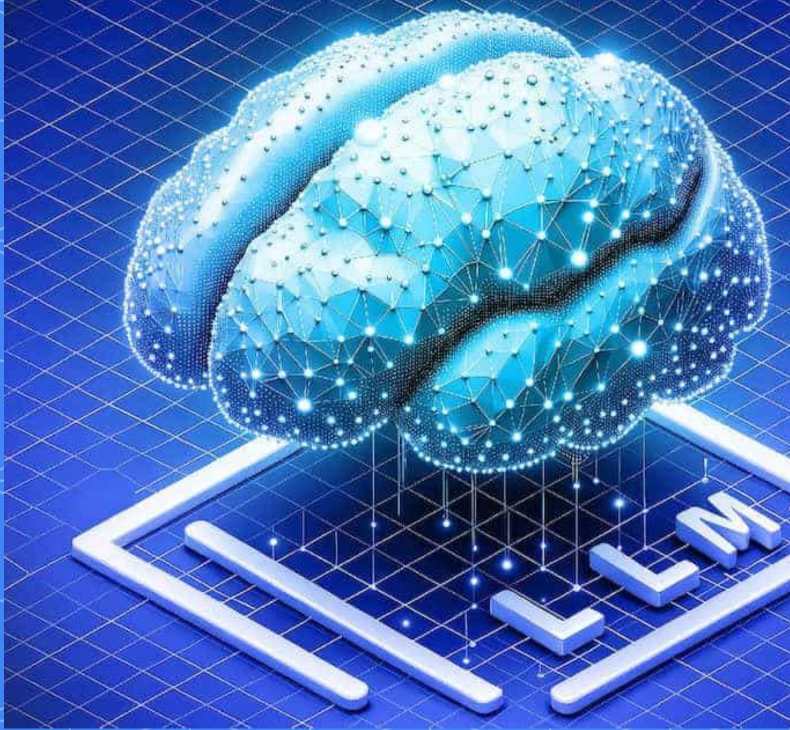
Google Developer Student Clubs
University of Toronto Mississauga

Build with AI: Intro to LLMs



Presented by;
Hamza & Rahul

Google Developer Student Clubs
University of Toronto Mississauga

Introduction

```
lookup.KeyValue
f.constant(['em
=tf.constant([0
.lookup.StaticV

_buckets=5)
```

Important Notice

- Please scan the QR code to join the UTM GDSC discord where you can keep up to date with all our activities
- Also you should follow the Instagram @gdscutm
- Women In Tech Conference coming up March 23rd and 24th, almost sold out, go to @witgdsc for more details

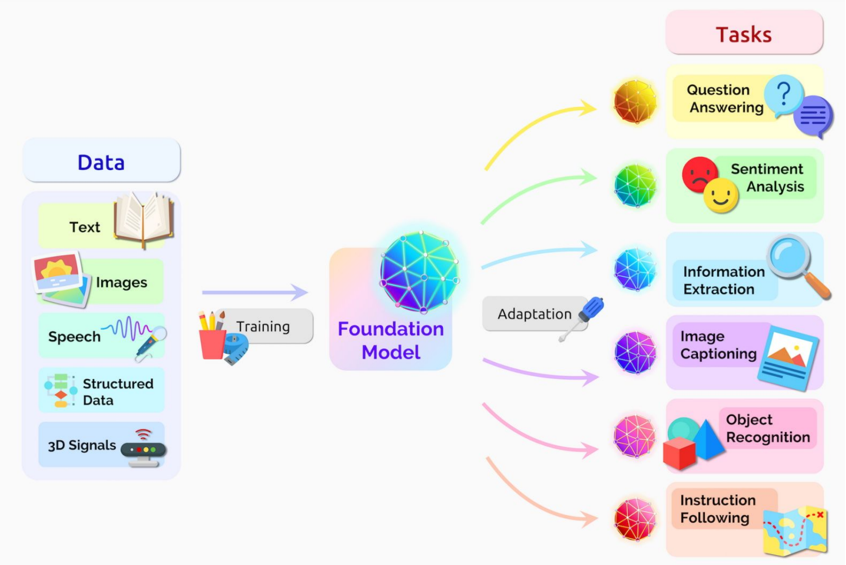
SCAN ME

Citations

- https://pages.cs.wisc.edu/~kandasamy/courses/23spring-cs760/slides/lecture26-langmodels.pdf
- https://docs.google.com/presentation/d/1FpNBGb7Ow2p5krUcfPaIHHq9Y8SlGV5Eb2ESAIWkWHA/edit#slide=id.p
- https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers
- https://nlp.seas.harvard.edu/2018/04/03/attention.html
- https://jalammar.github.io/illustrated-transformer/
- https://www.youtube.com/watch?v=zjkBMFhNj_g
- https://aws.amazon.com/what-is/foundation-models/

Foundation Models

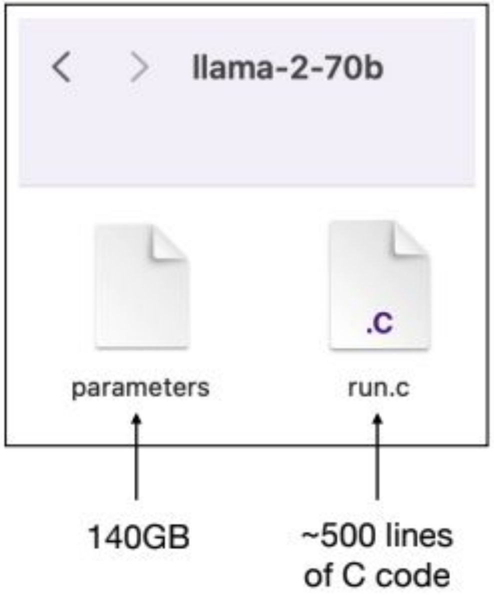
- Foundation models are a form of generative artificial intelligence
- They generate output from one or more human language inputs (prompts)
- Models are based on complex neural networks including generative adversarial networks (GANs), transformers, and variational encoders



What really is a Large Language Model…?

- Large language models are a type of foundation model
- Large language models are parameterized as a transformer, a type of deep neural network
- Generally, decoder-only (eg. GPT), and can be encoder-decoder
- Large language models are pre-trained via self-supervision (next-token prediction)
- Ok now you know essentially a lot of words but what does this really look like

What really is a Large Language Model...?



Google Developer Student Clubs
University of Toronto Mississauga

LLM Training

```
lookup.KeyValue
f.constant(['em
=tf.constant([
.lookup.StaticV

_buckets=5)
```

Compressing the Internet

Chunk of the internet,
~10TB of text

→

6,000 GPUs for 12 days, ~$2M
~1e24 FLOPS

→

parameters.zip

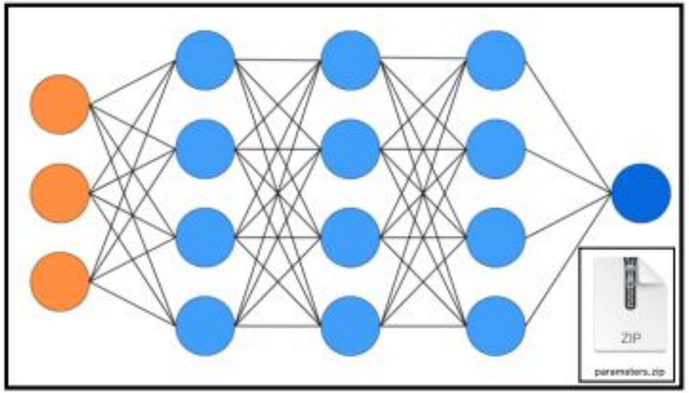
~140GB file

*numbers for Llama 2 70B

Neural Network

Predicts the next word in the sequence.

cat →
sat →
on →
a →



→ mat (97%)

e.g. context of 4 words

predict next word

Language Modeling

- Goal: Model the probability distribution p(x), for any sentence x
- Mathematically, we can break it down into next-token-prediction;

$p(x) = p(x_t, x_{t-1}, …, x_1)$

$= p(x_t \mid x_{t-1}, . . ., x_1)\ p(x_{t-1} …, x_1)$ **[apply chain rule of probability]**

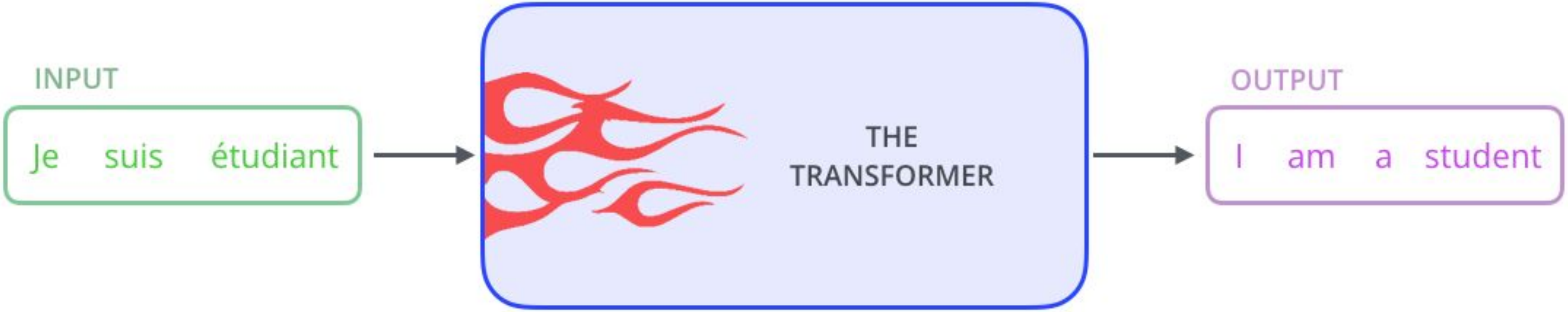
$= p(x_t \mid x_{t-1}, . . ., x_1)\ p(x_{t-1} \mid x_{t-2}, . . ., x_1)\ . . .\ p(x_1)$ **[recursively]**

Google Developer Student Clubs
University of Toronto Mississauga

Transformer Architecture

```
lookup.KeyValue
f.constant(['em
=tf.constant([
.lookup.Static

_buckets=5)
```

High Level Overview



INPUT

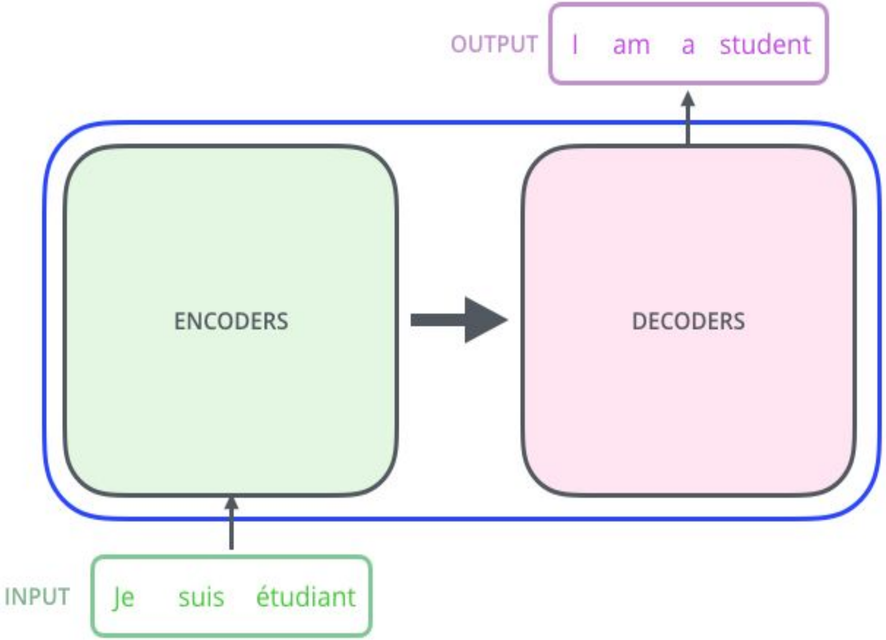
Je suis étudiant

THE TRANSFORMER

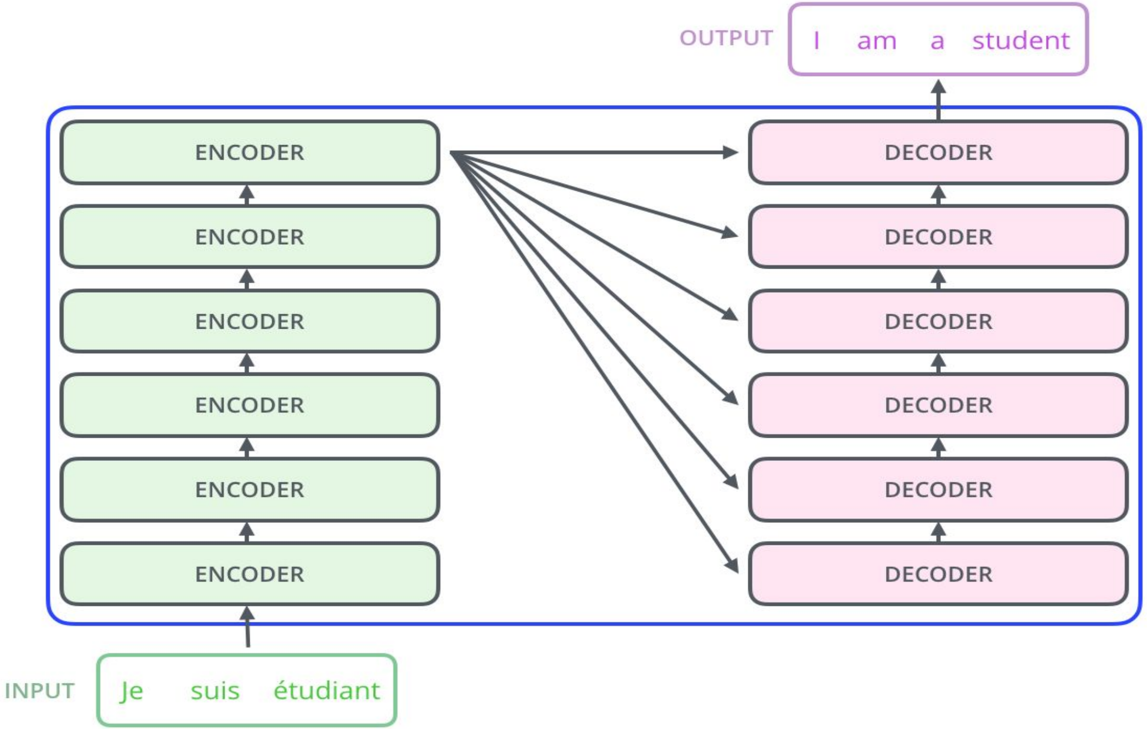
OUTPUT

I am a student

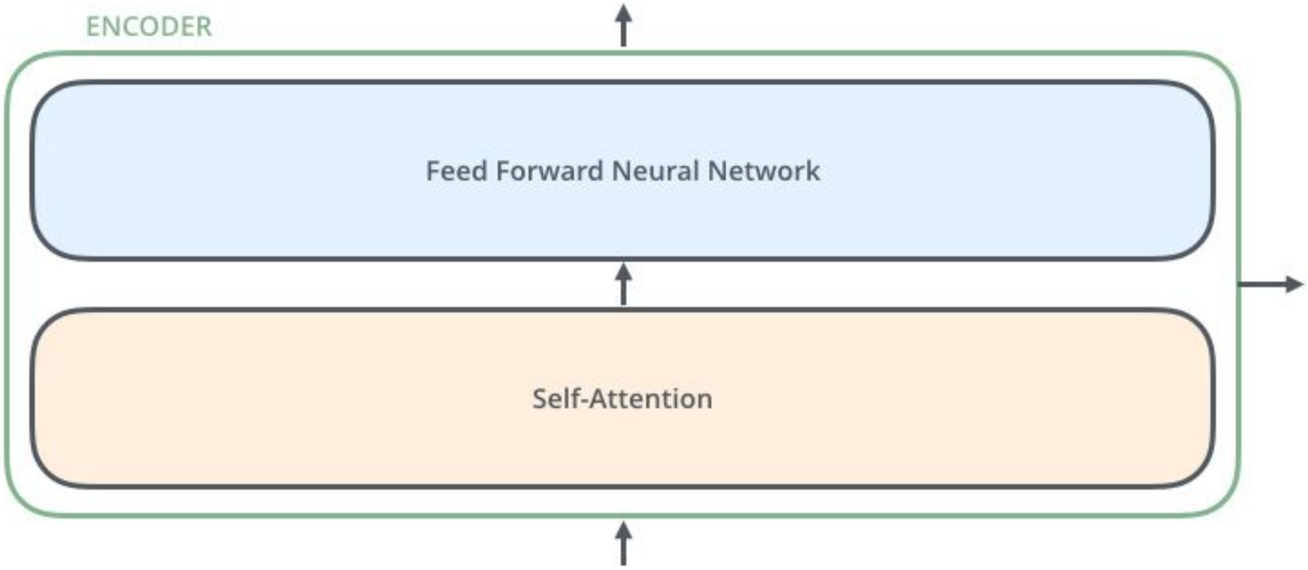
High Level Overview



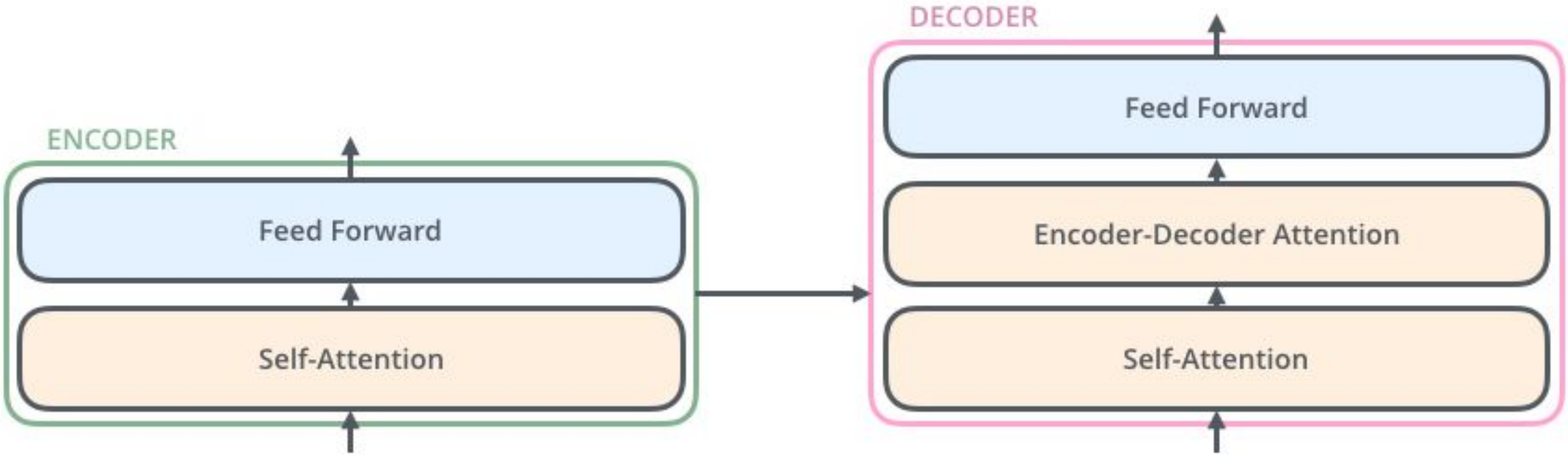
High Level Overview



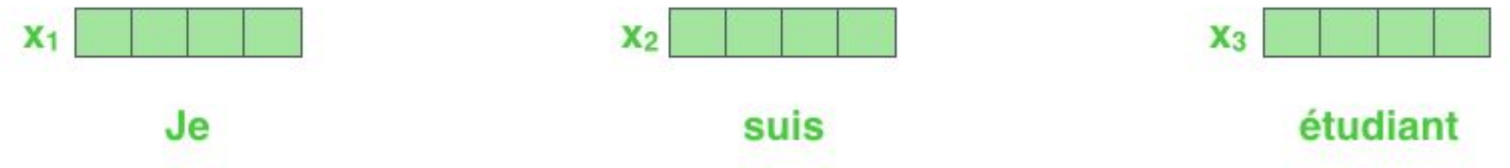
High Level Overview



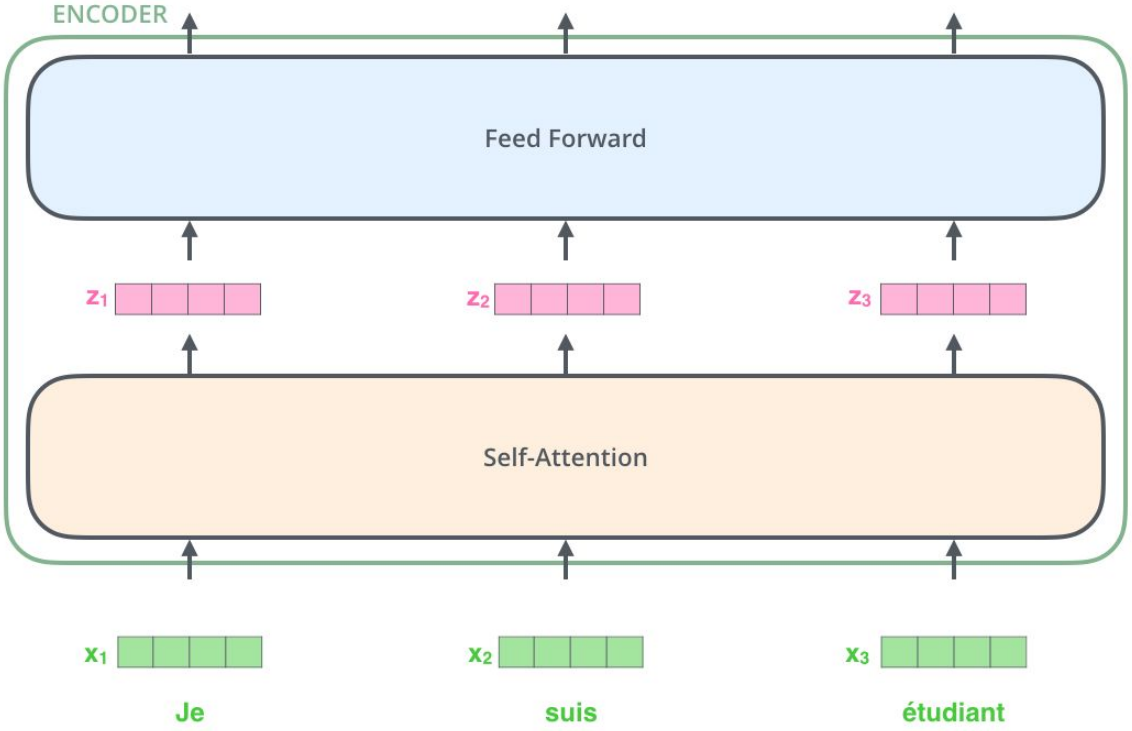
High Level Overview



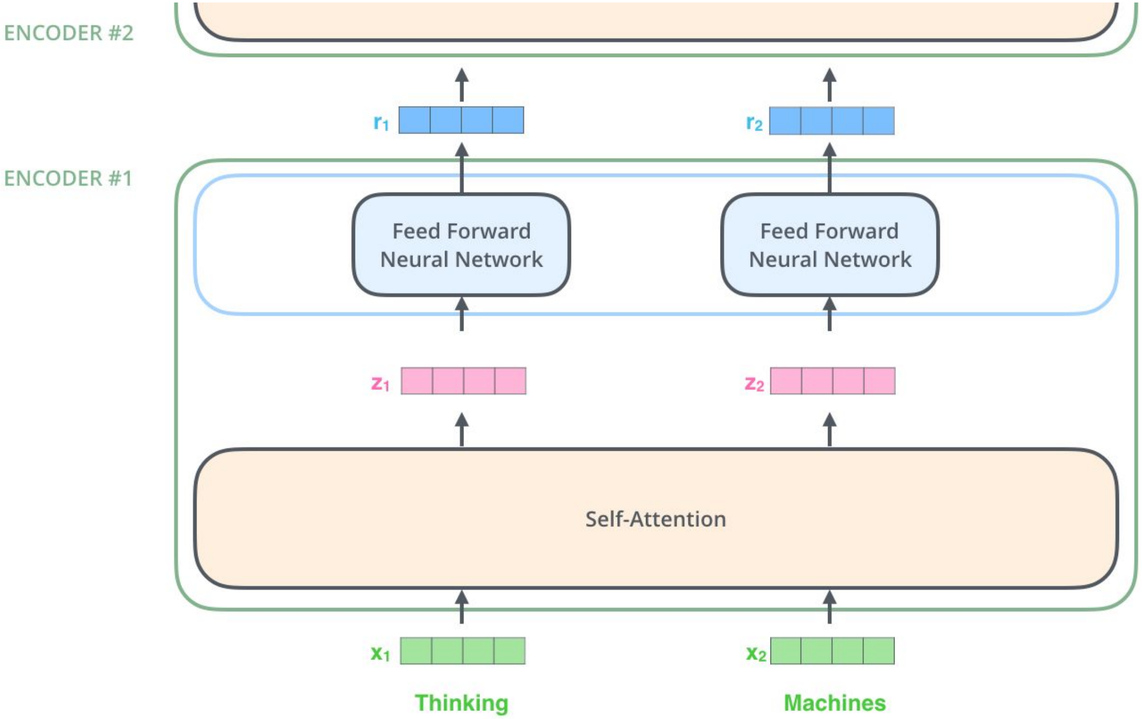
Input



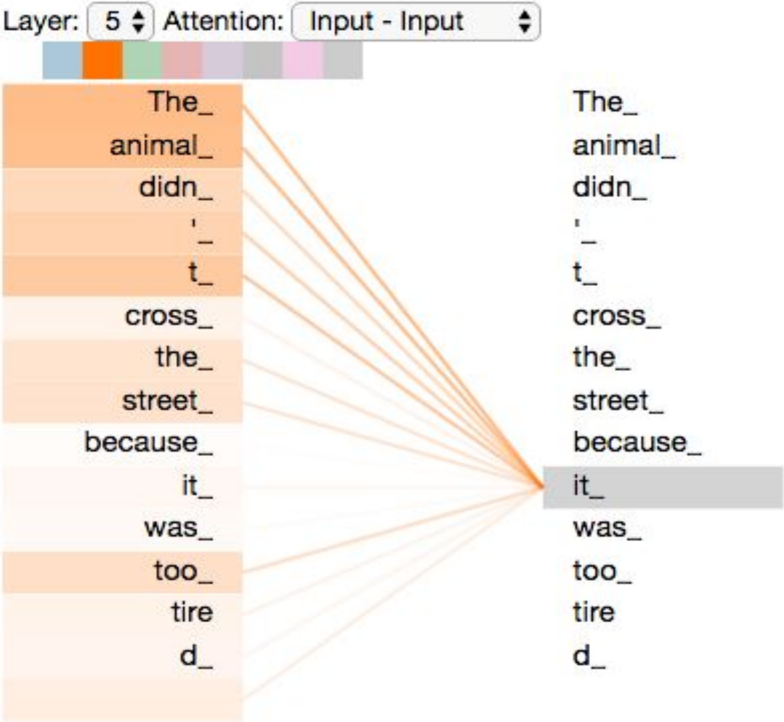
Input with Encoder



Multi-Encoder



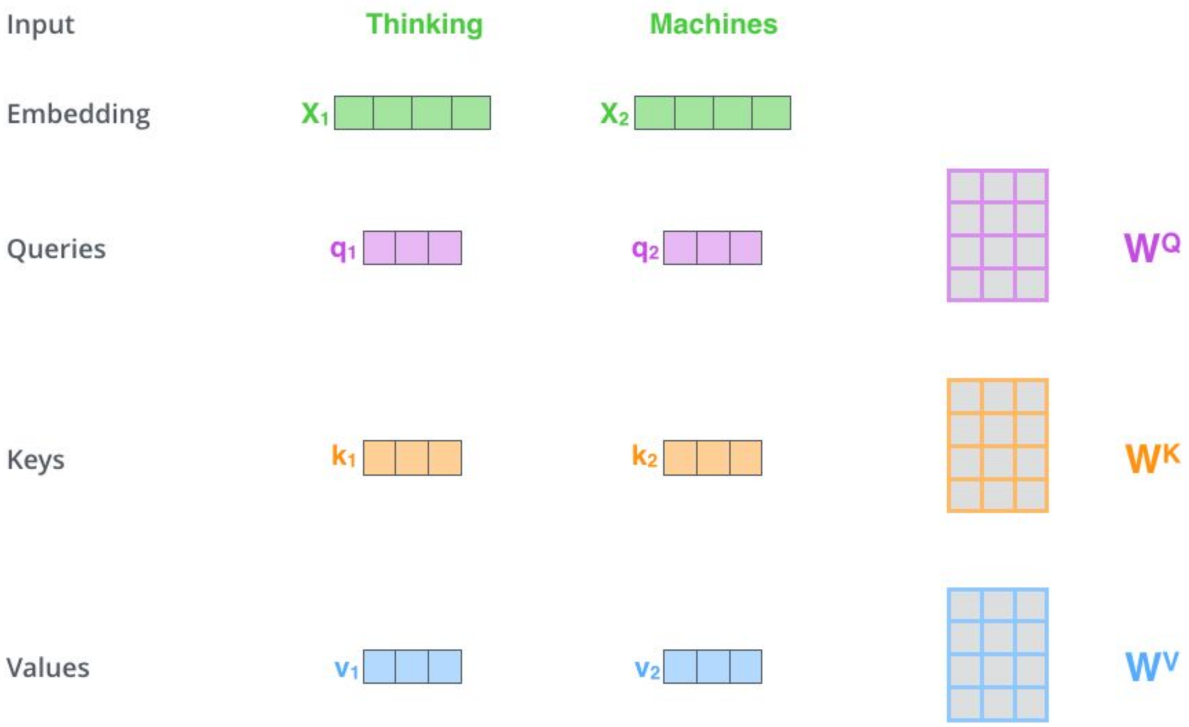
Self-Attention



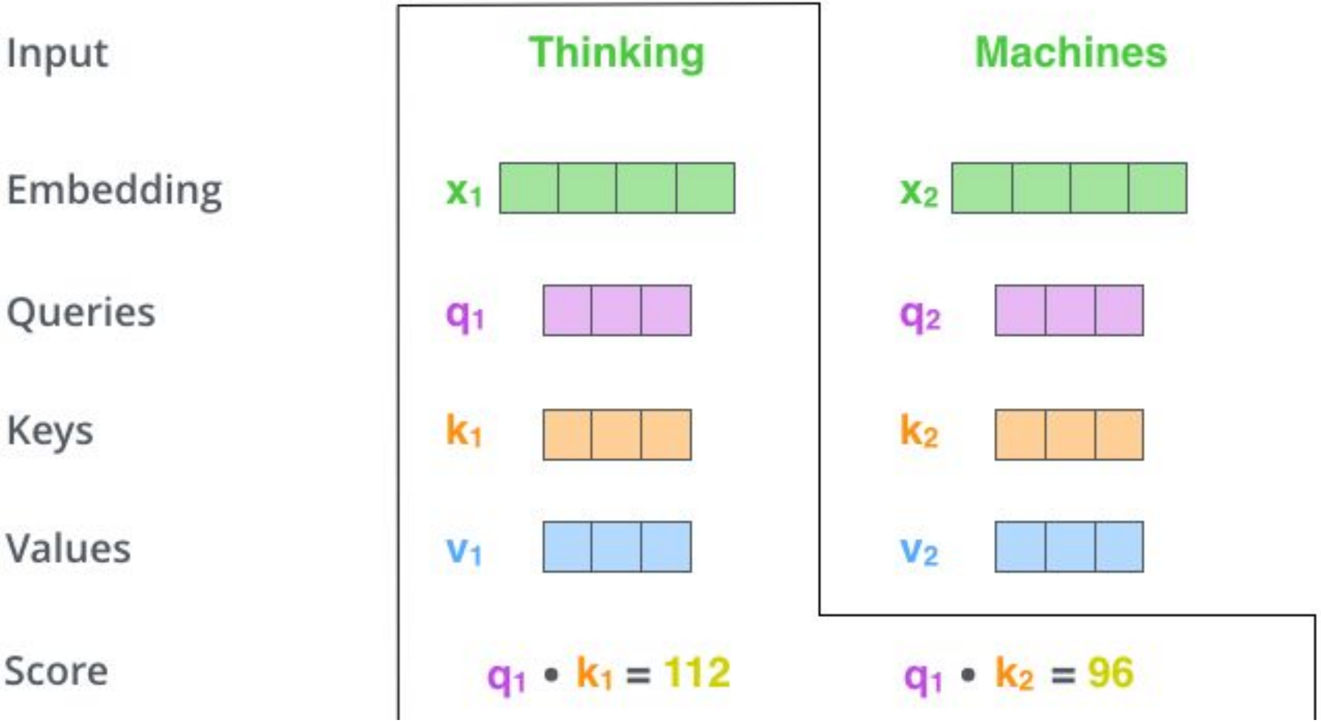
Demo

- https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfqOC

Self-Attention Step 1

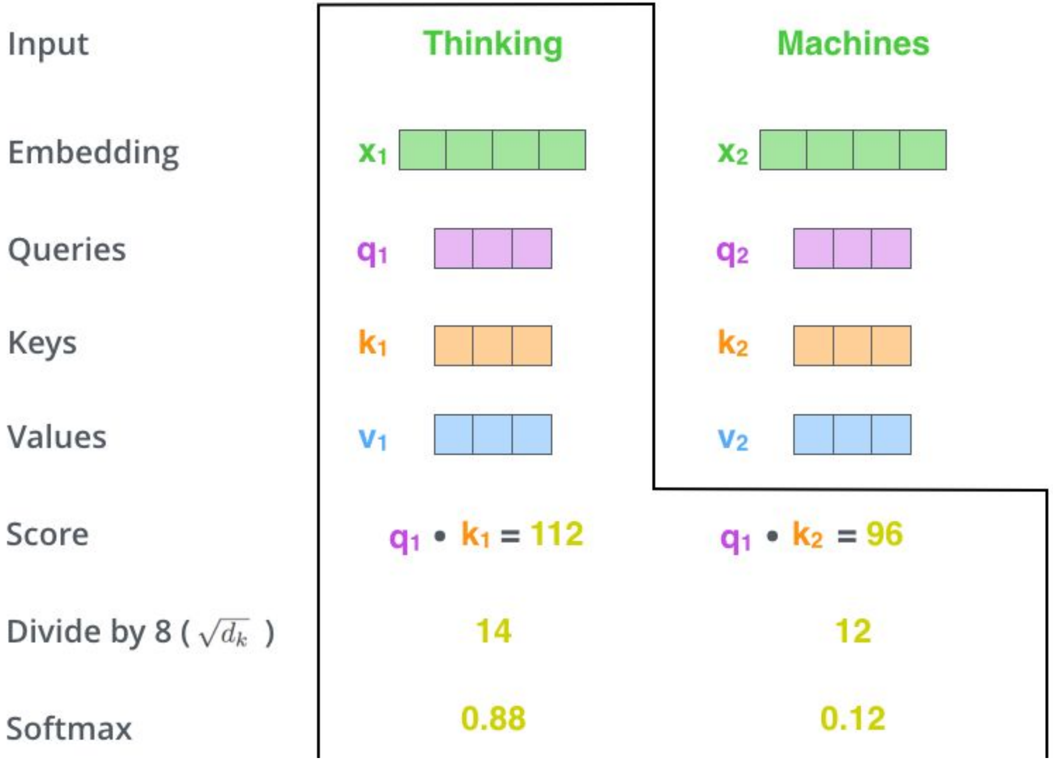


Self-Attention Step 2



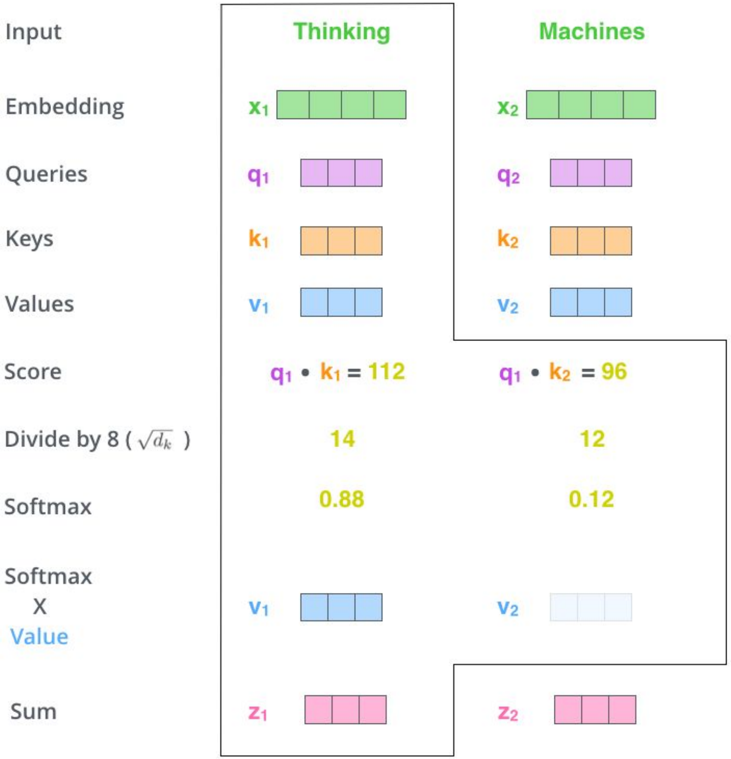
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$

Self-Attention Step 3



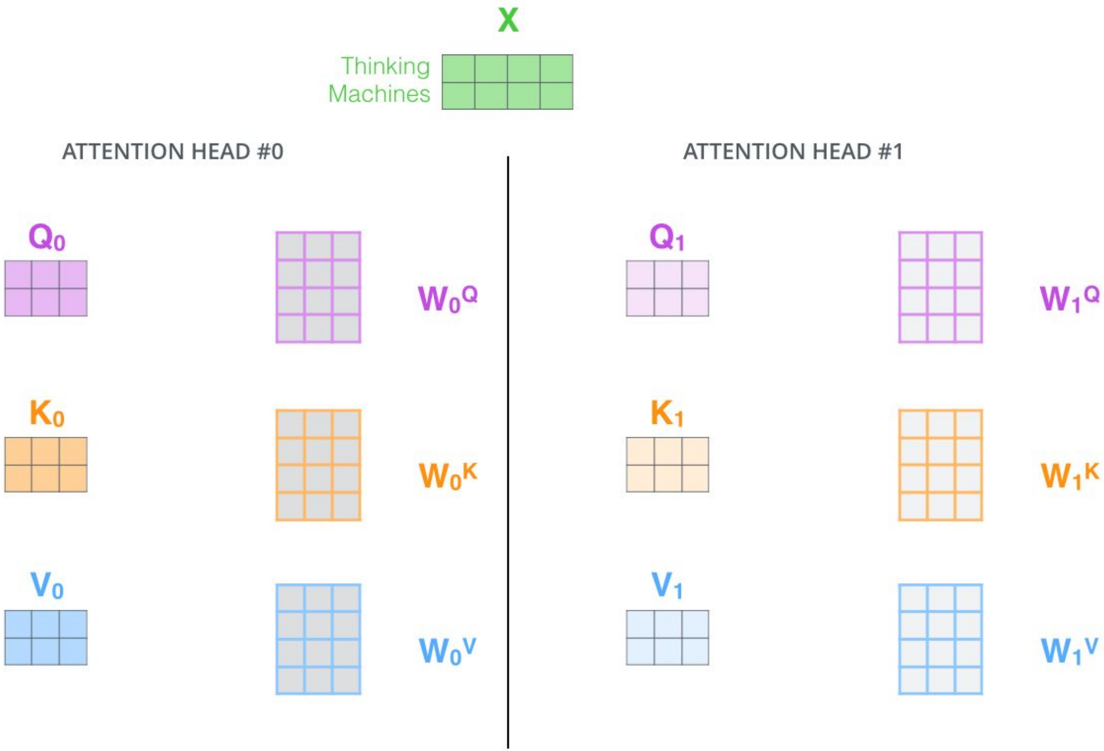
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

Self-Attention Step 4

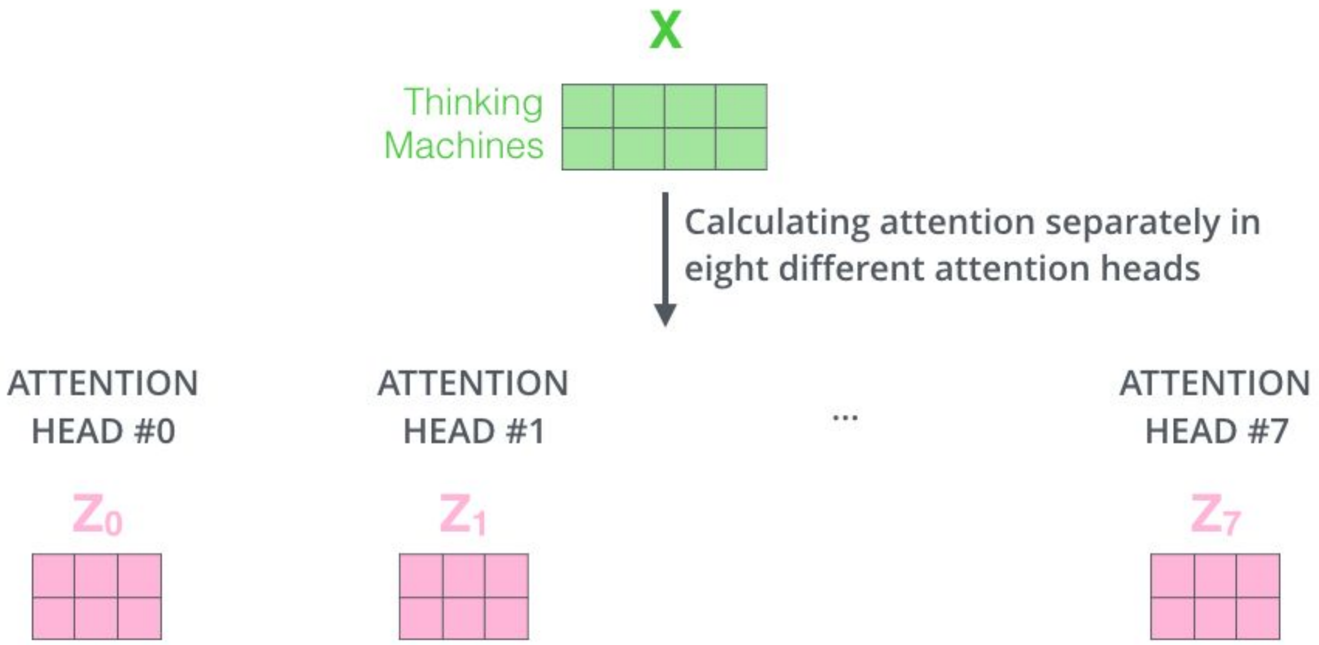


Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1	v_2
Sum	z_1	z_2

Multi-Head Self-Attention



Multi-Head Self-Attention



X

Thinking
Machines

Calculating attention separately in eight different attention heads

ATTENTION HEAD #0

Z_0

ATTENTION HEAD #1

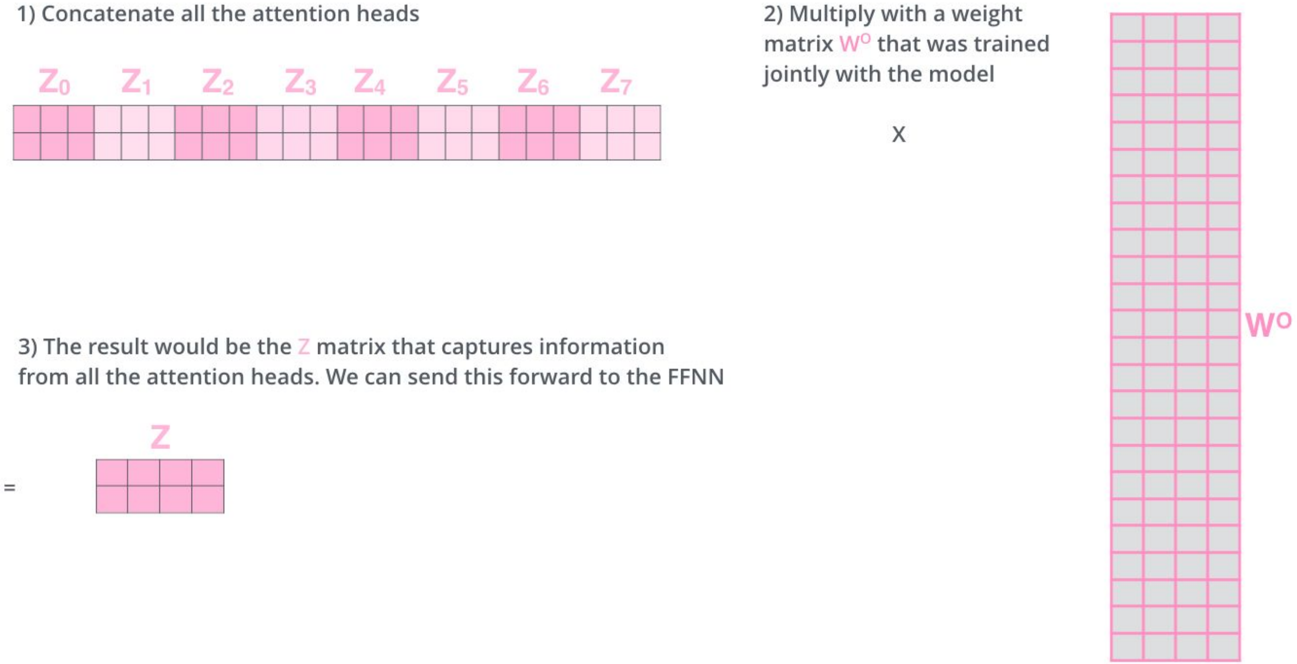
Z_1

...

ATTENTION HEAD #7

Z_7

Multi-Head Self-Attention Unification



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

Multi-Head Self-Attention Unification Summary

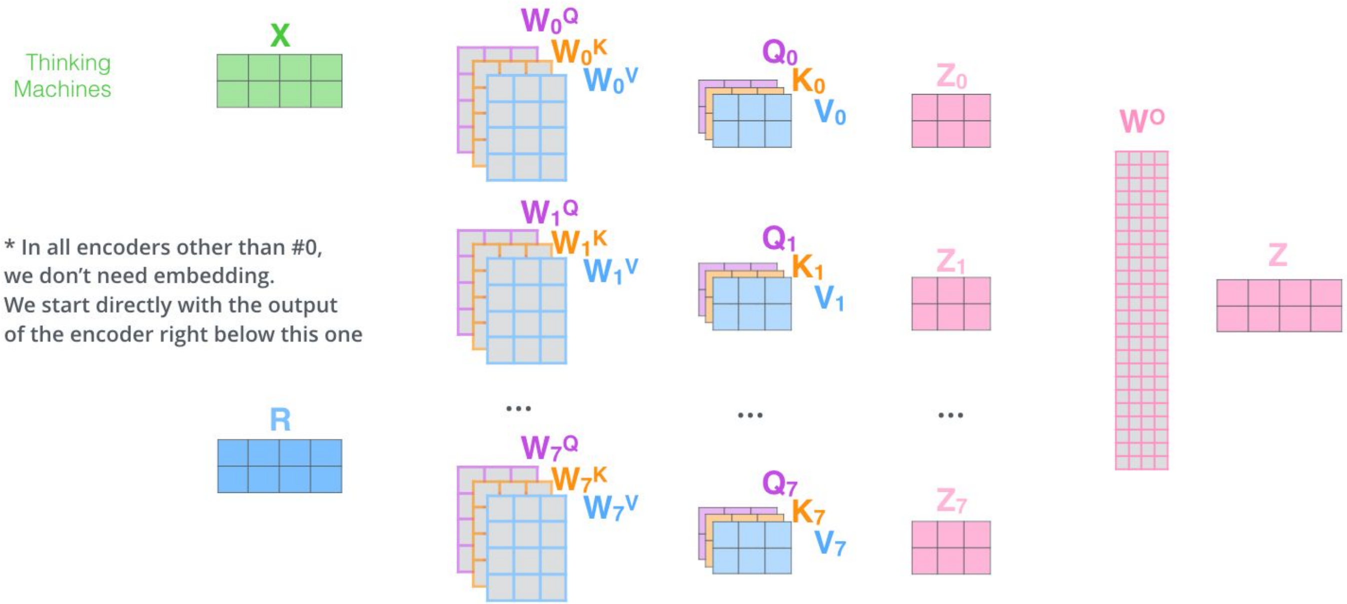
1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

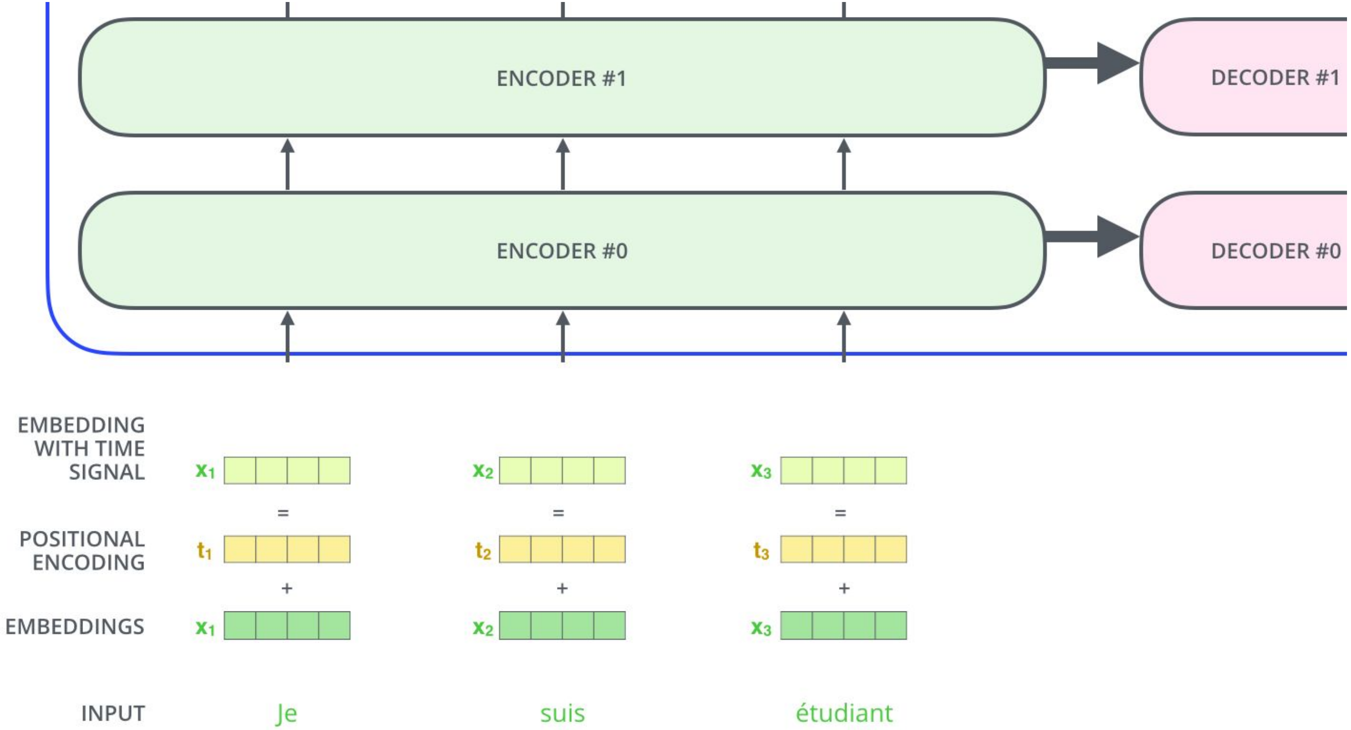
4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

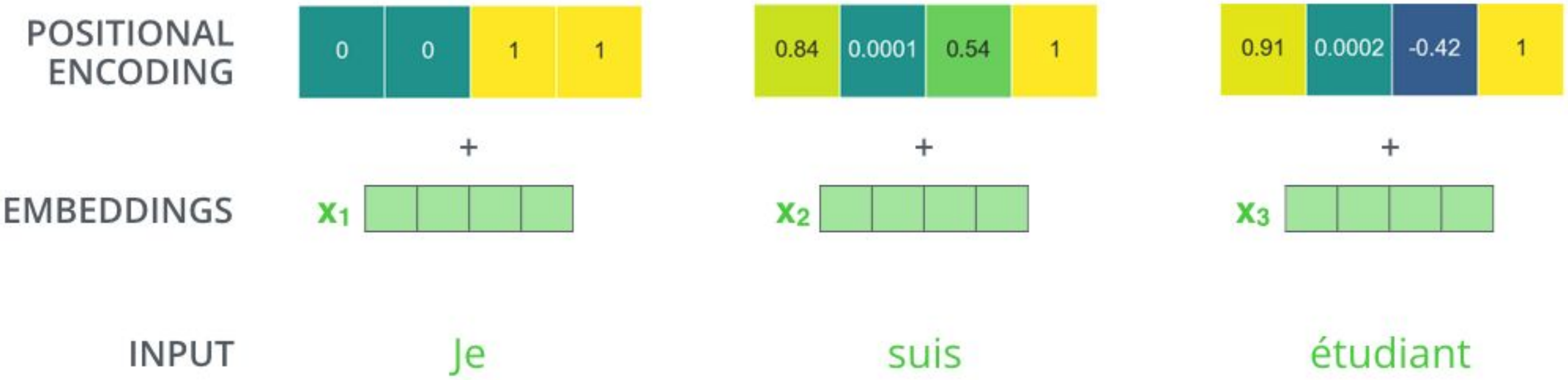


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Positional Encoding



Positional Encoding Example



POSITIONAL ENCODING

0	0	1	1

0.84	0.0001	0.54	1

0.91	0.0002	-0.42	1

+ + +

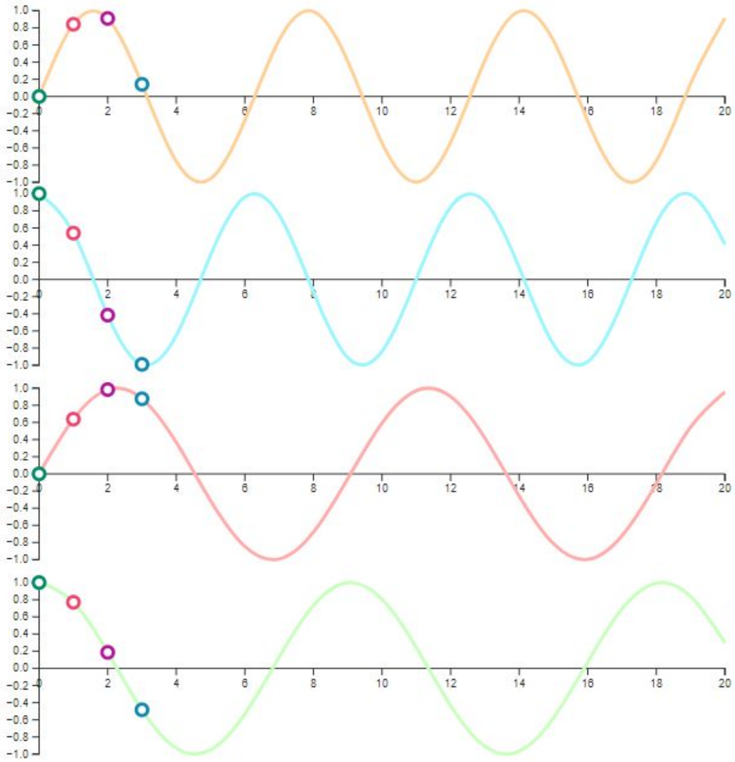
EMBEDDINGS

x_1 x_2 x_3

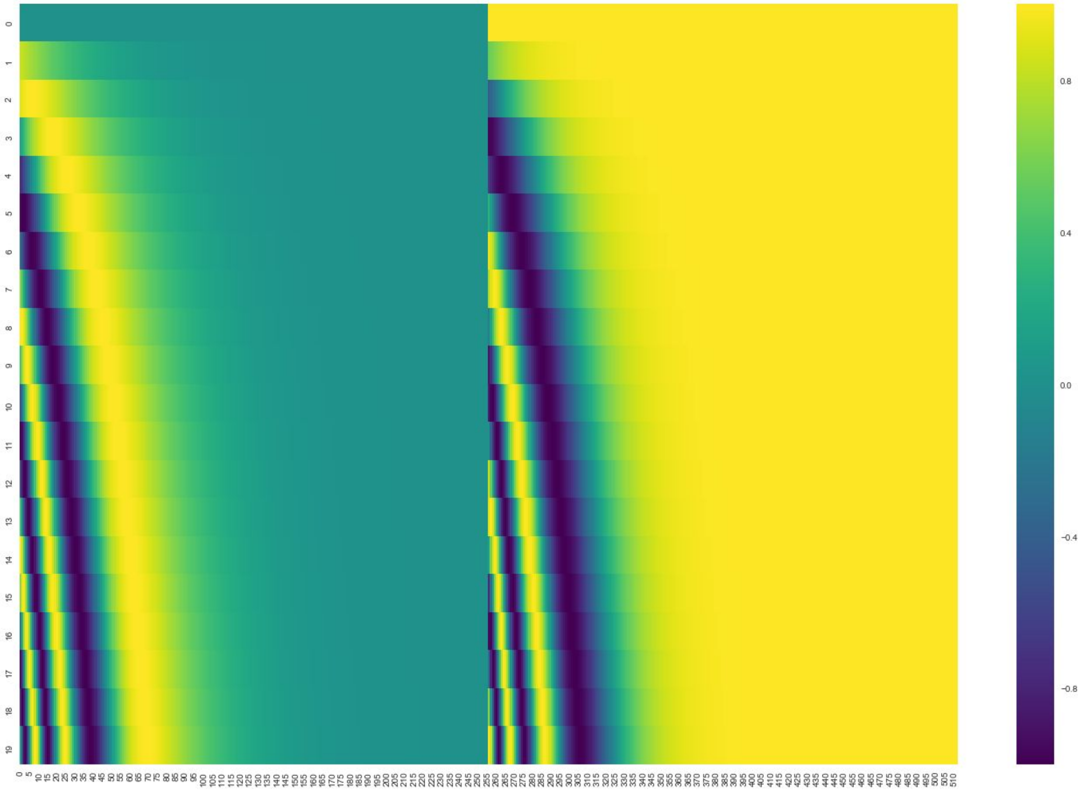
INPUT

Je suis étudiant

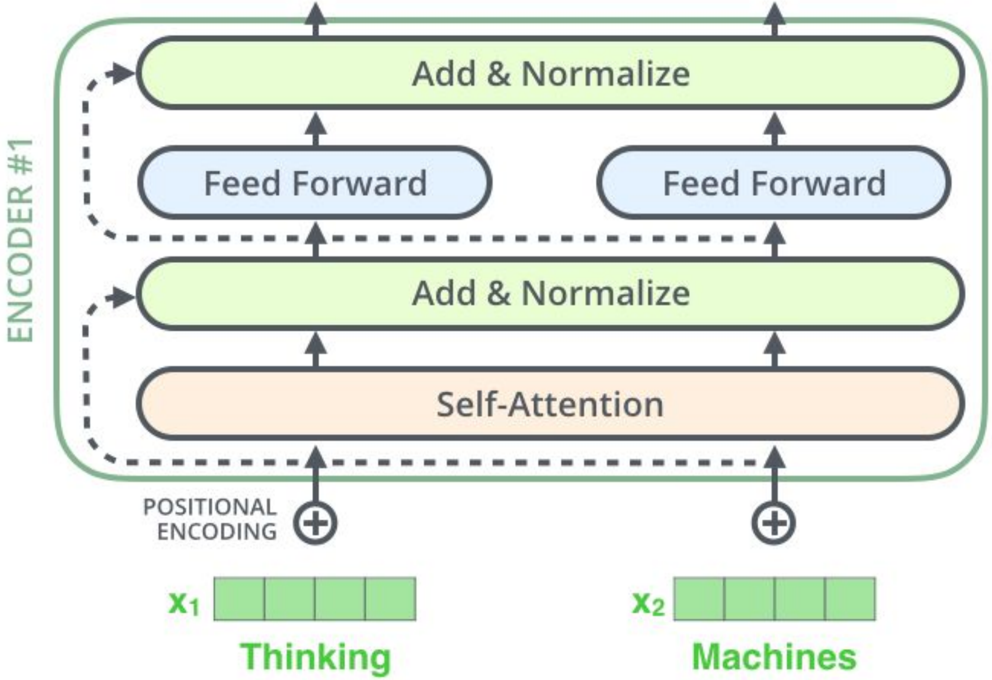
Positional Encoding Multi-Function Approach



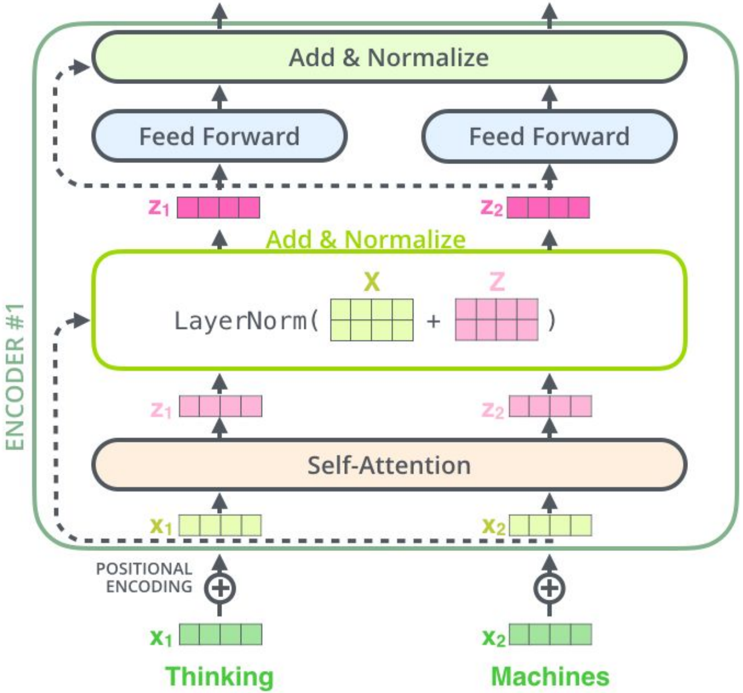
How Positional Encoding actually looks like



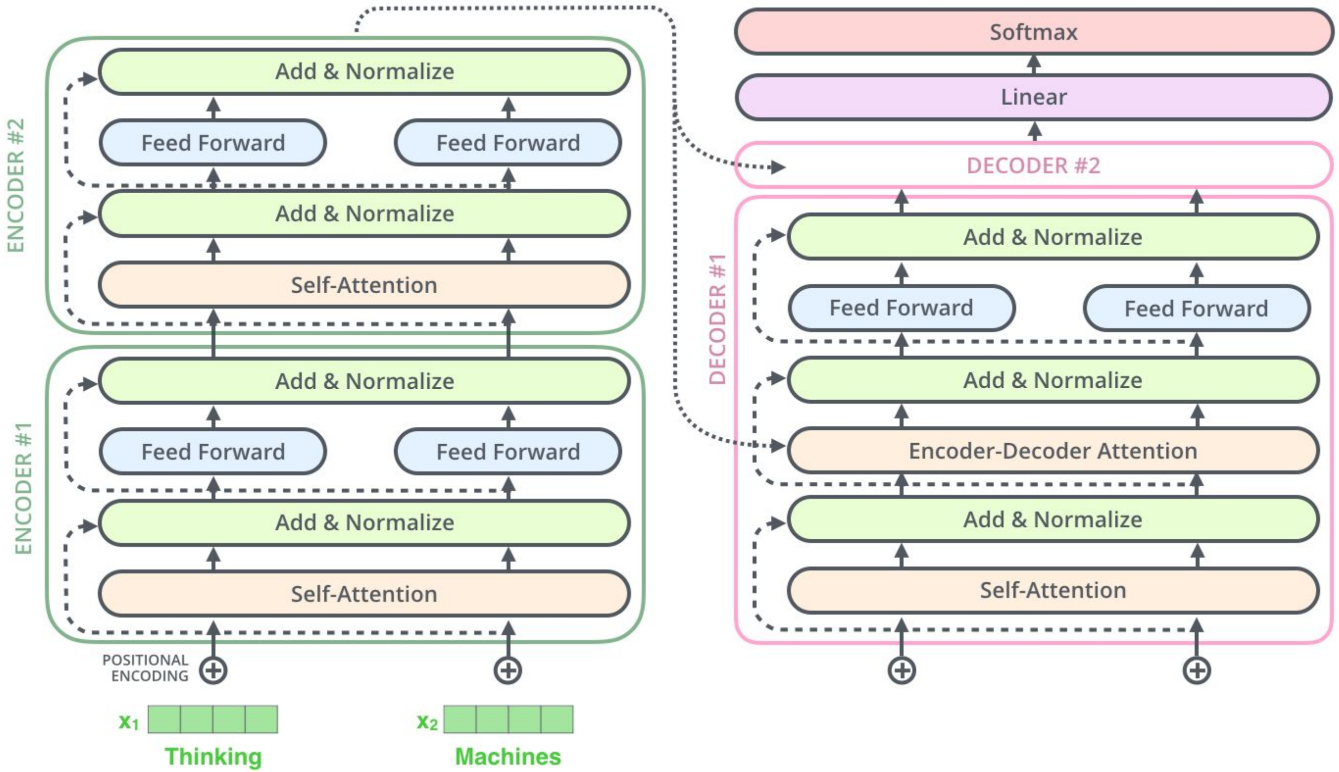
Residuals


Add & Normalize
Feed Forward
Feed Forward
ENCODER #1
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x1
x2
Thinking
Machines

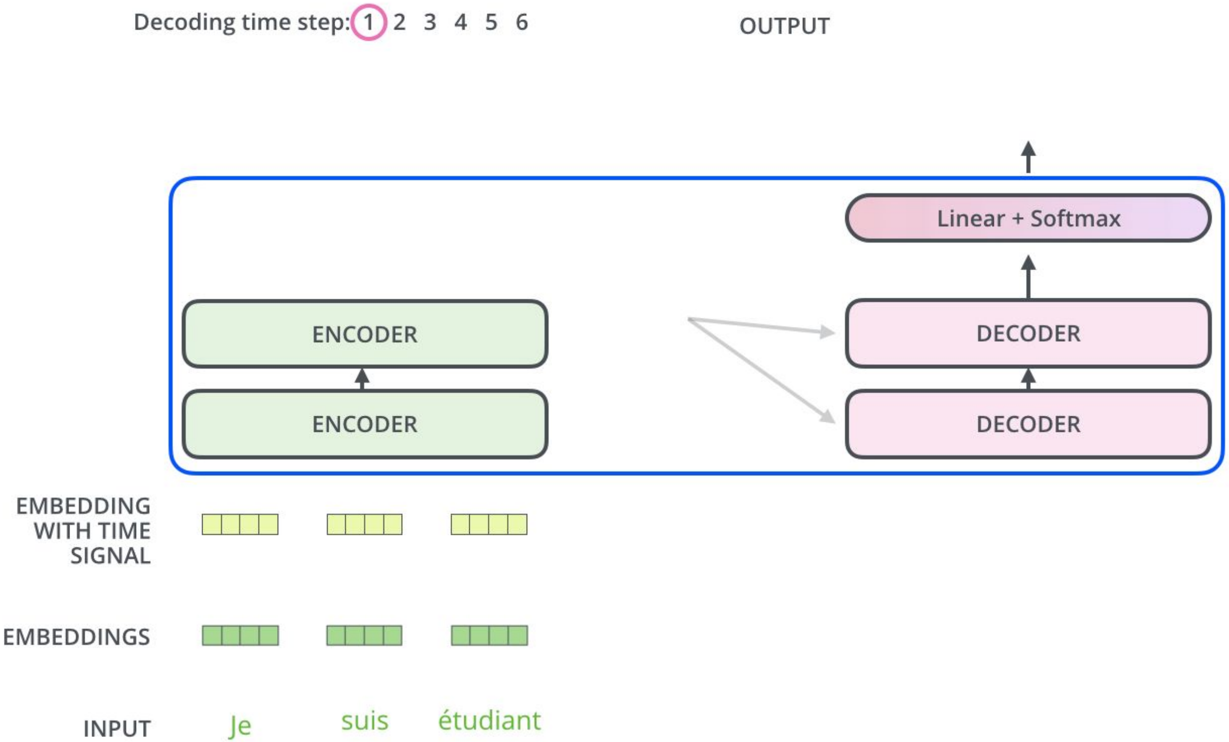
Encoder Deep Dive



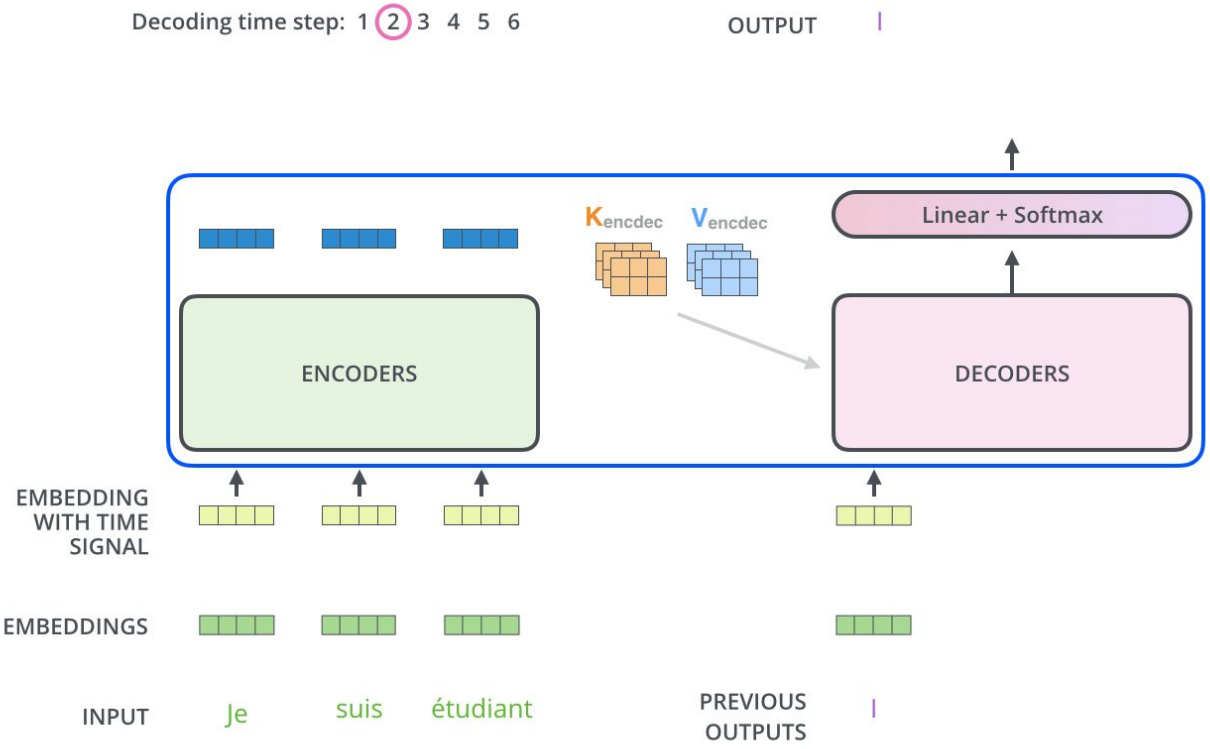
Encoder-Decoder Connection



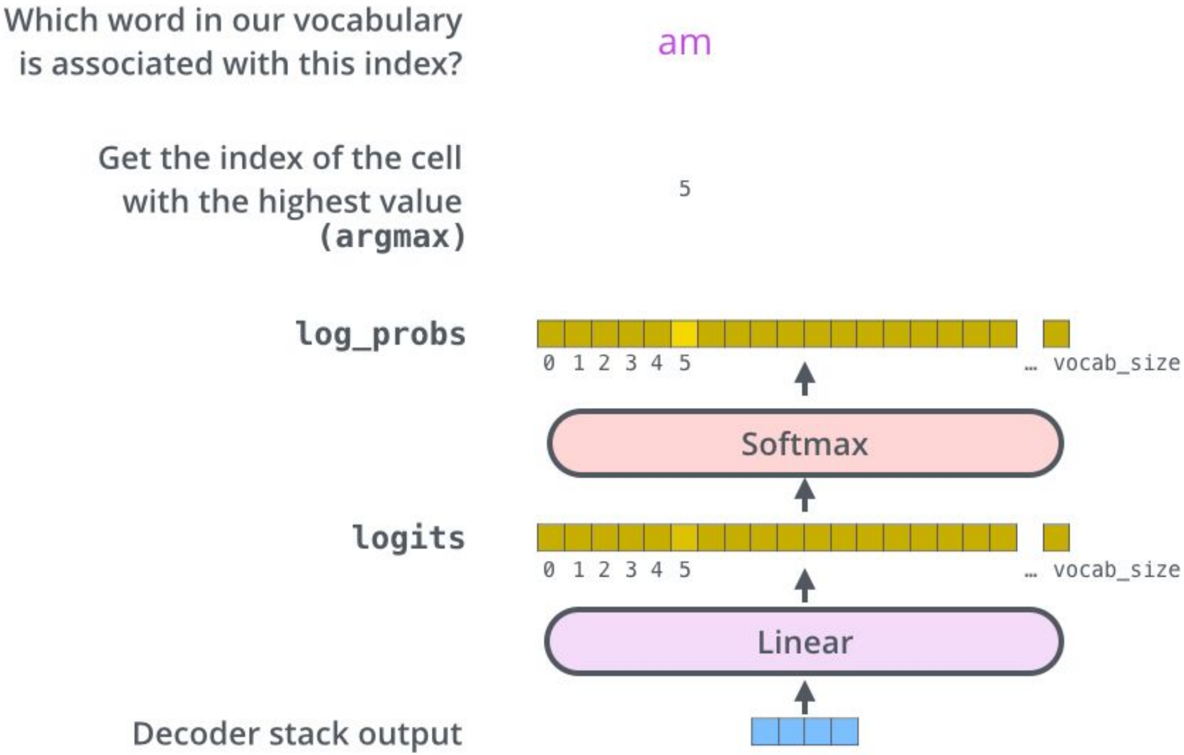
Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

Self-Attention Vectors



Final Layer and Softmax



Which word in our vocabulary is associated with this index?

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

Training

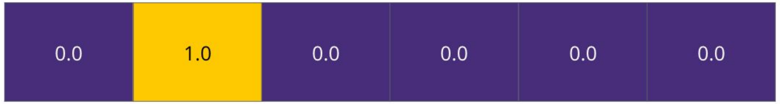
Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

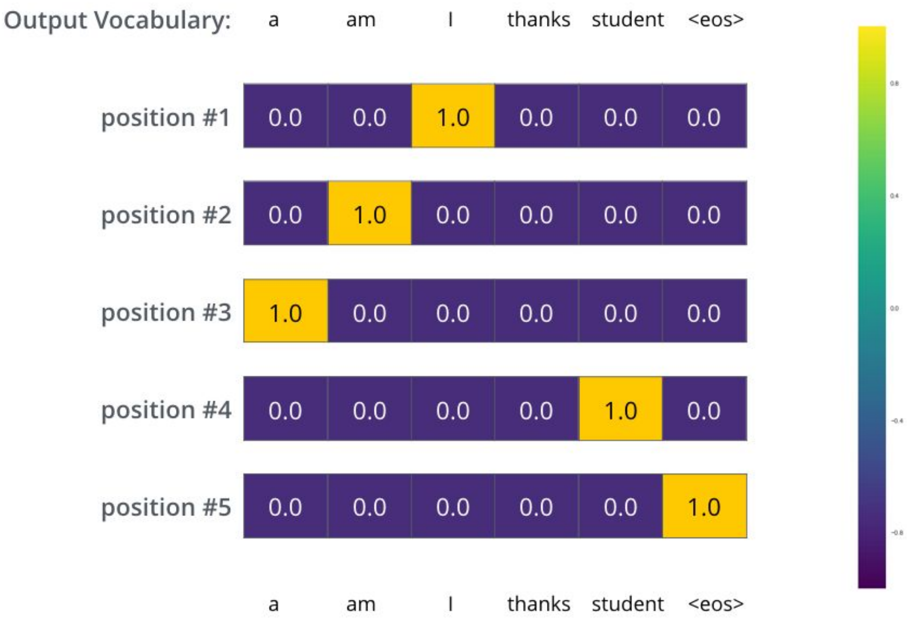
One-hot encoding of the word “am”



0.0	1.0	0.0	0.0	0.0	0.0

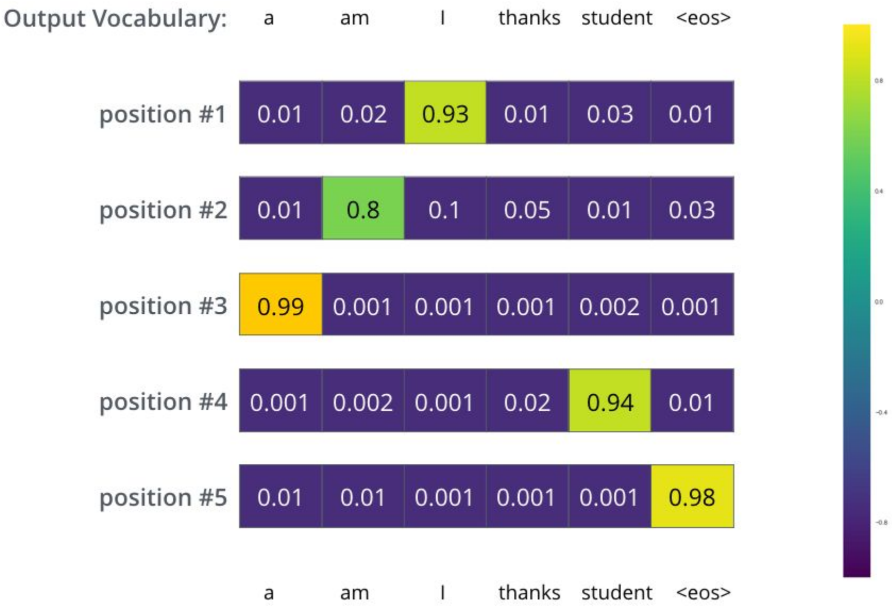
Loss Function

Target Model Outputs



Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0
	a	am	I	thanks	student	<eos>

Trained Model Outputs



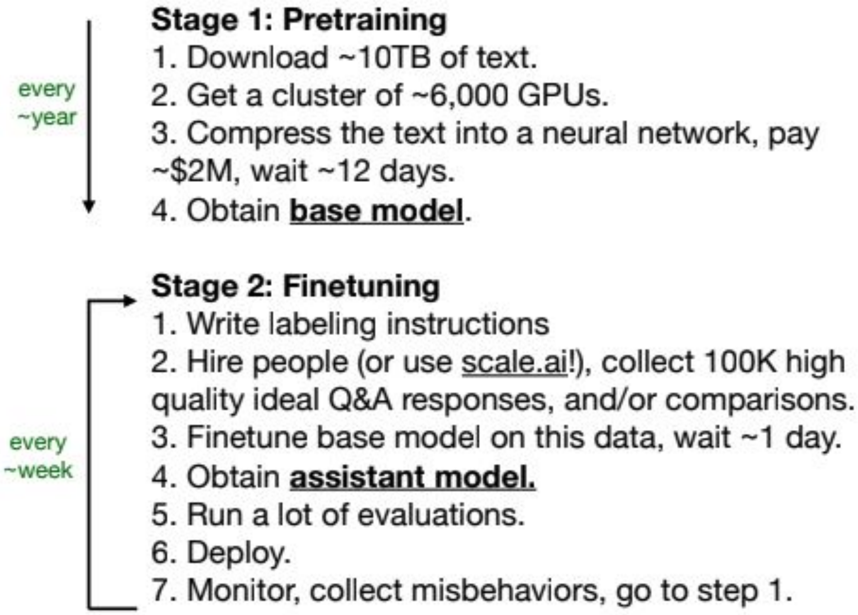
Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98
	a	am	I	thanks	student	<eos>

Google Developer Student Clubs
University of Toronto Mississauga

Finetuning

```
lookup.KeyValue
f.constant(['em
=tf.constant([0
.lookup.StaticV

_buckets=5)
```

What is Finetuning?



every ~year

Stage 1: Pretraining

1. Download ~10TB of text.
2. Get a cluster of ~6,000 GPUs.
3. Compress the text into a neural network, pay ~$2M, wait ~12 days.
4. Obtain **base model**.

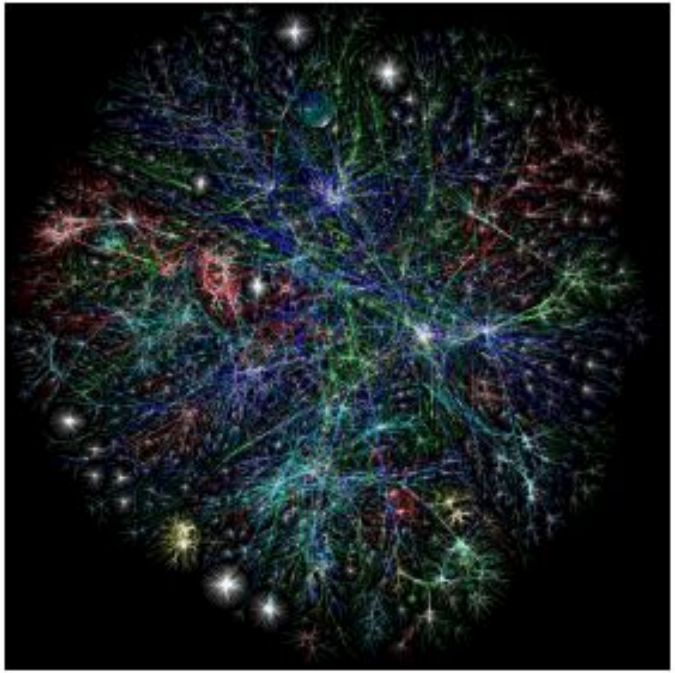
every ~week

Stage 2: Finetuning

1. Write labeling instructions
2. Hire people (or use scale.ai!), collect 100K high quality ideal Q&A responses, and/or comparisons.
3. Finetune base model on this data, wait ~1 day.
4. Obtain **assistant model.**
5. Run a lot of evaluations.
6. Deploy.
7. Monitor, collect misbehaviors, go to step 1.

Example

written by people:

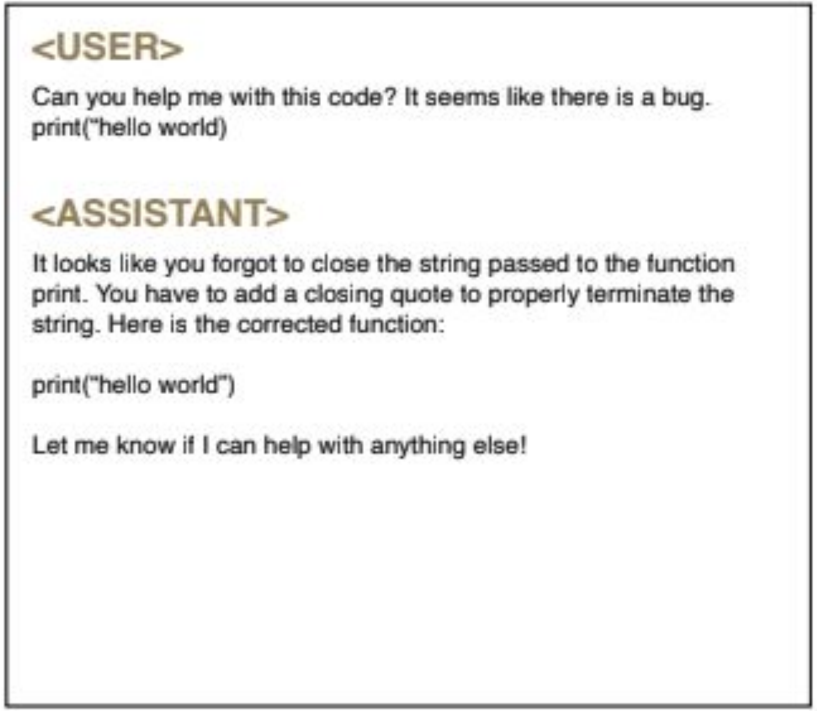


<USER>
Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research.

<ASSISTANT>
"Monopsony" refers to a market structure where there is only one buyer for a particular good or service. In economics, this term is particularly relevant in the labor market, where a monopsony employer has significant power over the wages and working conditions of their employees. The presence of a monopsony can result in lower wages and reduced employment opportunities for workers, as the employer has little incentive to increase wages or provide better working conditions…

~100K conversations

Results



<USER>

Can you help me with this code? It seems like there is a bug.
print("hello world)

<ASSISTANT>

It looks like you forgot to close the string passed to the function print. You have to add a closing quote to properly terminate the string. Here is the corrected function:

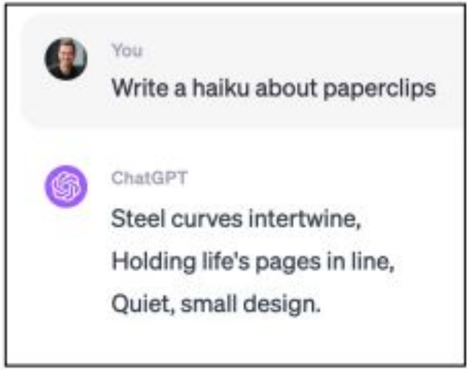
print("hello world")

Let me know if I can help with anything else!

Comparison Labeling

It is often much easier to compare Answers instead of writing Answers.

Simple example: it's much easier to spot a good haiku than it is to generate one:

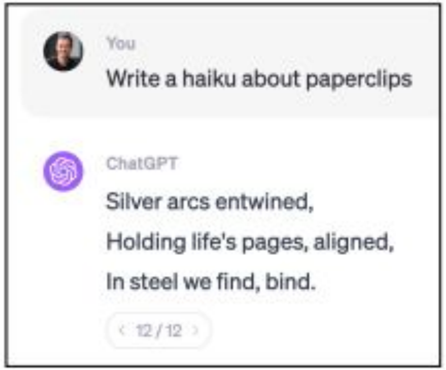


You

Write a haiku about paperclips

ChatGPT

Steel curves intertwine,
Holding life's pages in line,
Quiet, small design.

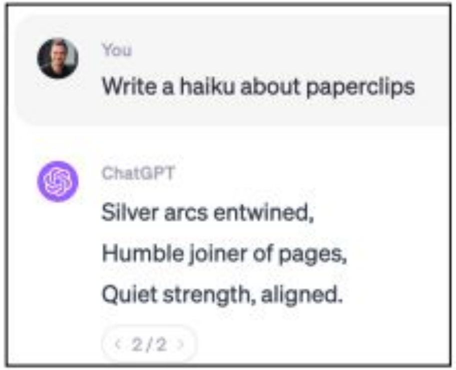


You

Write a haiku about paperclips

ChatGPT

Silver arcs entwined,
Holding life's pages, aligned,
In steel we find, bind.



You

Write a haiku about paperclips

ChatGPT

Silver arcs entwined,
Humble joiner of pages,
Quiet strength, aligned.

Google Developer Student Clubs
University of Toronto Mississauga

Scaling Laws

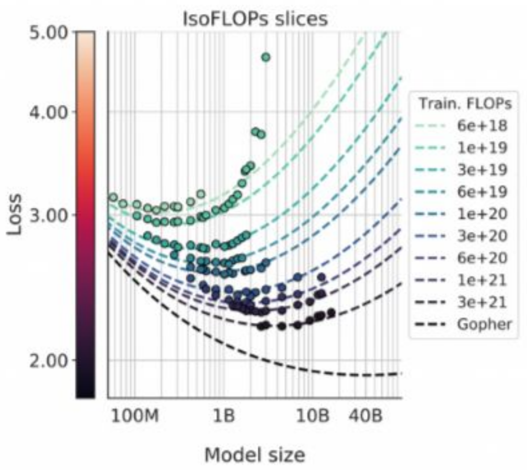
```
lookup.KeyValue
f.constant(['em
=tf.constant([0
.lookup.StaticV

_buckets=5)
```

What are Scaling Laws?

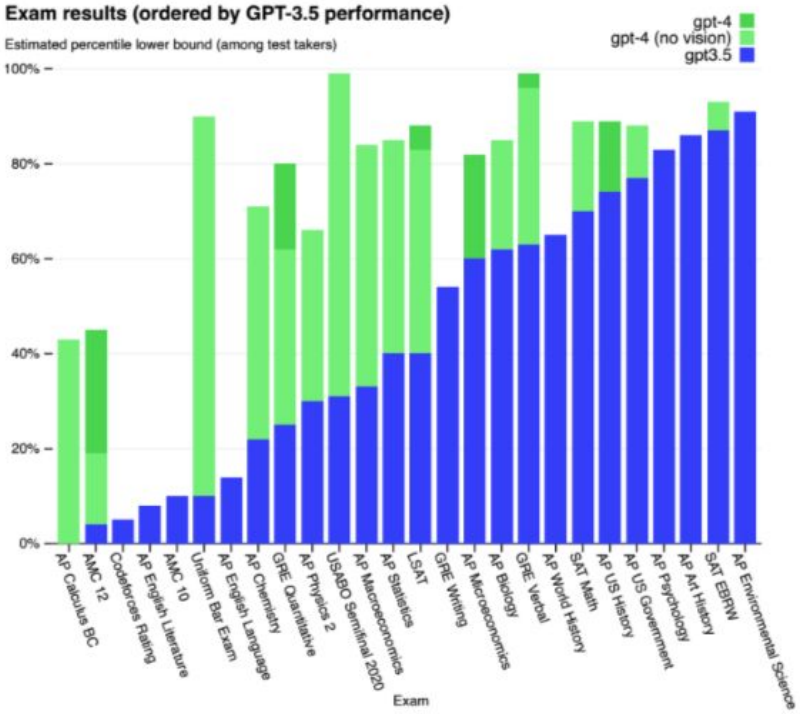
Performance of LLMs is a smooth, well-behaved, predictable function of:
- **N**, the number of parameters in the network
- **D**, the amount of text we train on

And the trends do not show signs of “topping out”

=> We can expect more intelligence “for free” by scaling



Example change in capabilities



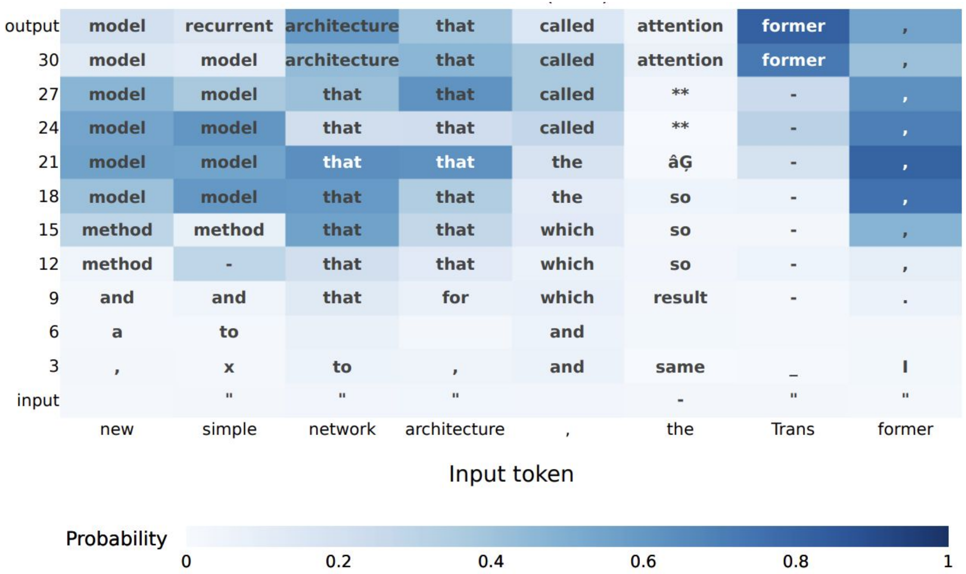
Google Developer Student Clubs
University of Toronto Mississauga

Conclusion

```
lookup.KeyValue
f.constant(['em
=tf.constant([0
.lookup.StaticV

_buckets=5)
```

We don’t know what goes on inside LLMs (yet)

- Simply, there's too many neurons to probe and deconstruct



output	model	recurrent	architecture	that	called	attention	former	,
30	model	model	architecture	that	called	attention	former	,
27	model	model	that	that	called	**	-	,
24	model	model	that	that	called	**	-	,
21	model	model	that	that	the	âĠ	-	,
18	model	model	that	that	the	so	-	,
15	method	method	that	that	which	so	-	,
12	method	-	that	that	which	so	-	,
9	and	and	that	for	which	result	-	.
6	a	to			and			
3	,	x	to	,	and	same	_	I
input		"	"	"		-	"	"
	new	simple	network	architecture	,	the	Trans	former

Input token

LLMs may or may not be better than humans

- Prior work in AI (eg. AlphaZero from Deepmind trained with RL) have outperformed humans, but does that imply LLMs will follow this path?
- LLMs aren’t humans they don’t learn like humans
- Eg. LLMs are very good at predicting the next token, which humans for the most part aren’t

Google Developer Student Clubs
University of Toronto Mississauga

Thank you!

```
lookup.KeyValue
f.constant(['em
=tf.constant([0
.lookup.StaticV

_buckets=5)
```