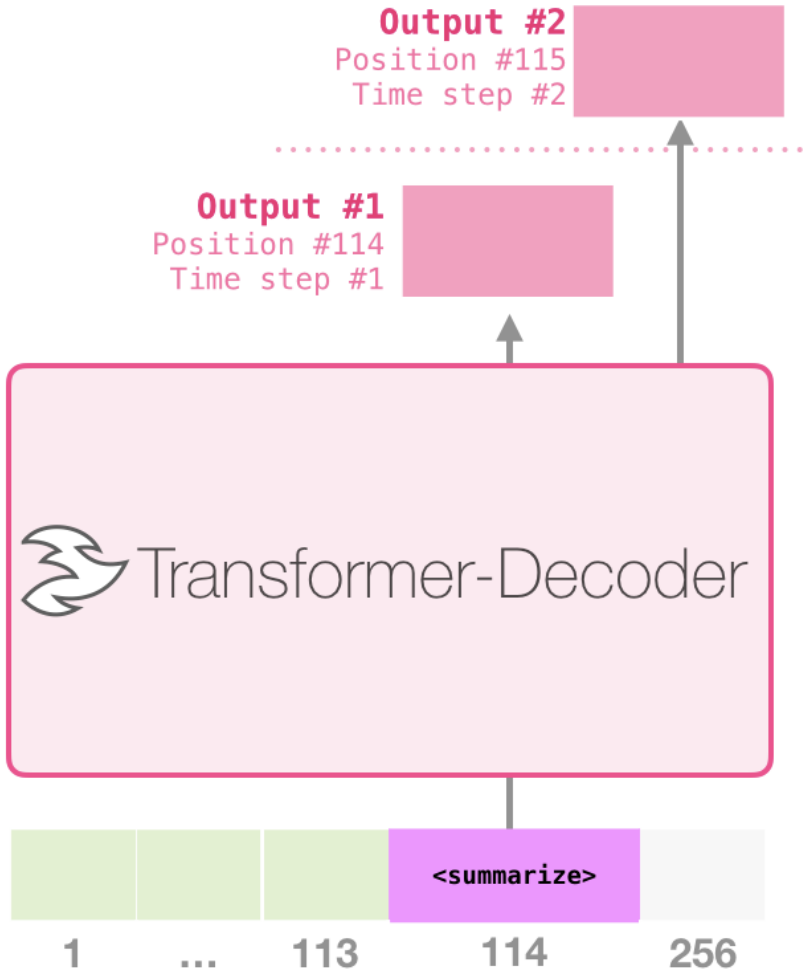
I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



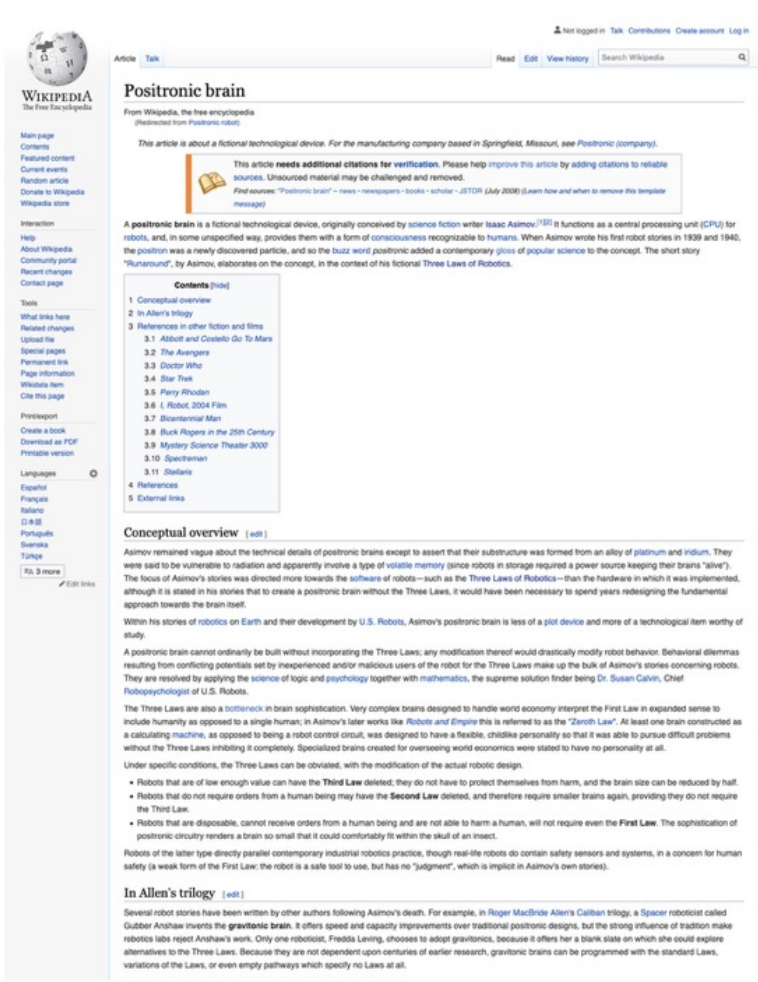
GPT-2 Application: Summarization

Training Dataset

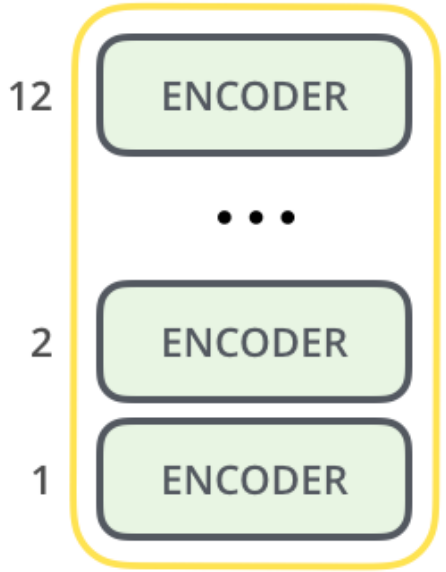
Article #1 tokens	<summarize>	Article #1 Summary	
Article #2 tokens	<summarize>	Article #2 Summary	padding
Article #3 tokens	<summarize>	Article #3 Summary	



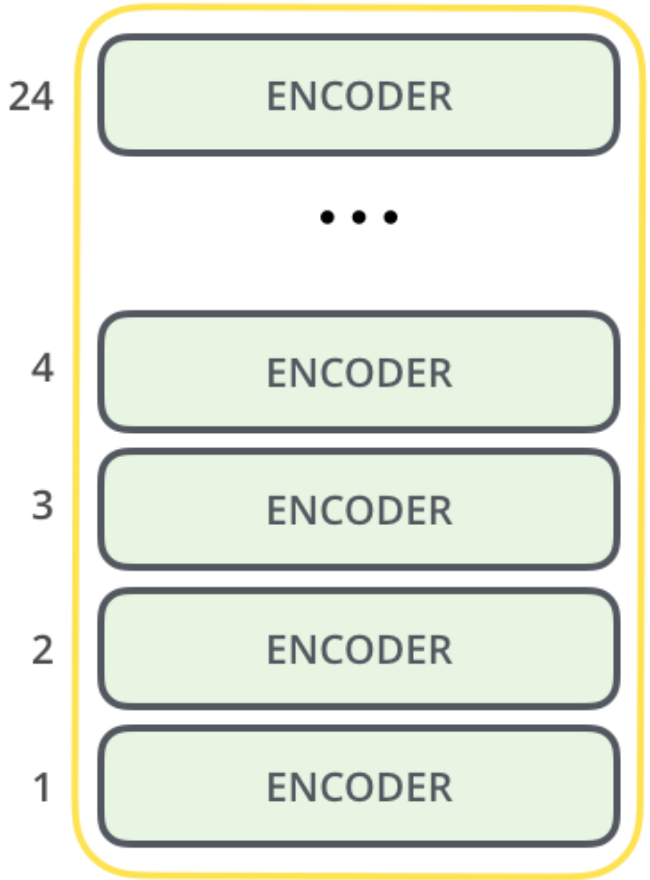
Using wikipedia data



BERT (Bidirectional Encoder Representation from Transformers)



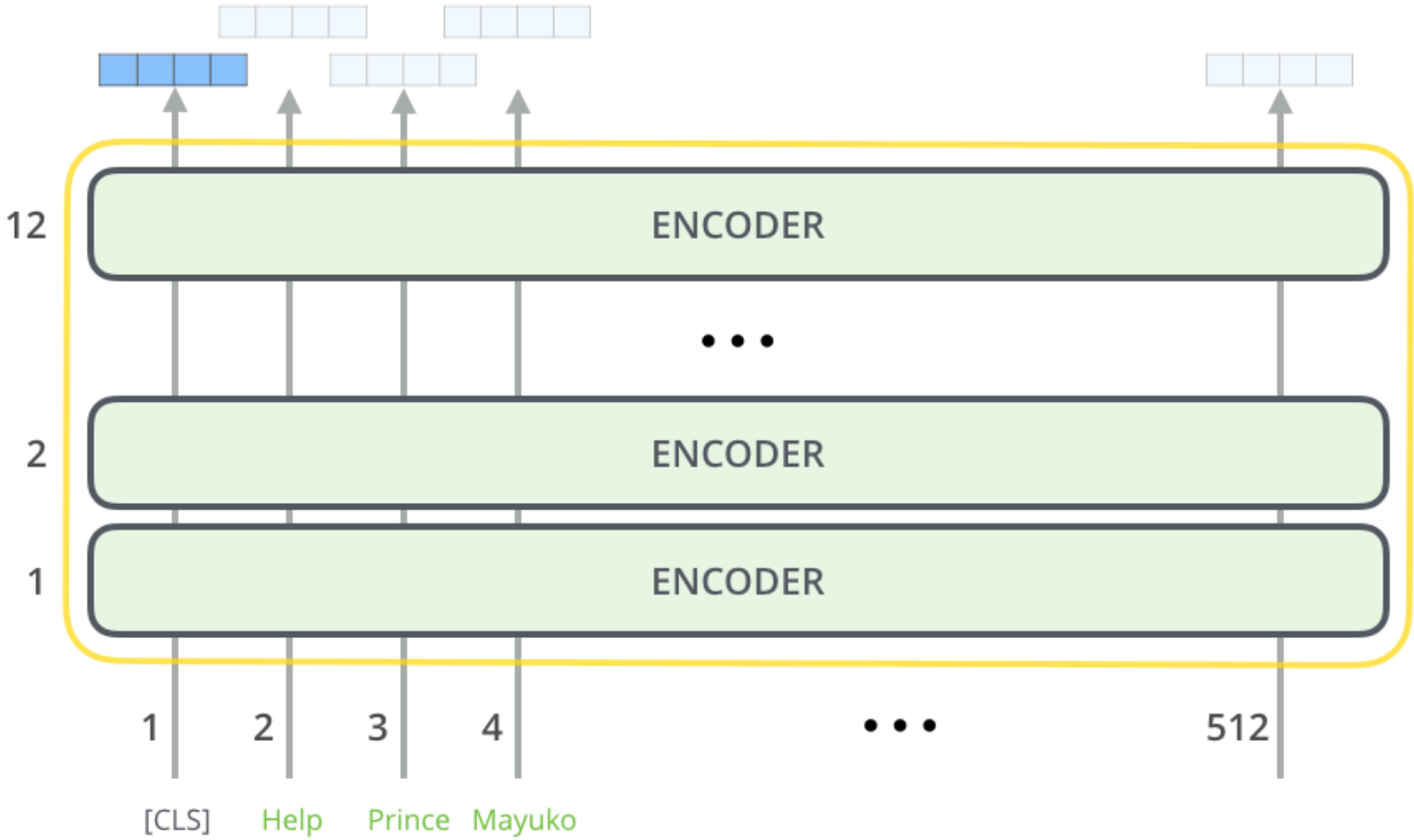
$BERT_{BASE}$



$BERT_{LARGE}$

Model input dimension 512

Input and output vector size

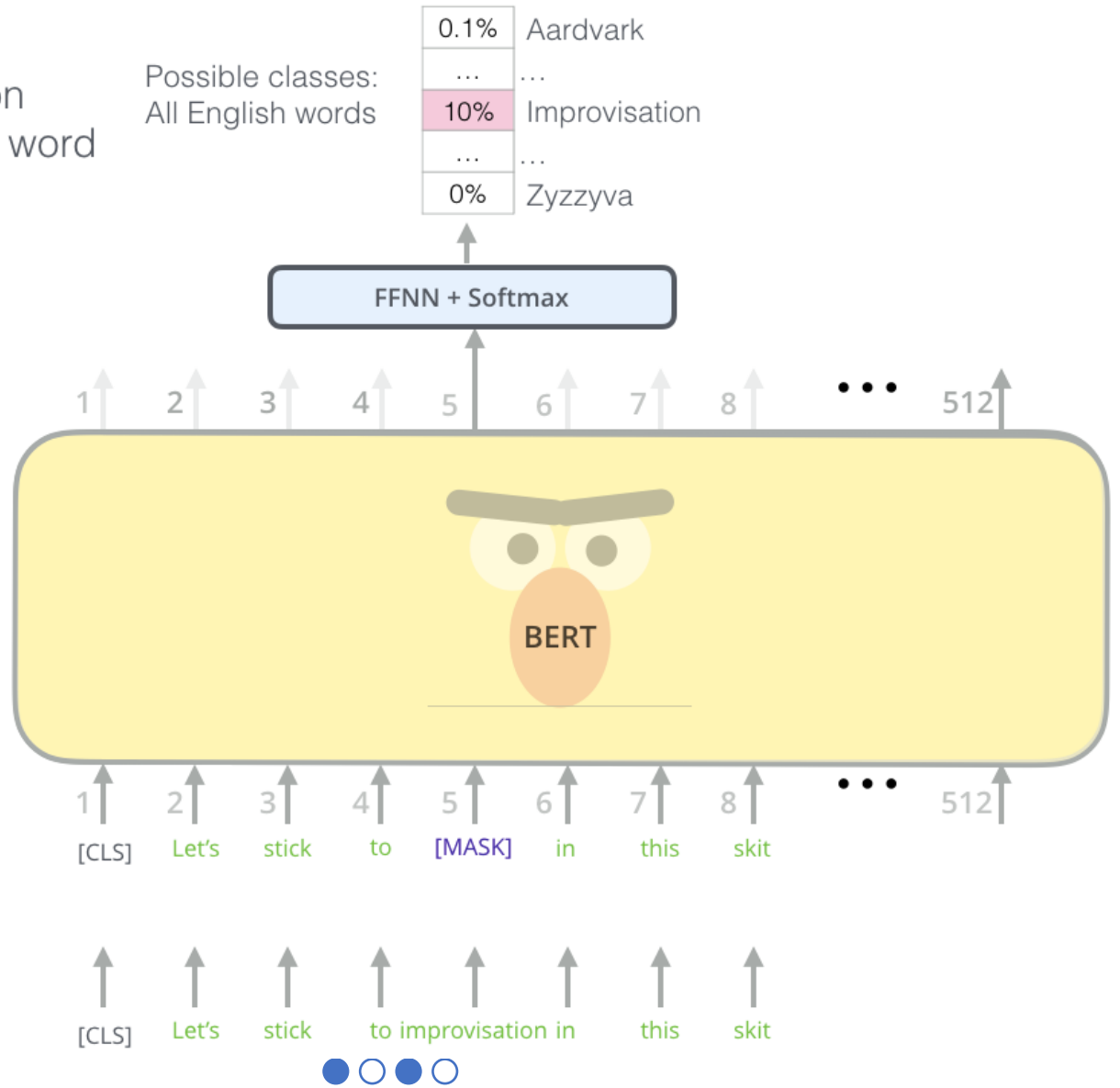


BERT pretraining

ULM-FiT (2018): Pre-training ideas, transfer learning in NLP.
ELMo: Bidirectional training (LSTM)
Transformer: Although used things from left, but still missing from the right.
GPT: Use Transformer Decoder half.
BERT: Switches from Decoder to Encoder, so that it can use both sides in training and invented corresponding training tasks: masked language model

BERT Pretraining Task 1: masked words

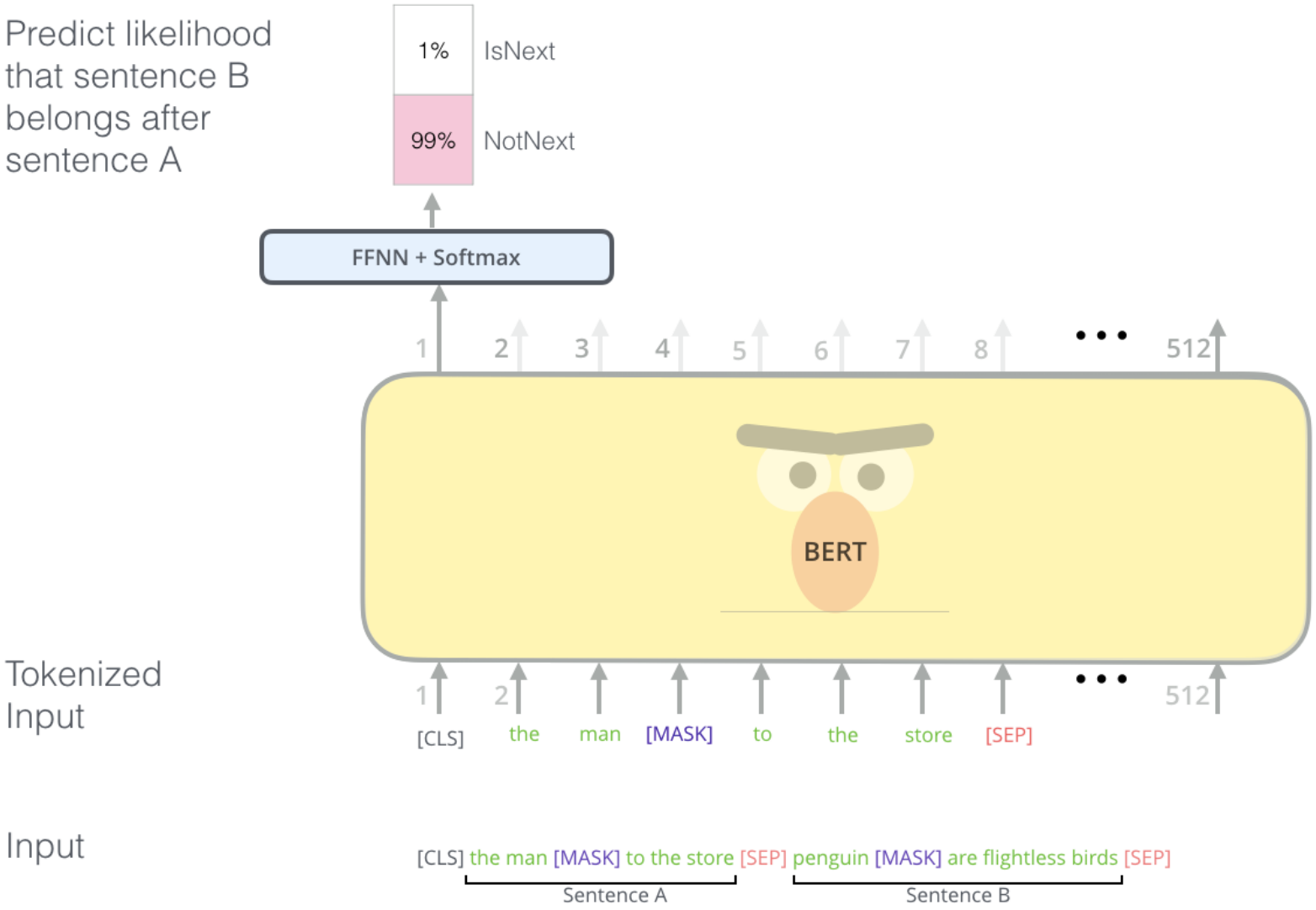
Use the output of the masked word's position to predict the masked word



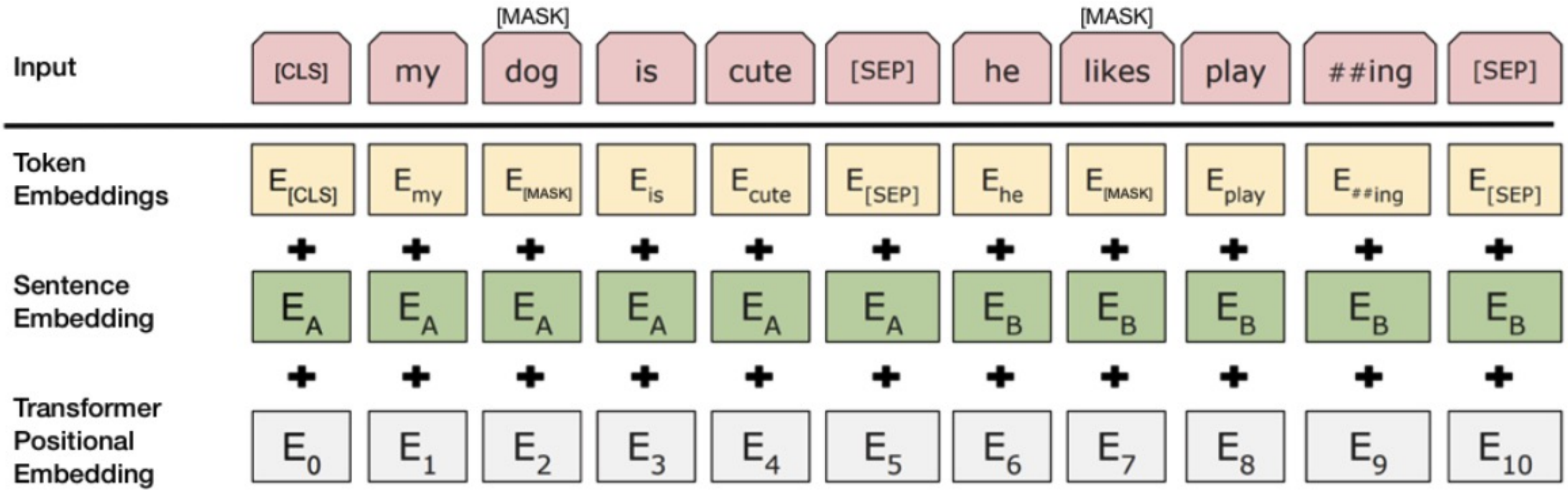
Out of this 15%,
80% are [Mask],
10% random words
10% original words

Randomly mask 15% of tokens

BERT Pretraining Task 2: two sentences



BERT Pretraining Task 2: two sentences

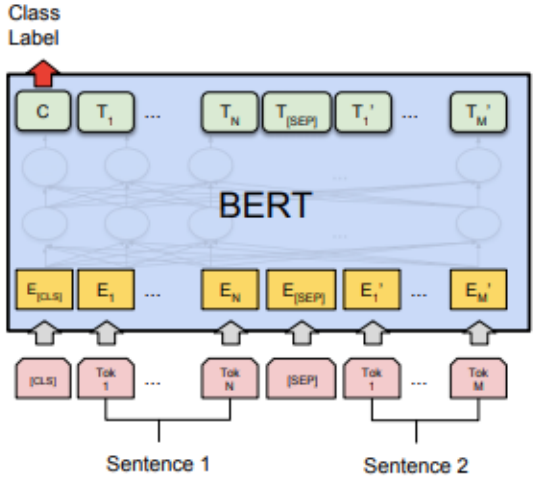


50% true second sentences

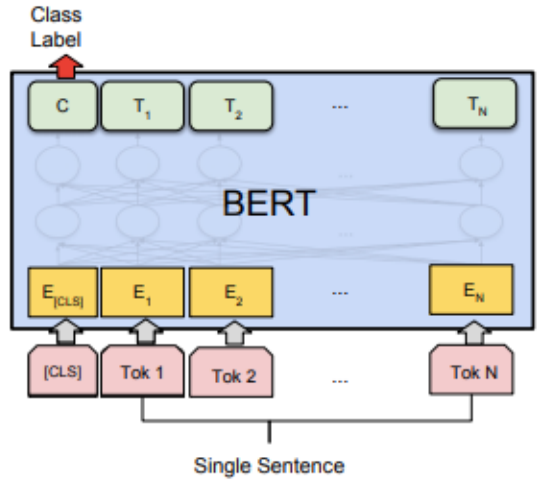
50% random second sentences

Fine-tuning BERT for other specific tasks

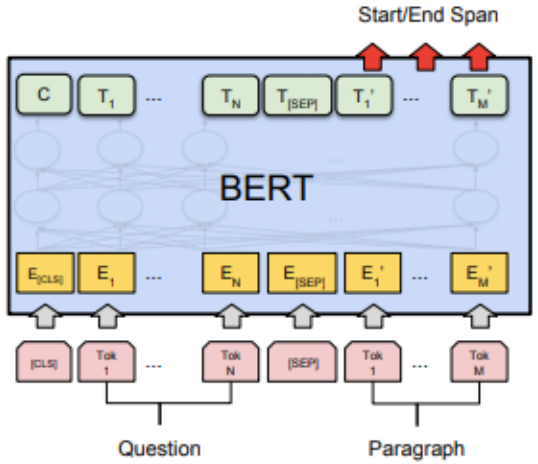
MNLI
QQP (Quaro Question Pairs)
Semantic equivalence)
QNLI (NL inference dataset)
STS-B (texture similarity)
MRPC (paraphrase, Microsoft)
RTE (textual entailment)
SWAG (commonsense inference)
SST-2 (sentiment)
CoLA (linguistic acceptability
SQuAD (question and answer)



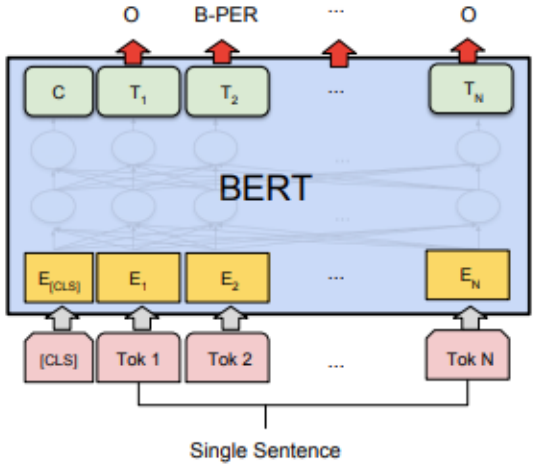
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

SST (Stanford sentiment treebank): 215k phrases with fine-grained sentiment labels in the parse trees of 11k sentences.

NLP Tasks: Multi-Genre Natural Lang. Inference

MNLI: 433k pairs of examples, labeled by entailment, neutral or contraction

Met my first girlfriend that way.	FACE-TO-FACE **contradiction** C C N C	I didn't meet my first girlfriend until later.
8 million in relief in the form of emergency housing.	GOVERNMENT **neutral** N N N N	The 8 million dollars for emergency housing was still not enough to solve the problem.
Now, as children tend their gardens, they have a new appreciation of their relationship to the land, their cultural heritage, and their community.	LETTERS **neutral** N N N N	All of the children love working in their gardens.
At 8:34, the Boston Center controller received a third transmission from American 11	9/11 **entailment** E E E E	The Boston Center controller got a third transmission from American 11.
I am a lacto-vegetarian.	SLATE **neutral** N N E N	I enjoy eating cheese too much to abstain from dairy.
someone else noticed it and i said well i guess that's true and it was somewhat melodious in other words it wasn't just you know it was really funny	TELEPHONE **contradiction** C C C C	No one noticed and it wasn't funny at all.

Table 1: Randomly chosen examples from the development set of our new corpus, shown with their genre labels, their selected gold labels, and the validation labels (abbreviated E, N, C) assigned by individual annotators.

NLP Tasks (SQuAD -- Stanford Question Answering Dataset):

Sample: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Which NFL team represented the AFC at Super Bowl 50?

Ground Truth Answers: Denver Broncos

Which NFL team represented the NFC at Super Bowl 50?

Ground Truth Answers: Carolina Panthers

Add indices for sentences and paragraphs

SegaTron/SegaBERT

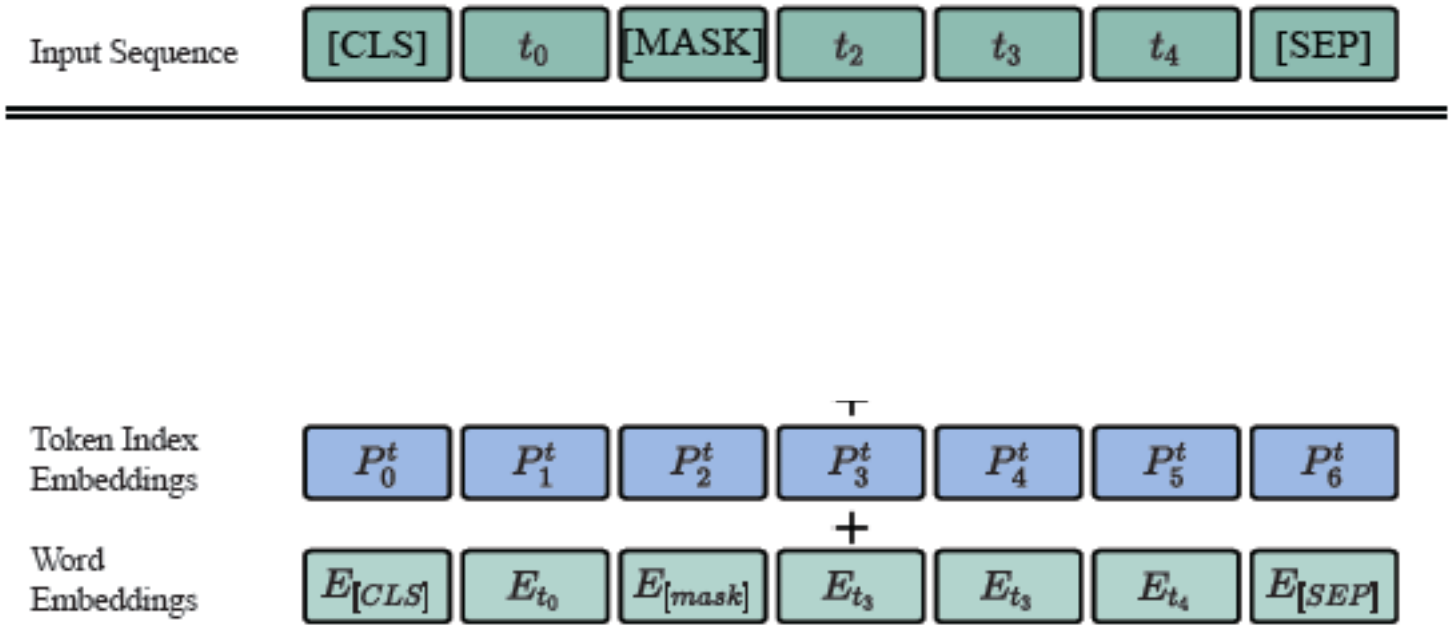
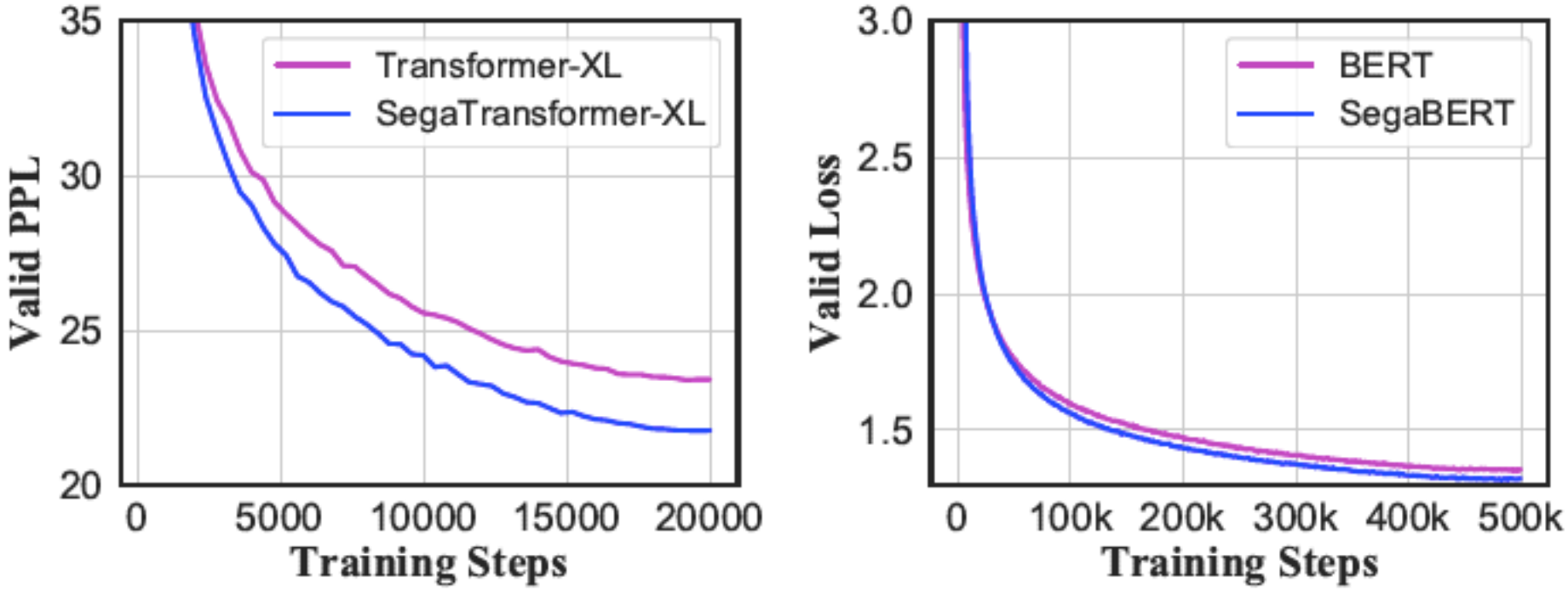


Figure 1: Input Representation of SegaBERT

H. Bai, S. Peng, J. Lin, L. Tan, K. Xiong, W. Gao, M. Li: SgaTron: Segment-aware transformer for language modeling and understanding. AAAI'2021

Conversion speed much faster ：



(a) Language modeling perplexities

(b) Pre-training losses

Figure 2: Valid perplexities and losses during the training processes of language modeling and pre-training.

Testing on GLUE dataset

Task(Metrics)	BASE model(wikipedia 500K steps)				LARGE model(wikibooks 1000K steps)			
	dev		test		dev		test	
	BERT	SegaBERT	BERT	SegaBERT	BERT	SegaBERT	BERT	SegaBERT
CoLA (Matthew Corr.)	**55.0**	54.7	43.5	**50.7**	60.6	**65.3**	60.5	**62.6**
SST-2 (Acc.)	91.3	**92.1**	91.2	**91.5**	93.2	**94.7**	**94.9**	94.8
MRPC (F1)	**92.6**	92.4	88.9	**89.3**	-	92.3	89.3	**89.7**
STS-B (Spearman Corr.)	88.9	**89.0**	83.9	**84.6**	-	90.3	86.5	**88.6**
QQP (F1)	86.5	**87.0**	70.8	**71.4**	-	89.1	72.1	**72.5**
MNLI-m (Acc.)	83.2	**83.8**	82.9	**83.5**	86.6	**87.6**	86.7	**87.9**
MNLI-mm (Acc.)	83.4	**84.1**	82.8	**83.2**	-	87.5	85.9	**87.7**
QNLI (Acc.)	90.4	**91.5**	90.1	**90.8**	92.3	**93.6**	92.7	**94.0**
RTE (Acc.)	68.3	**71.8**	65.4	**68.1**	70.4	**78.3**	70.1	**71.6**
Average	82.2	**82.9**	77.7	**79.2**	-	86.5	82.1	**83.3**

Table 2: The results on GLUE benchmark. All base models are pre-trained by this work. Every result of the dev set is the average score of 4 times finetuning with different random seeds. Scores of BERT large dev are from (Sun et al., 2019) and scores of BERT large test are from (Devlin et al., 2018).

H. Bai, S. Peng, J. Lin, L. Tan, K. Xiong, W. Gao, M. Li: SgaTron: Segment-aware transformer for language modeling and understanding. AAAI'2021

Reading comprehesion – SQUAD tasks

System	Dev	
	EM	F1
BERT base (Single)	80.8	88.5
BERT large (Single	84.1	90.9
BERT large (Single+DA)	84.2	91.1
KT-NET	85.2	91.7
StructBERT Large (Single)	85.2	92.0
SegaBERT base (Single)	83.2	90.2
SegaBERT large (Single)	85.3	92.4

Table 3: Evaluation results of SQUAD v1.1.

System	Dev	
	EM	F1
BERT base	72.3	75.6
BERT base (ours)	75.4	78.2
SegaBERT base	76.3	79.2
BERT large	78.7	81.9
BERT large wwm	80.6	83.4
SegaBERT large	81.8	85.2

Table 4: Evaluation results of SQUAD v2.0.

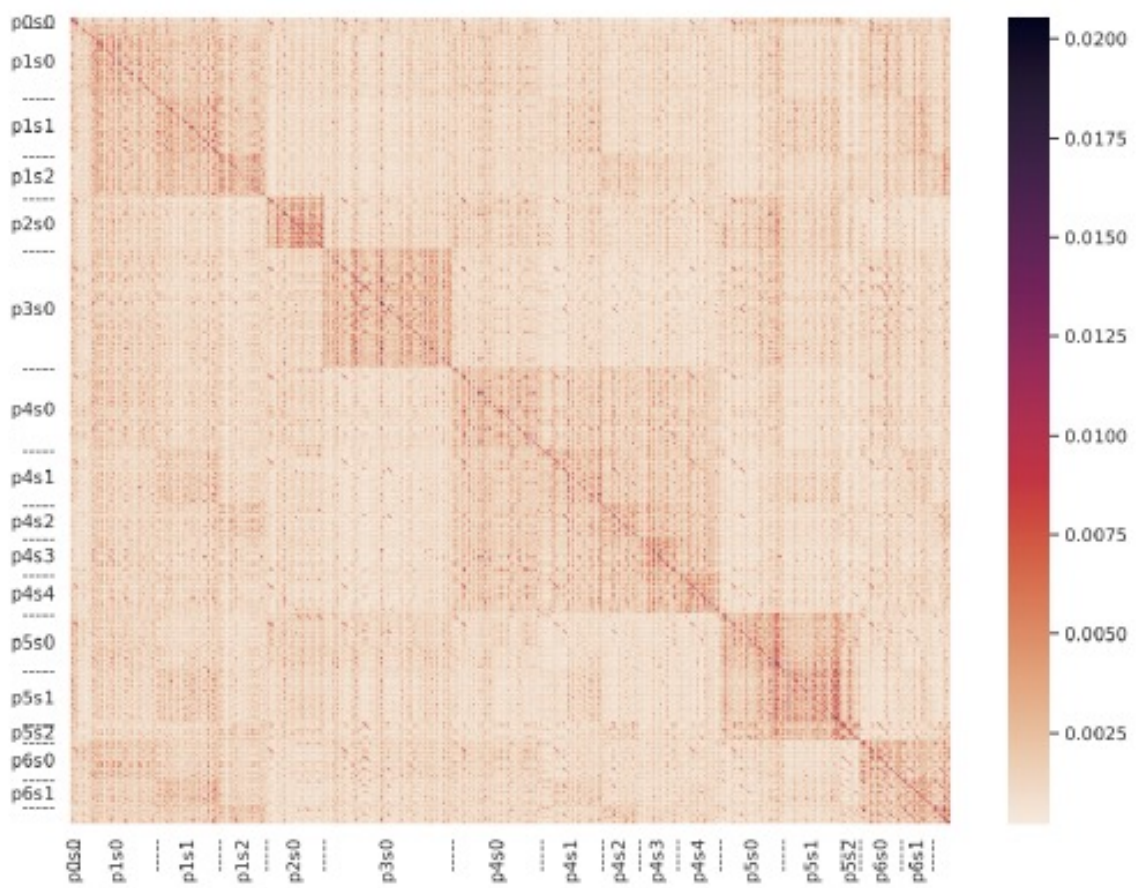
F1 = 2 (P*R) / (P+R), P is precision, R is recall, all in percentage, EM – exact match

Improving Transformer-XL

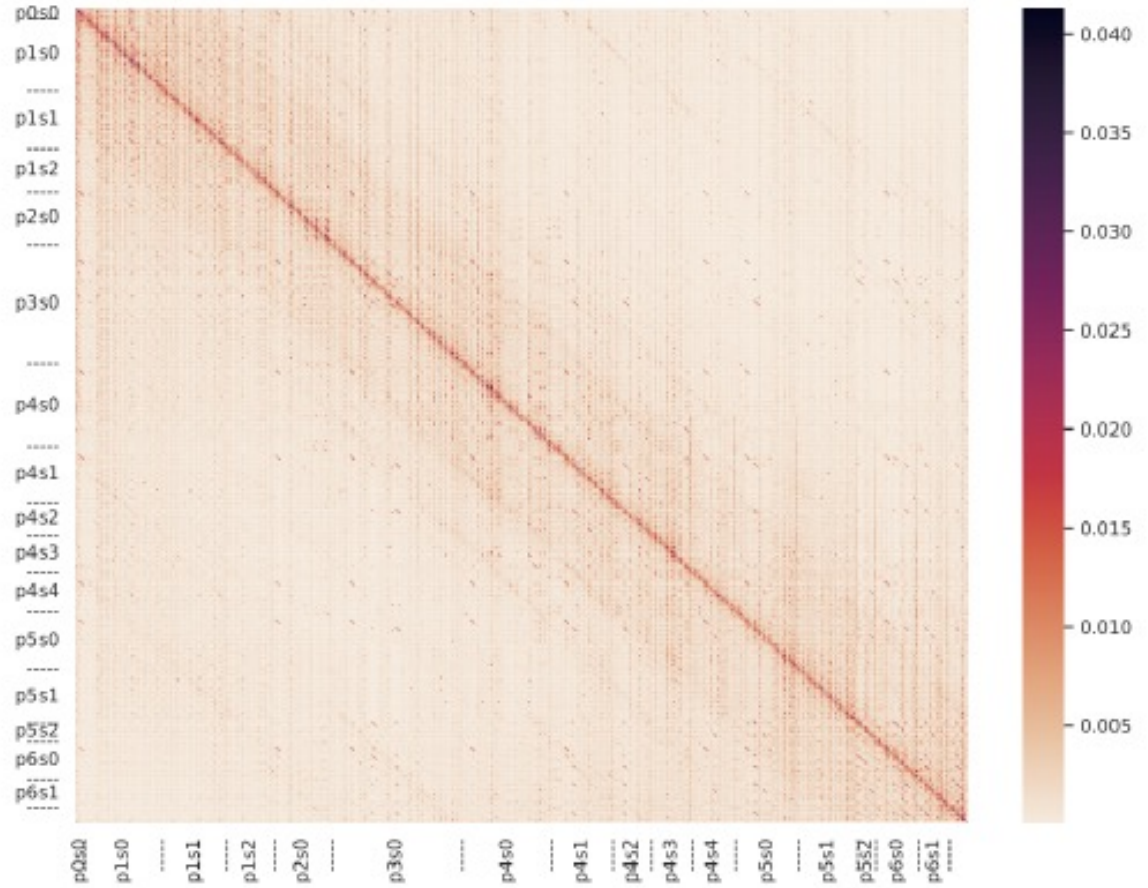
Model	#Param.	PPL
LSTM+Neural cache (Grave et al., 2017)	-	40.8
Hebbian+Cache (Rae et al., 2018)	-	29.9
Transformer-XL base, M=150 (Dai et al., 2019)	151M	24.0
Transformer-XL base, M=150 (ours)	151M	24.4
SegaTransformer-XL base, M=150	151M	**22.5**
Adaptive Input (Baevski and Auli, 2019)	247M	18.7
Transformer-XL large, M=384 (Dai et al., 2019)	257M	18.3
Compressive Transformer, M=1024 (Rae et al., 2020)	257M	17.1
SegaTransformer-XL large, M=384	257M	**17.1**

Table 1: Comparison with Transformer-XL and competitive baseline results on WikiText-103.

Looking at Attention

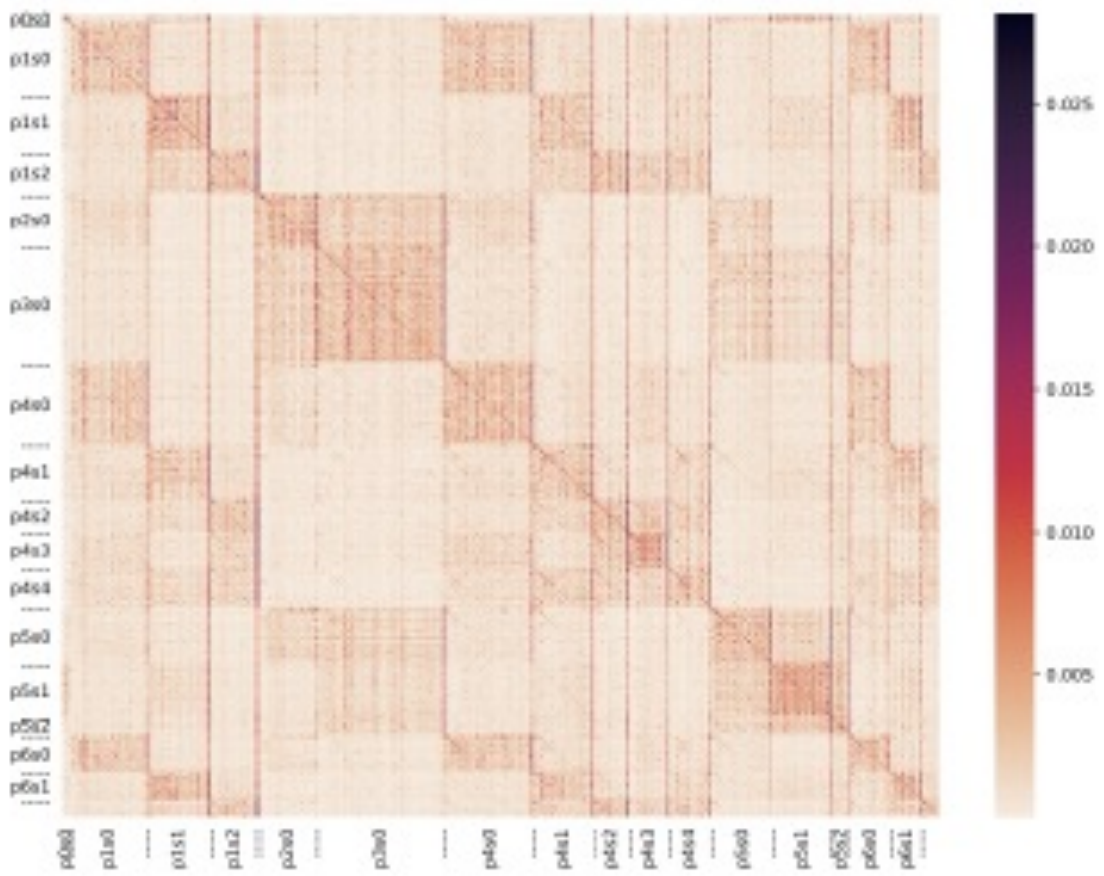


(a) SegaBERT-Layer 1

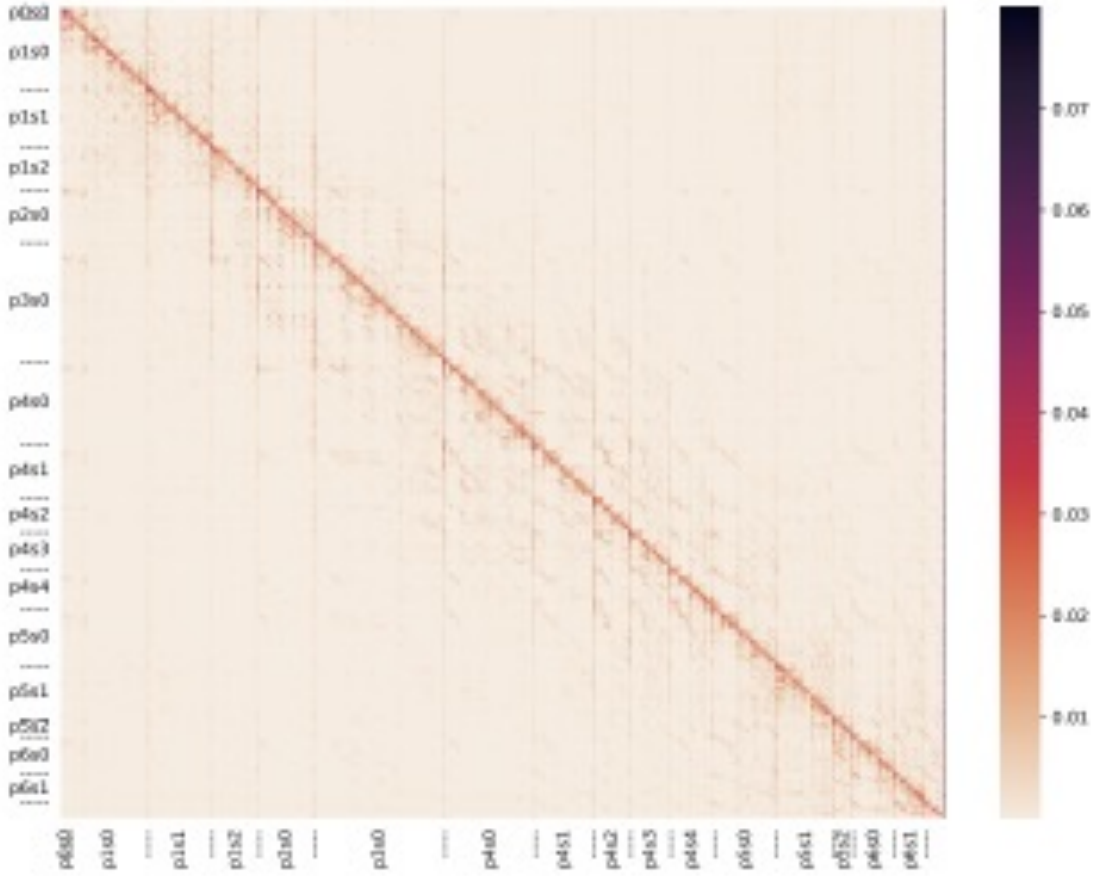


(b) BERT-Layer 1

Looking at Attention

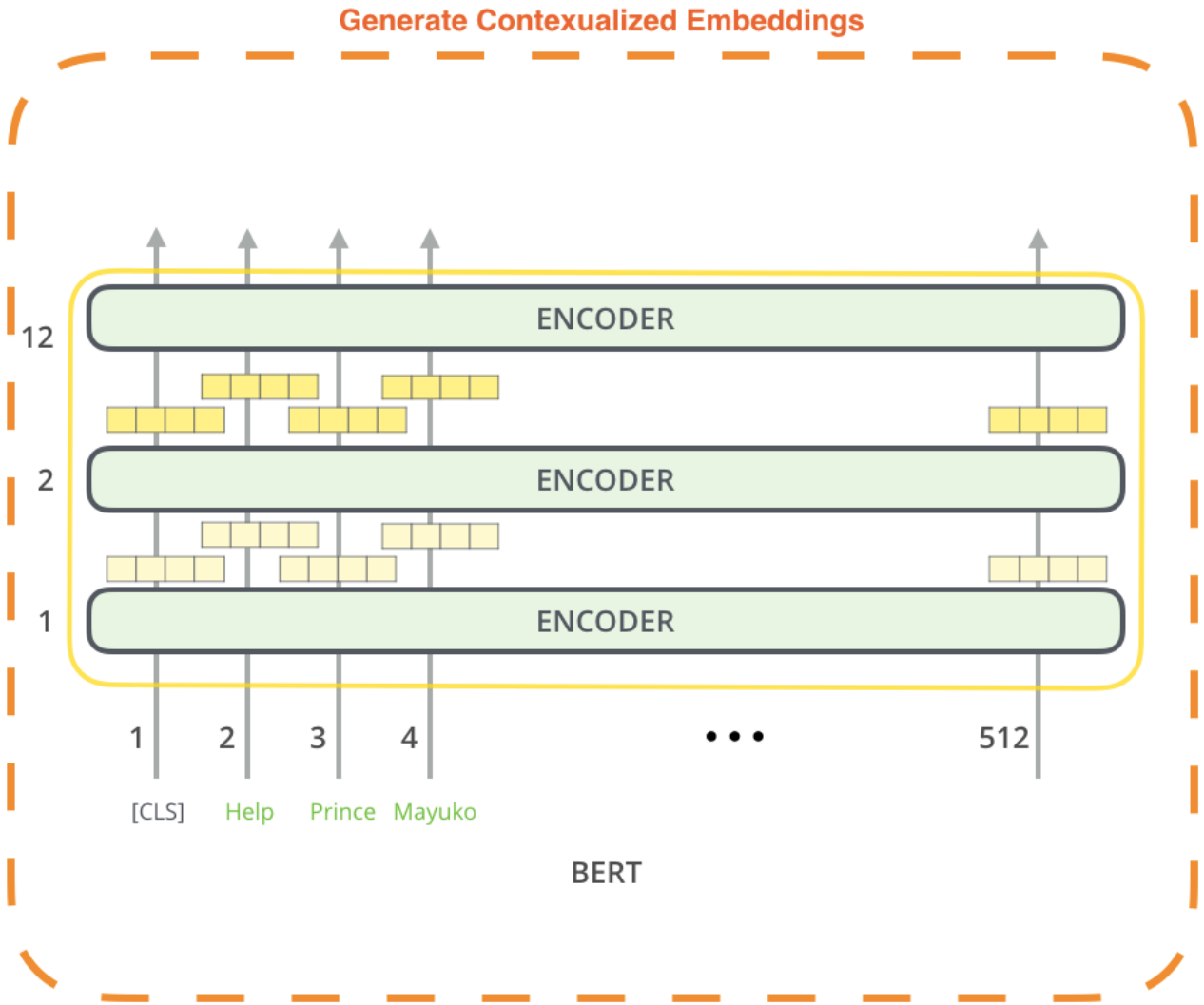


(a) SegaBERT-Layer 6

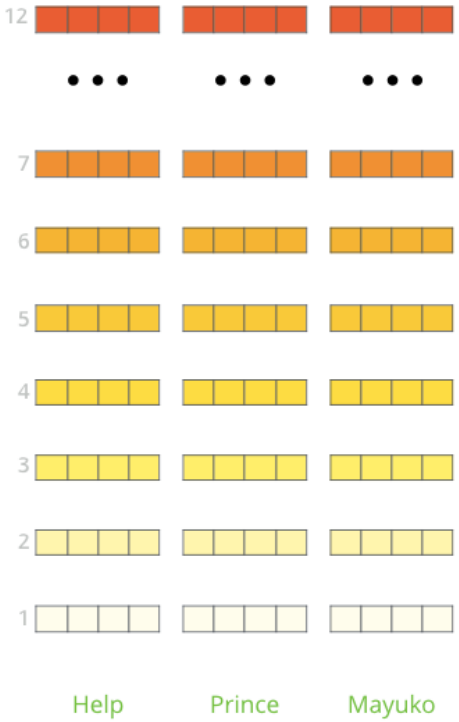


(b) BERT-Layer 6

Feature Extraction



The output of each encoder layer along each token's path can be used as a feature representing that token.



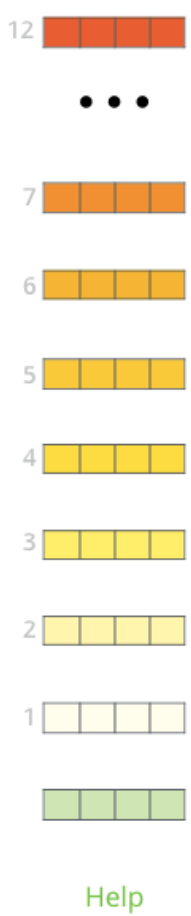
We end up with some embedding for each word related to current input

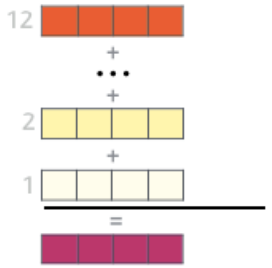
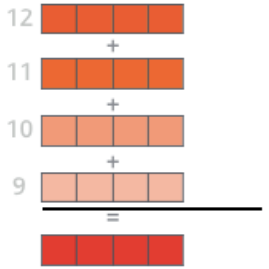
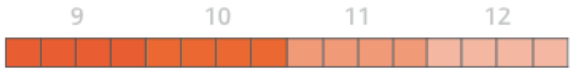
We start with independent word embedding at first level

But which one should we use?

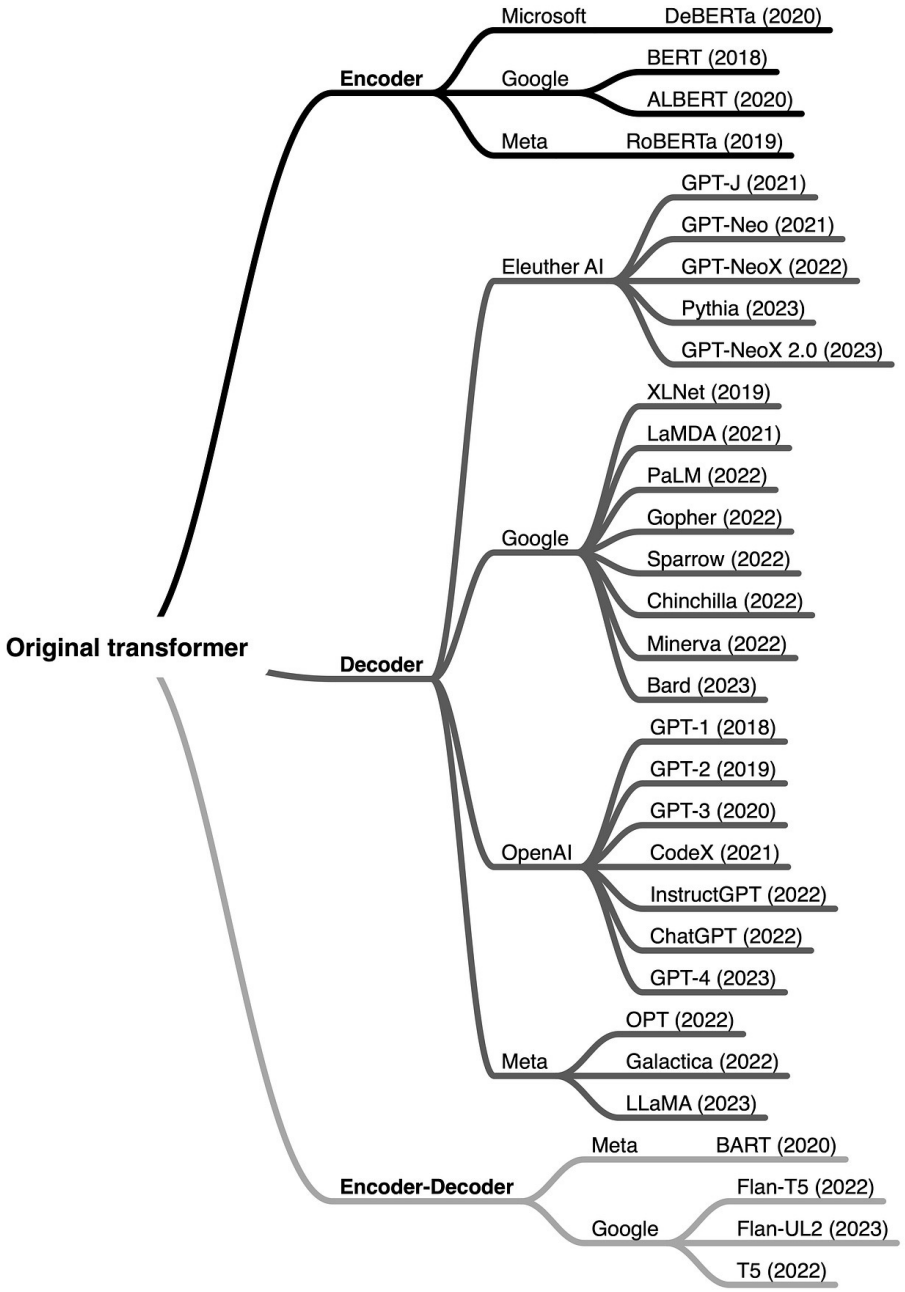
GPT-2, BERT Feature Extraction, which embedding to use?

What is the best contextualized embedding for "Help" in that context?
For named-entity recognition task CoNLL-2003 NER



		Dev F1 Score
First Layer		91.0
Last Hidden Layer		94.9
Sum All 12 Layers		95.5
Second-to-Last Hidden Layer		95.6
Sum Last Four Hidden		95.9
Concat Last Four Hidden		96.1

	GPT-3	BERT
Model	Autoregressive	Discriminative
Objective	Generates human-like text	Recognizes sentiment
Architecture	Unidirectional: it processes text in one direction using a decoder	Bidirectional: it processes text in both directions using an encoder
Size	175 billion parameters	340 million parameters
Training data	It is trained on language modeling by using hundreds of billions of words	340 million parameters
Pre-training	Unsupervised pre-training on a large data	Unsupervised pre-training on a large corpus of text
Fine-turning	Does not require but can be fine-tuned for specific tasks	Requires fine-tuning for specific tasks
Uses cases	• Coding • ML code generation • Chatbots and virtual assistants • Creative storytelling • Language translation	• Sentiment analysis • Text classification • Question answering • Machine translation
Accuracy	Based on the SuperGLUE benchmark, 86.9%	Based on the GLUE benchmark, 80.5%


Original transformer
Encoder
Microsoft
DeBERTa (2020)
Google
BERT (2018)
ALBERT (2020)
Meta
RoBERTa (2019)
Decoder
Eleuther AI
GPT-J (2021)
GPT-Neo (2021)
GPT-NeoX (2022)
Pythia (2023)
GPT-NeoX 2.0 (2023)
Google
XLNet (2019)
LaMDA (2021)
PaLM (2022)
Gopher (2022)
Sparrow (2022)
Chinchilla (2022)
Minerva (2022)
Bard (2023)
OpenAI
GPT-1 (2018)
GPT-2 (2019)
GPT-3 (2020)
CodeX (2021)
InstructGPT (2022)
ChatGPT (2022)
GPT-4 (2023)
Meta
OPT (2022)
Galactica (2022)
LLaMA (2023)
Encoder-Decoder
Meta
BART (2020)
Google
Flan-T5 (2022)
Flan-UL2 (2023)
T5 (2022)