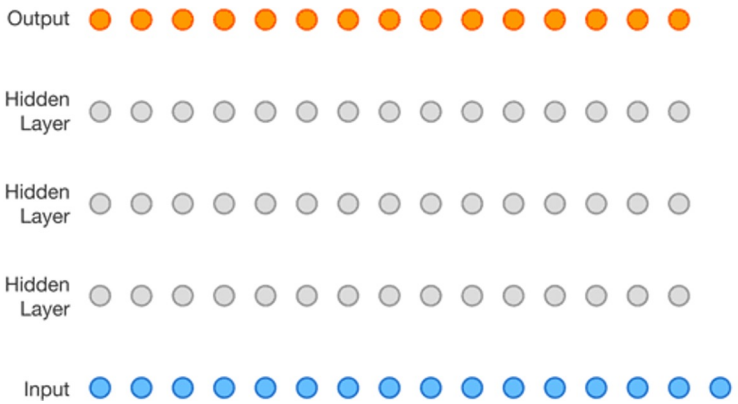
10707
Deep Learning

Russ Salakhutdinov

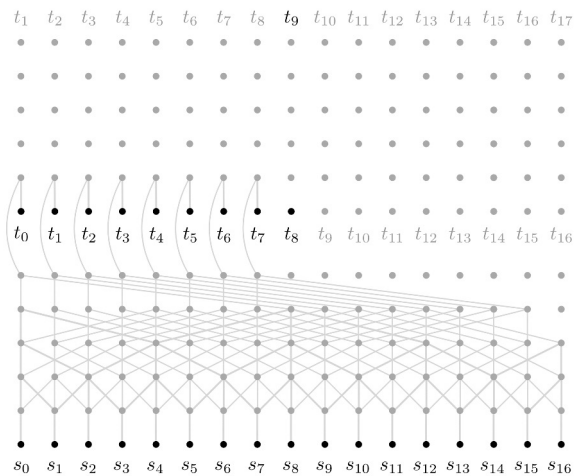
Machine Learning Department
rsalakhu@cs.cmu.edu

Sequence Model / Transformers

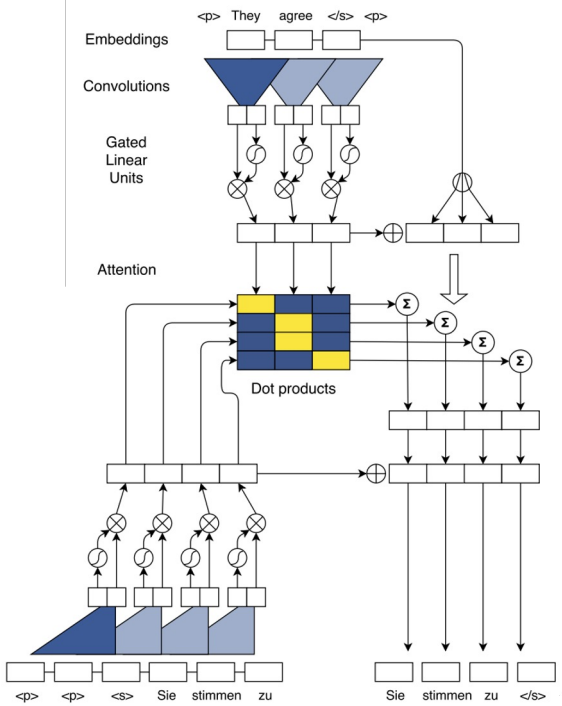
New Architectures



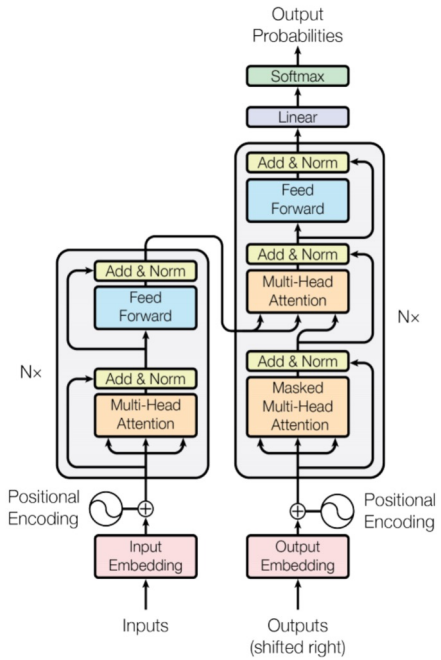
Wavenet, van den Oord, et al, 2016



Bytenet, Kalchbrenner, et al, 2016



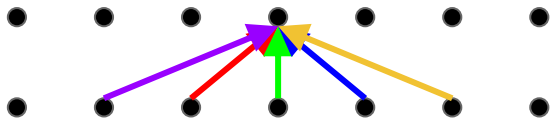
Conv seq2seq, Gehring, et al, 2017



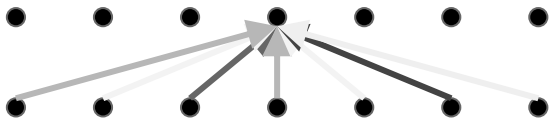
Att is all you need, Vaswani, et al, 2017

Self-Attention

Convolution

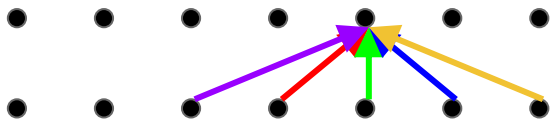


Self-Attention

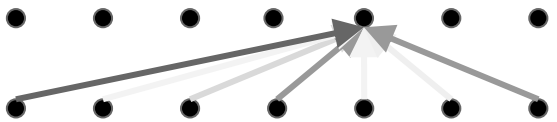


Self-Attention

Convolution

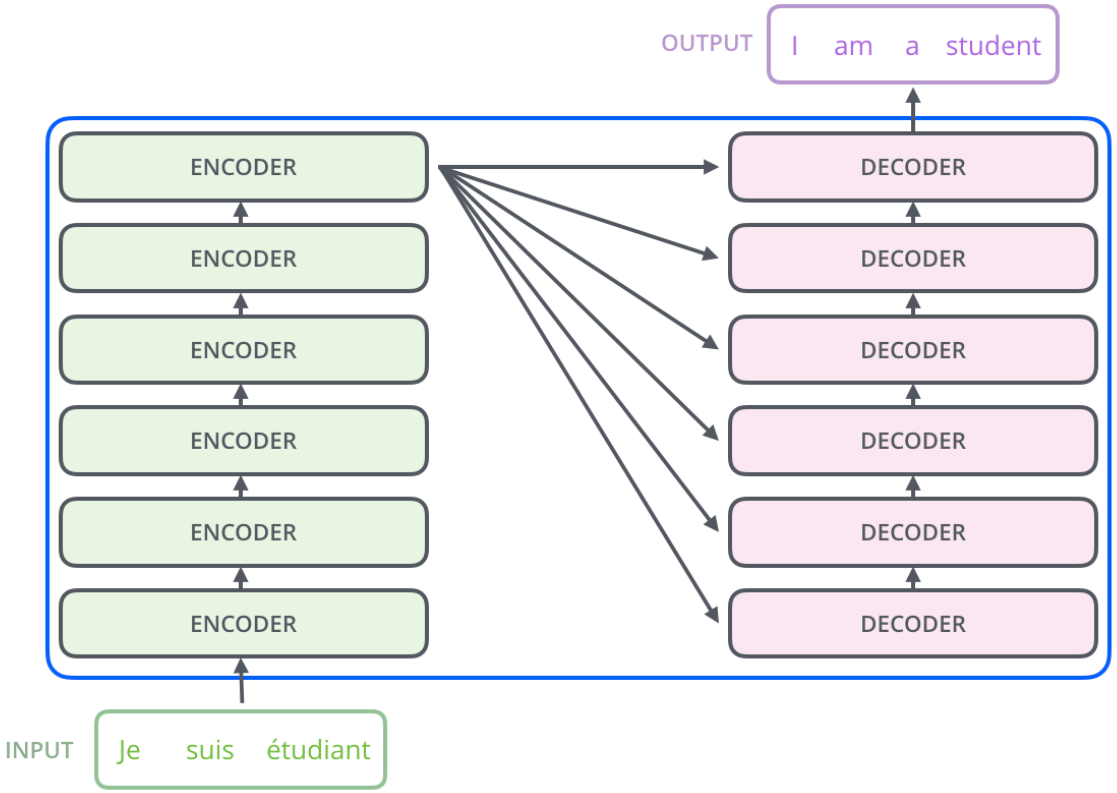


Self-Attention



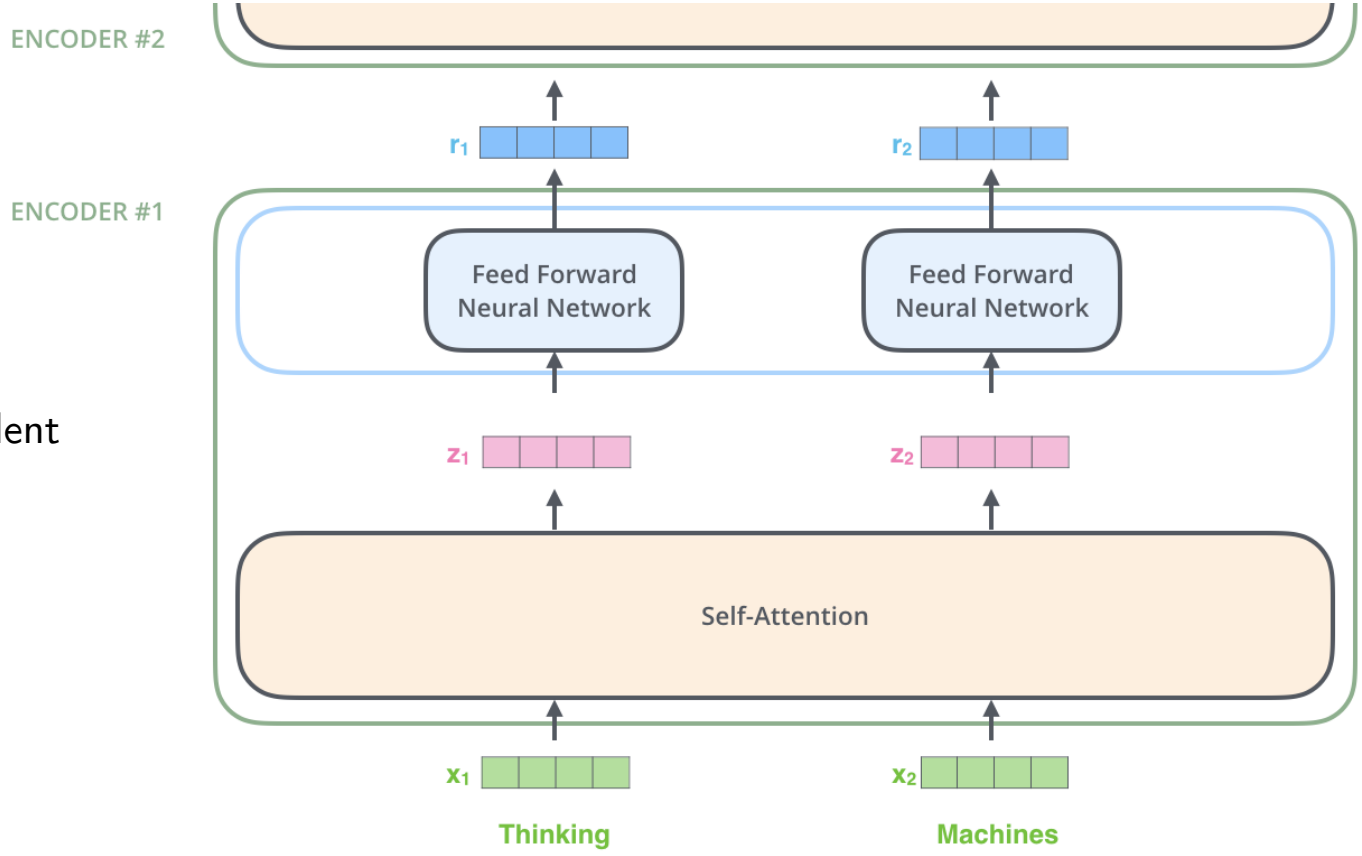
Transformer

- Example of Machine Translation



Transformer

- An Encoder Block: same structure, different parameters



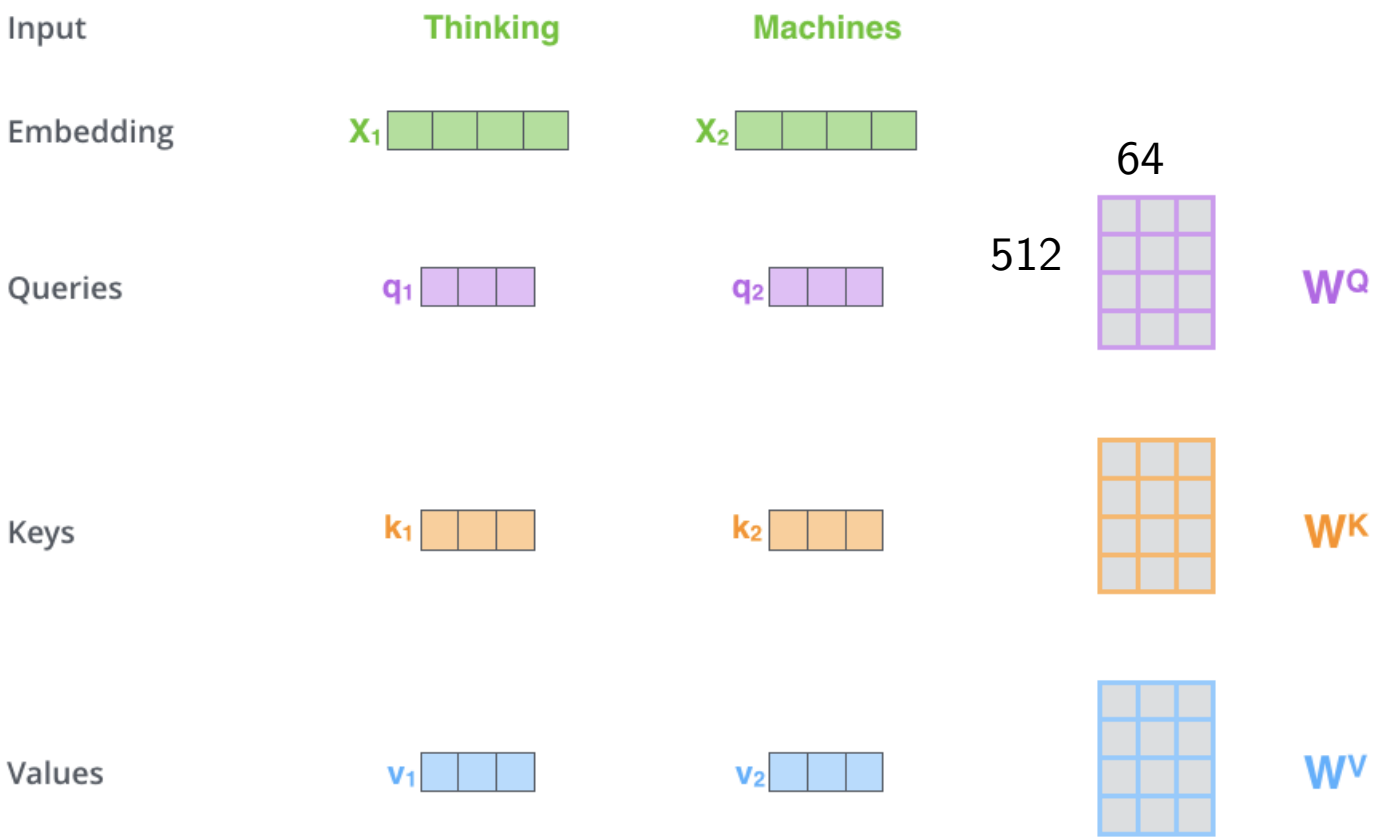
Note: The ffnn is independent for each word.
Hence can be parallelized

Self-Attention

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

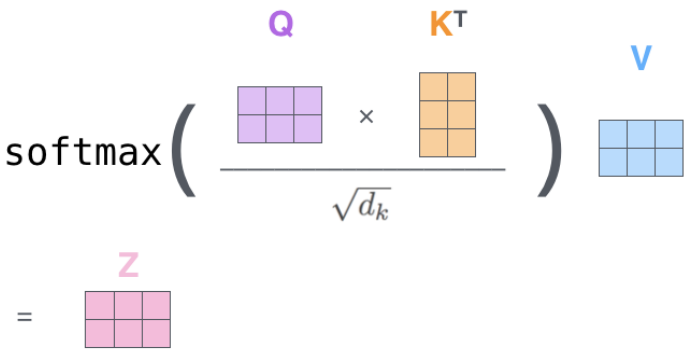
$q_i = x_i W^Q$
$K_i = x_i W^K$
$V_i = x_i W^V$

Embedding dimensions are chosen by the modeler.



Self-Attention

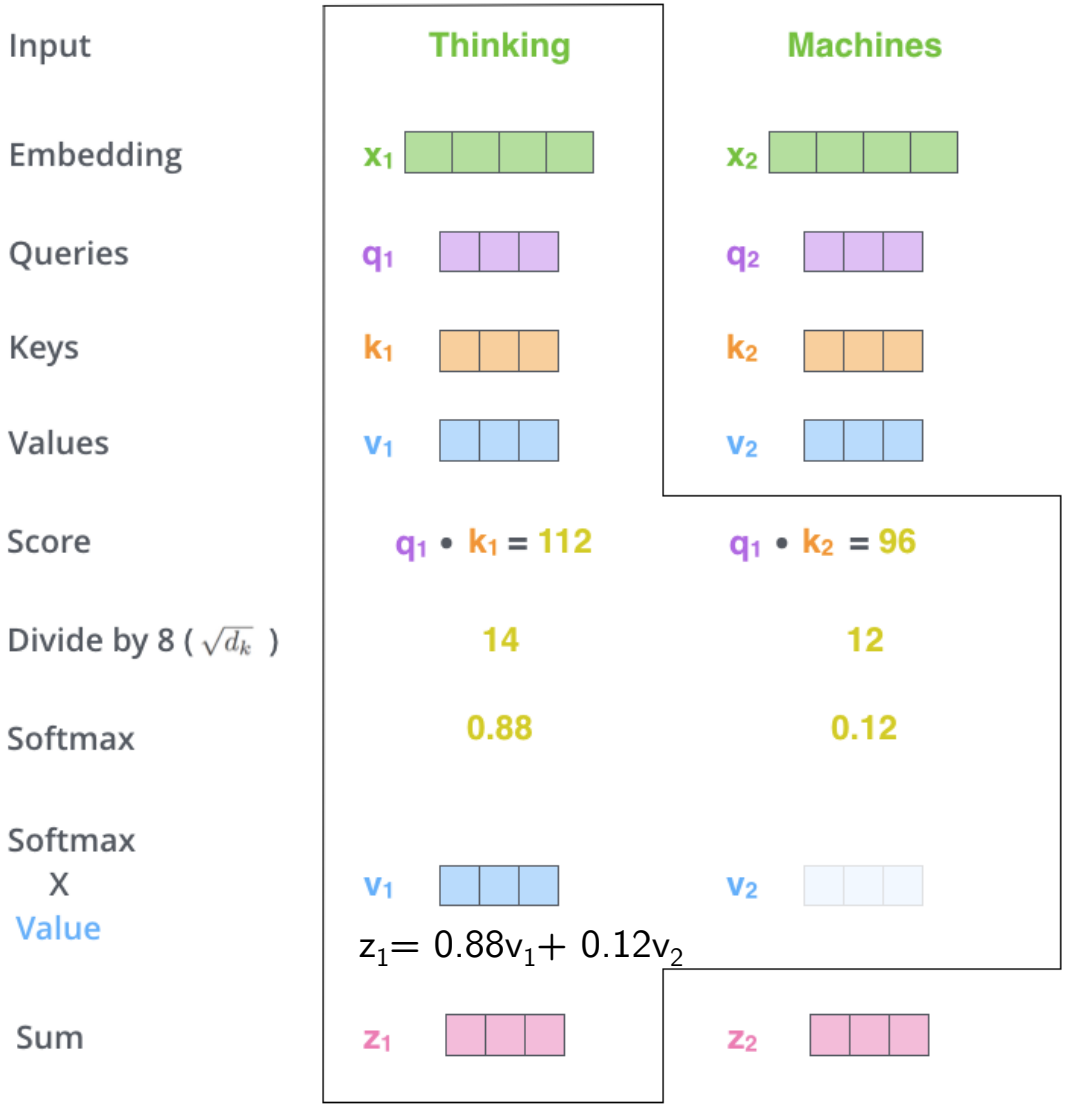
- Formula



d_k=64 is dimension of key vector

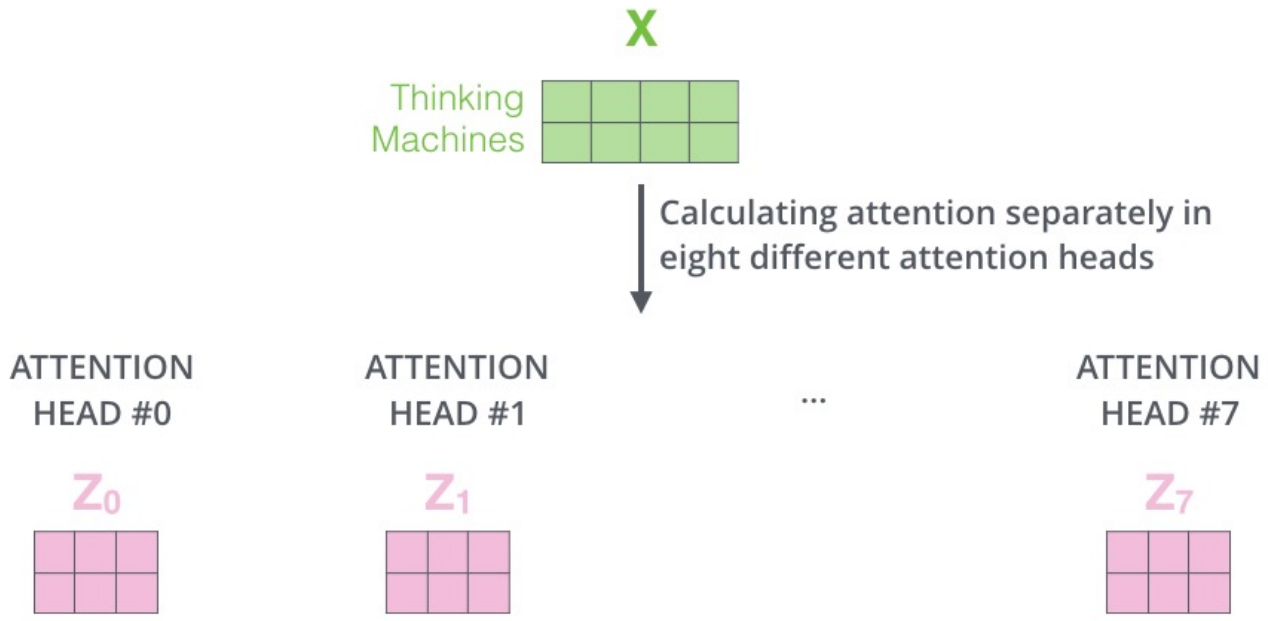
Th input is a 2x512 matrix

The output is a 2x64 matrix



Multiple Heads

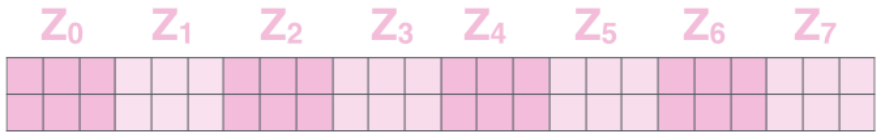
- Expands the model's ability to focus on different positions.
- It gives the attention layer multiple "representation subspaces"



Multiple Heads

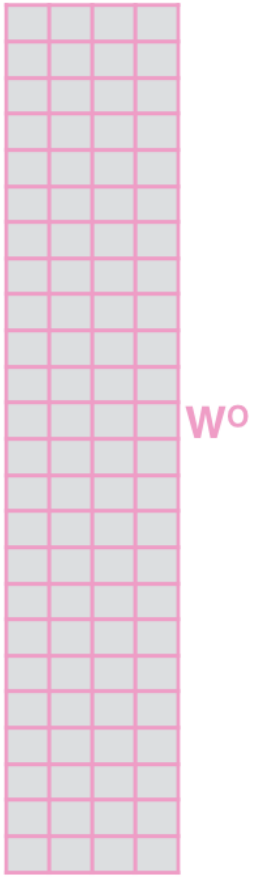
- Aggregate multiple Zs across different heads.

1) Concatenate all the attention heads



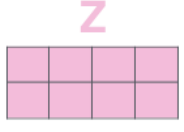
2) Multiply with a weight matrix W^O that was trained jointly with the model

X



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

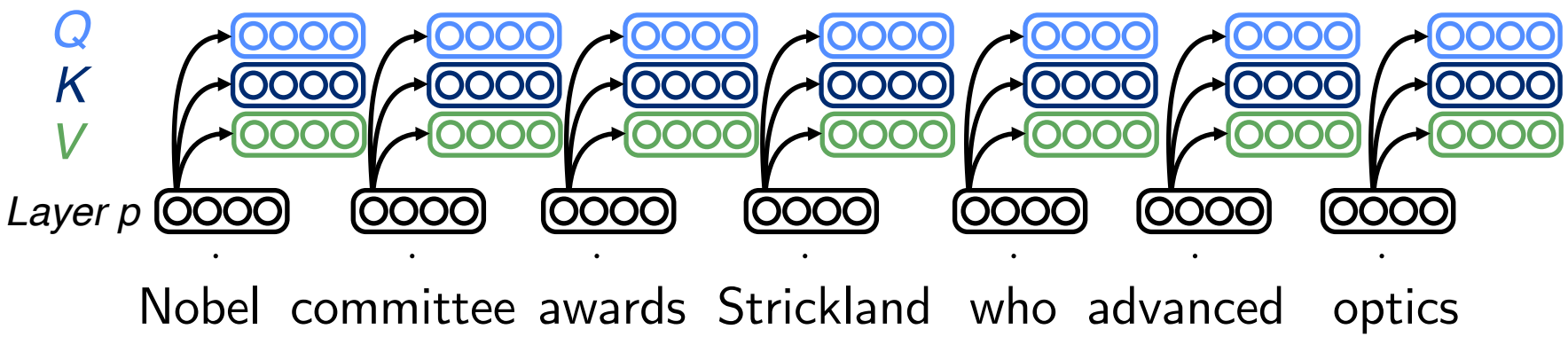
=



The output is a 2x64 matrix

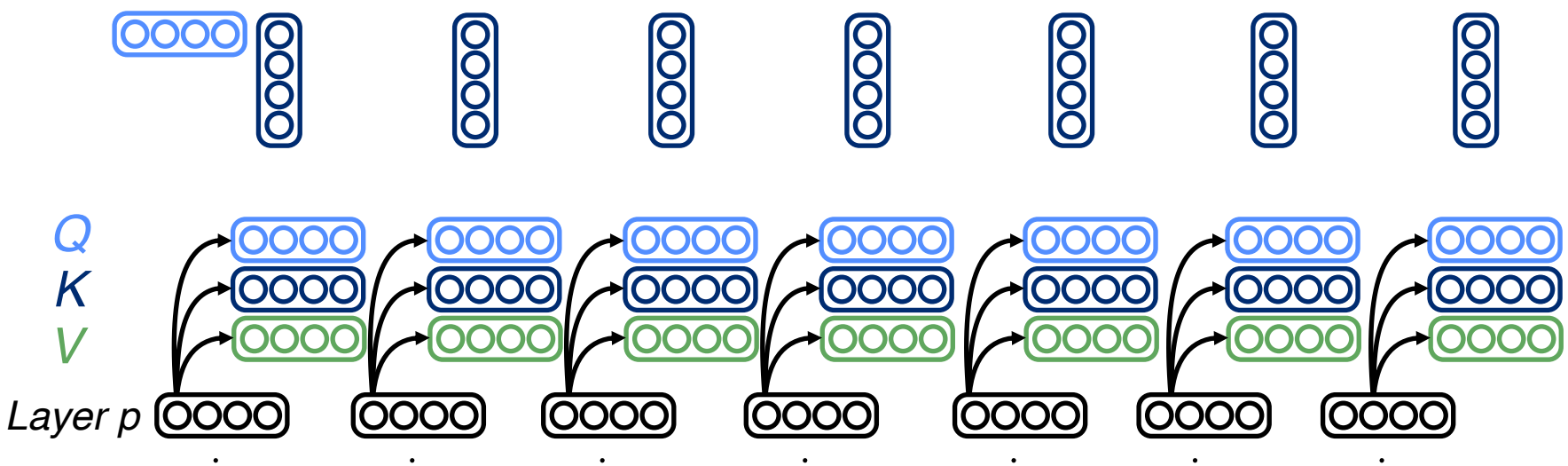
Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



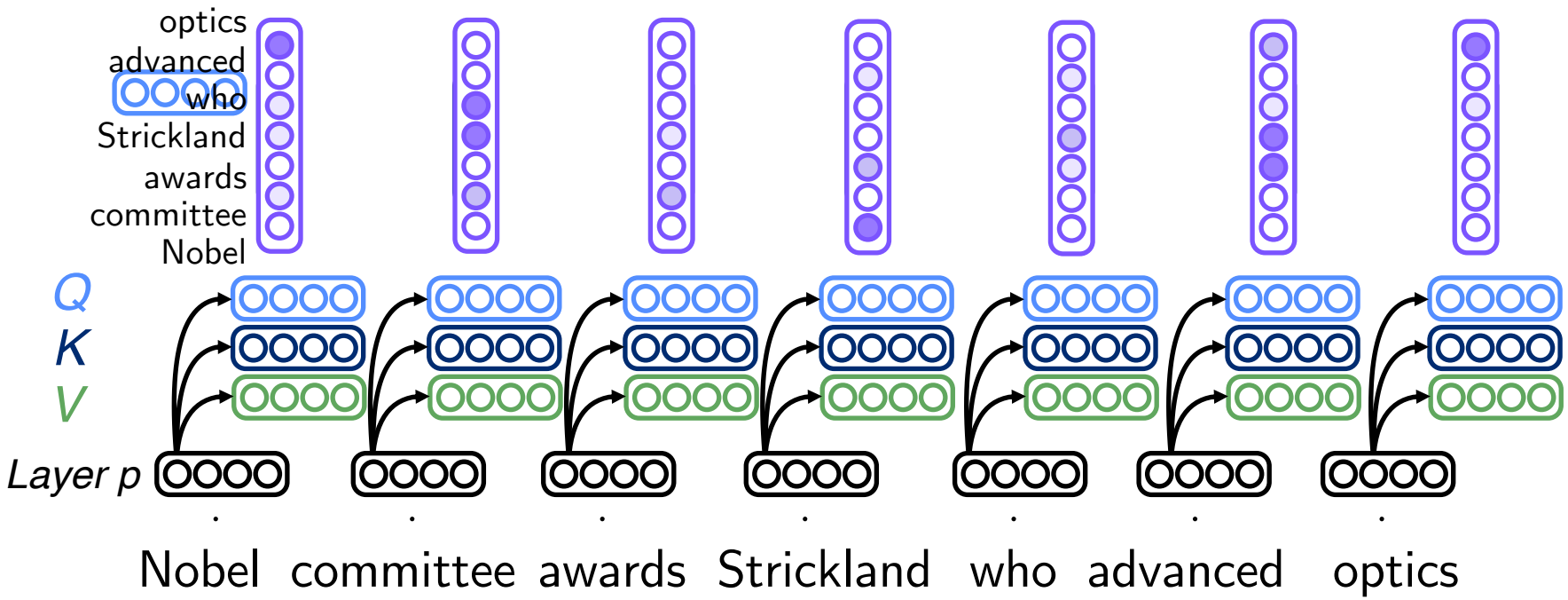
Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



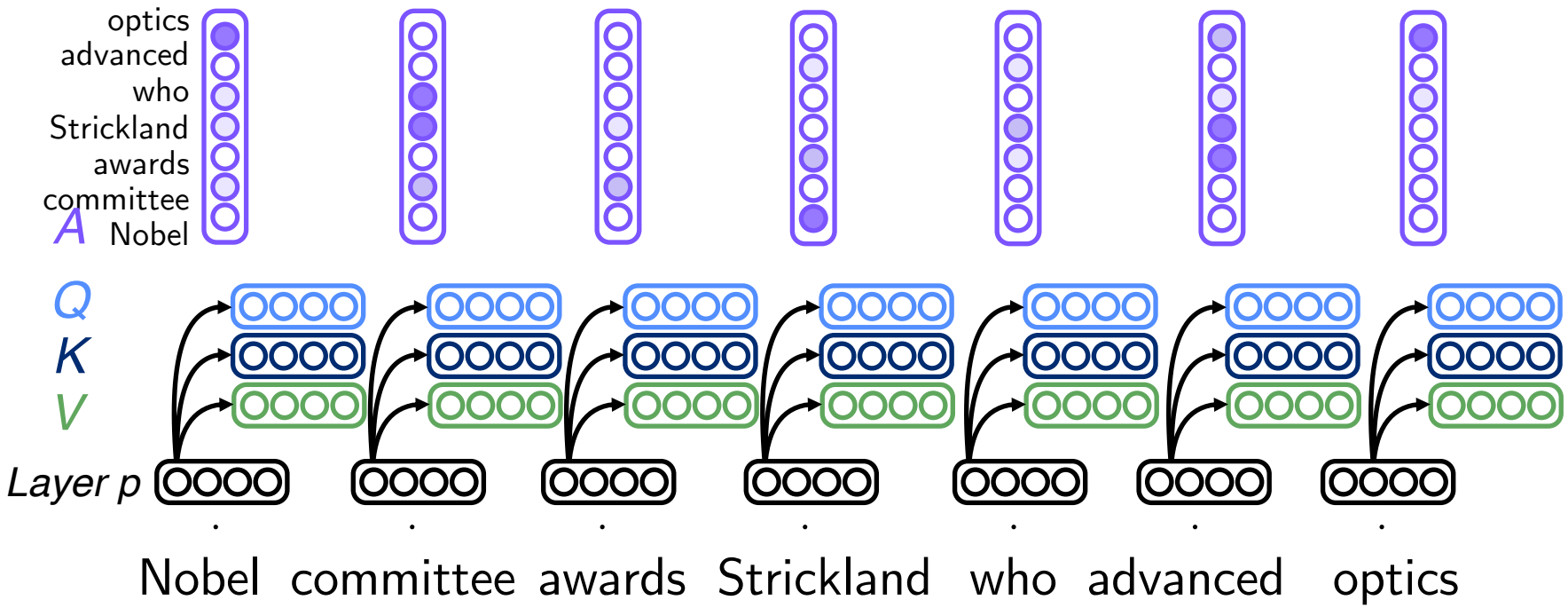
Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



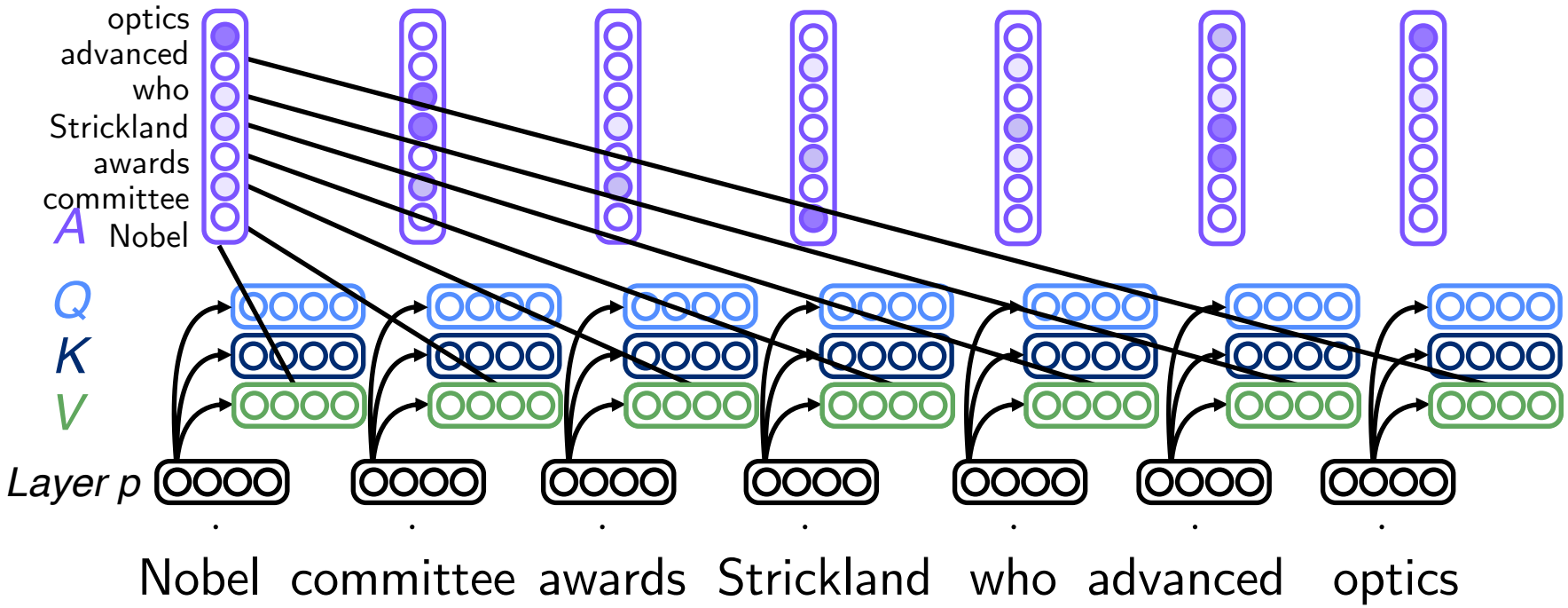
Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



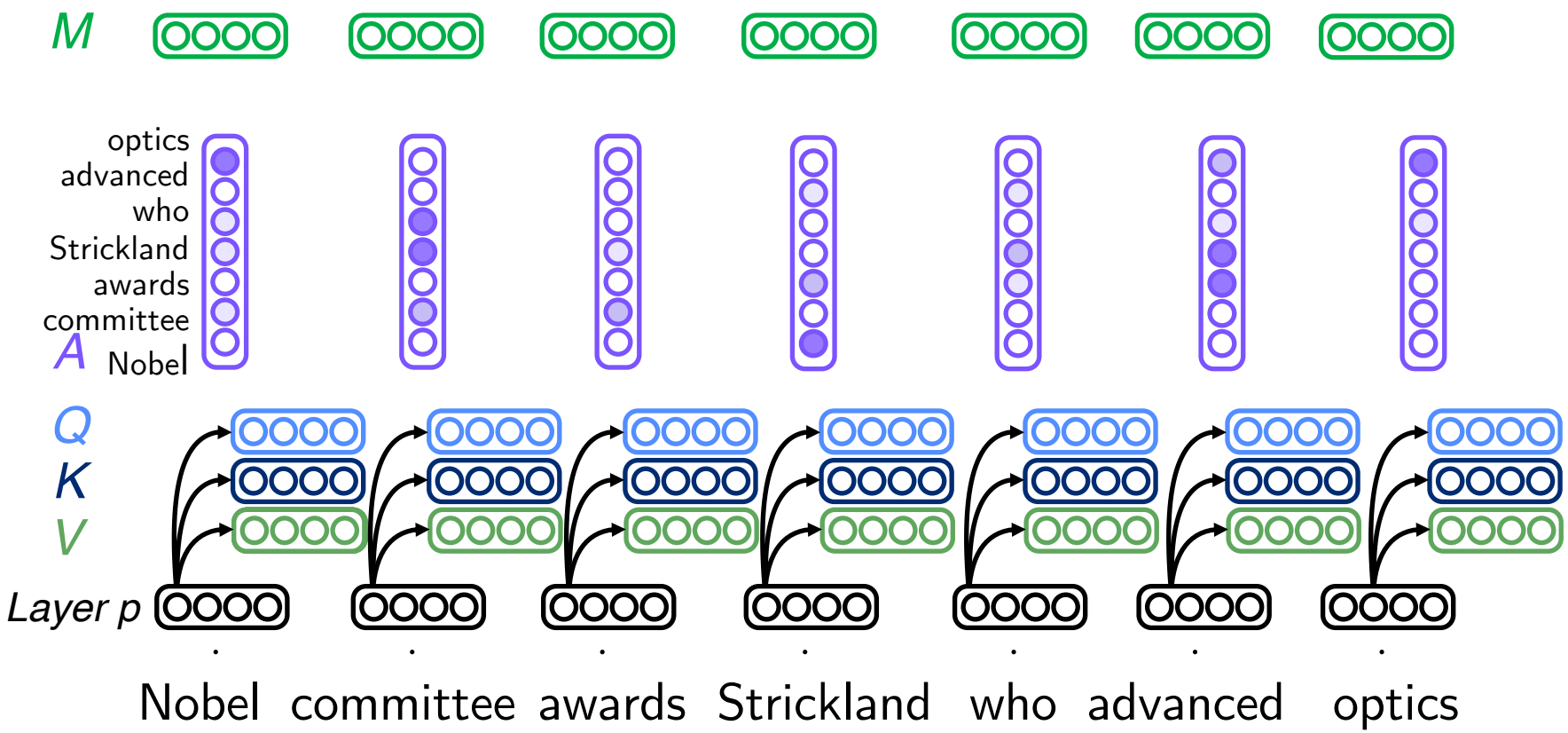
Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



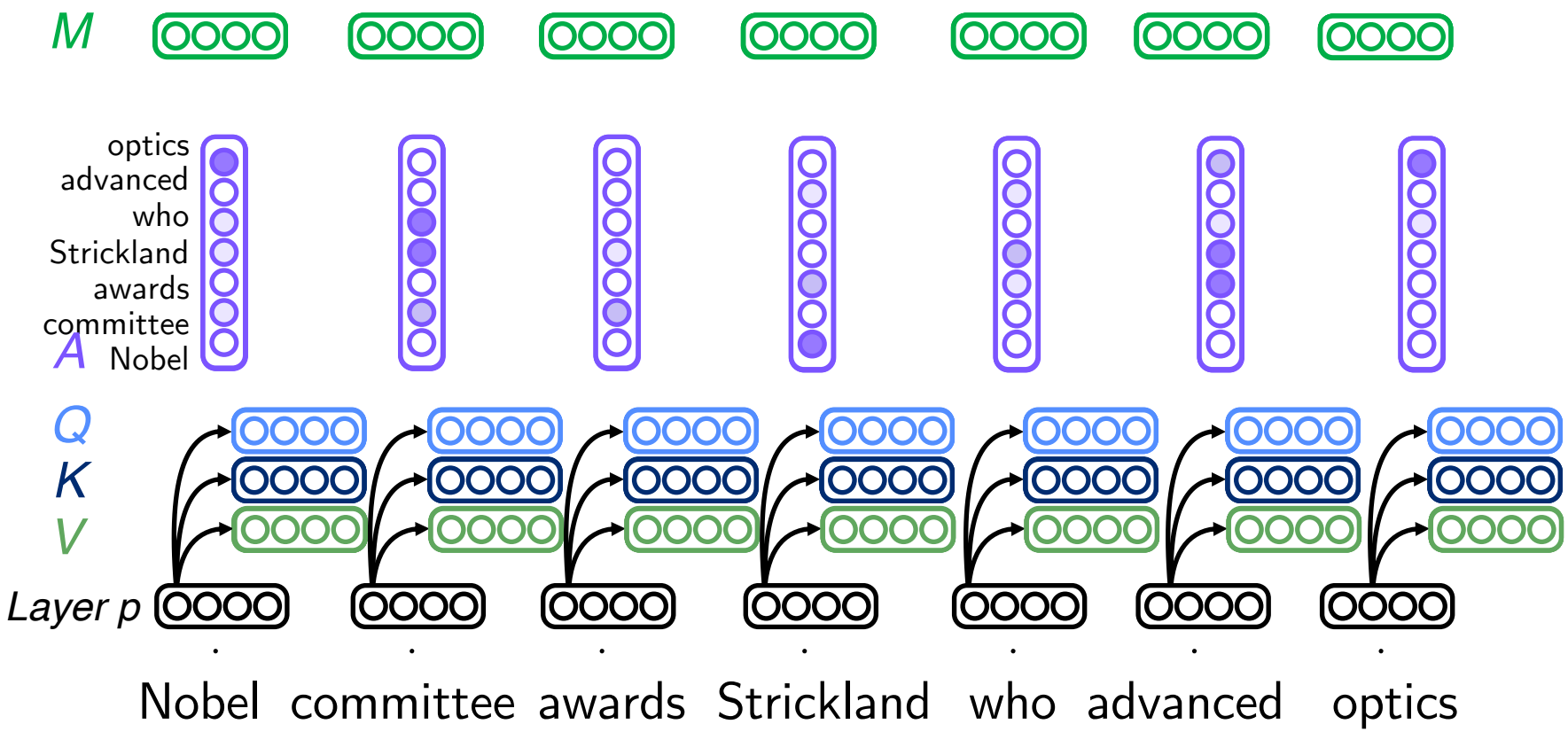
Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



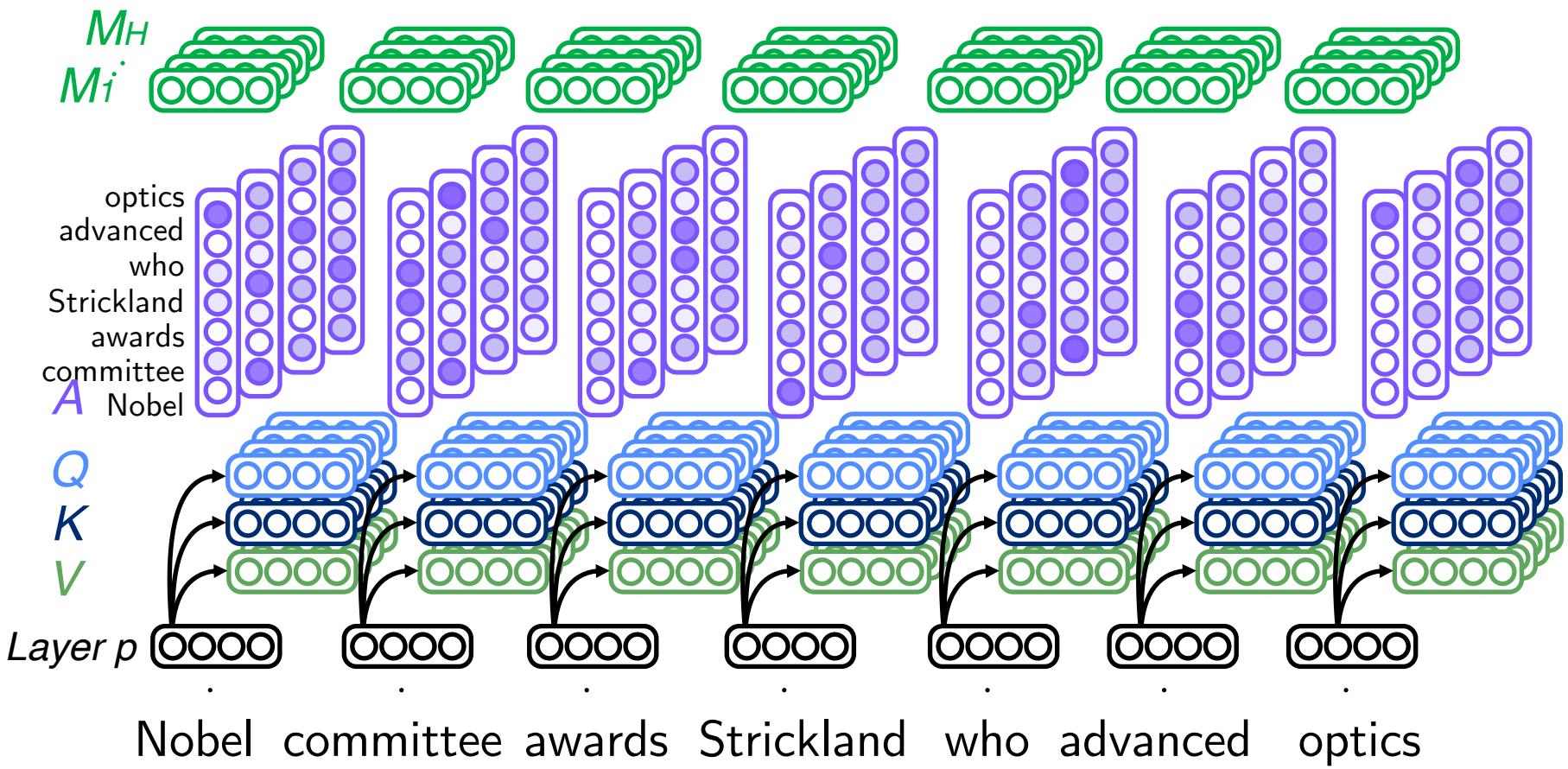
Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



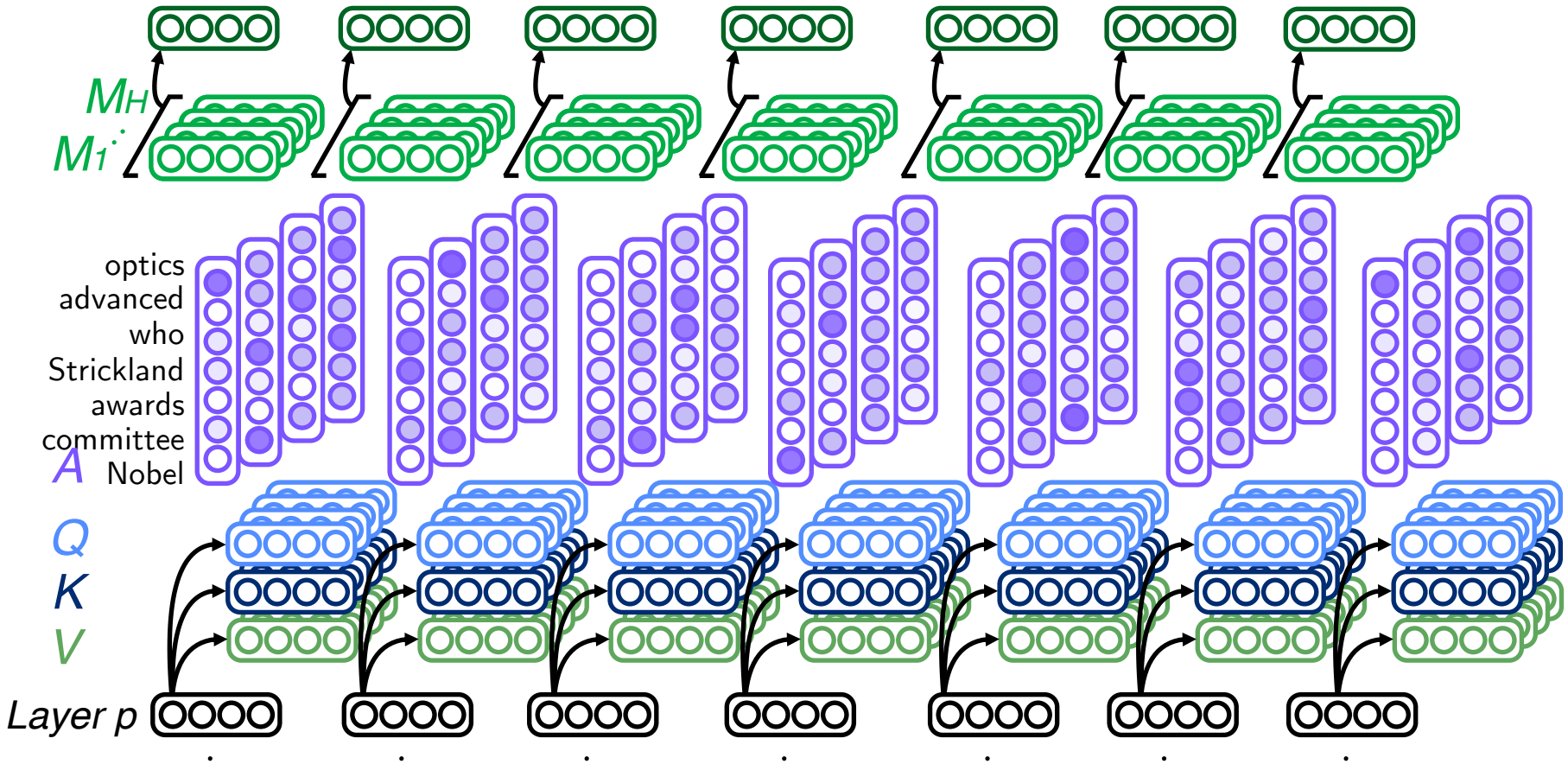
Multi-Head Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



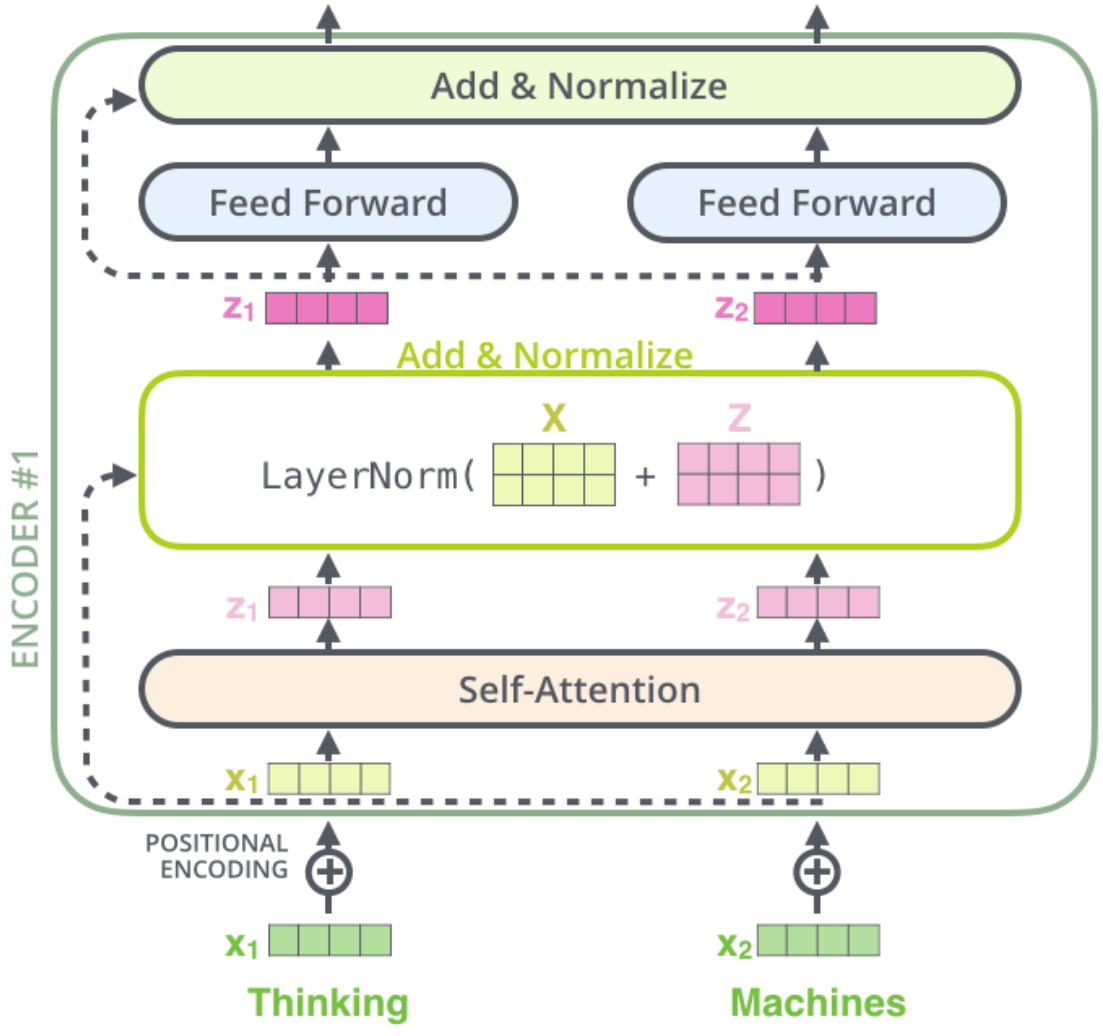
Multi-Head Self Attention

[Vaswani et al. 2017],
Slides borrowed form
Emma Strubell



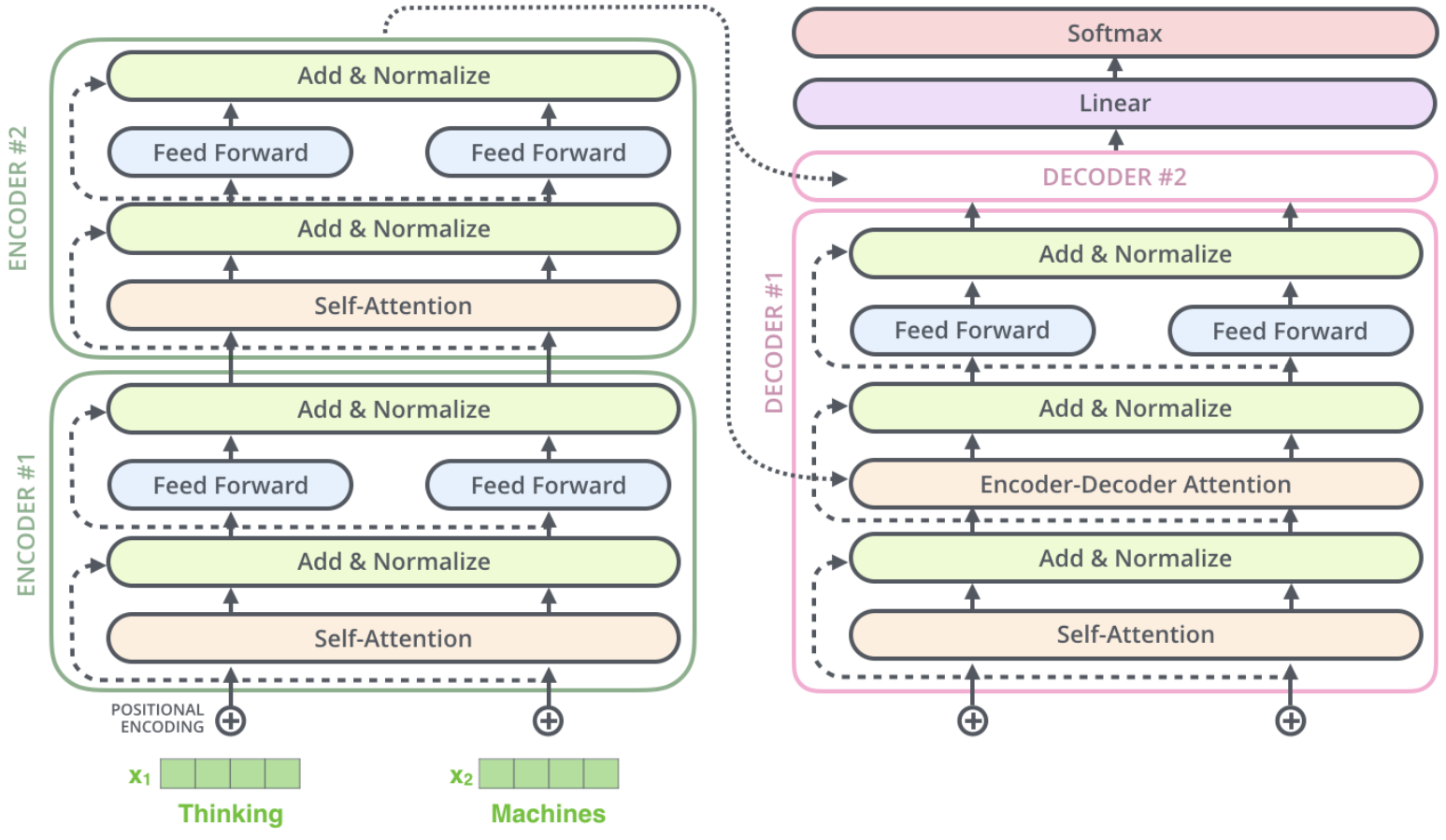
Transformer Block

- Sometimes one would normalize each layer so that each feature (column) have the same mean and standard deviation.



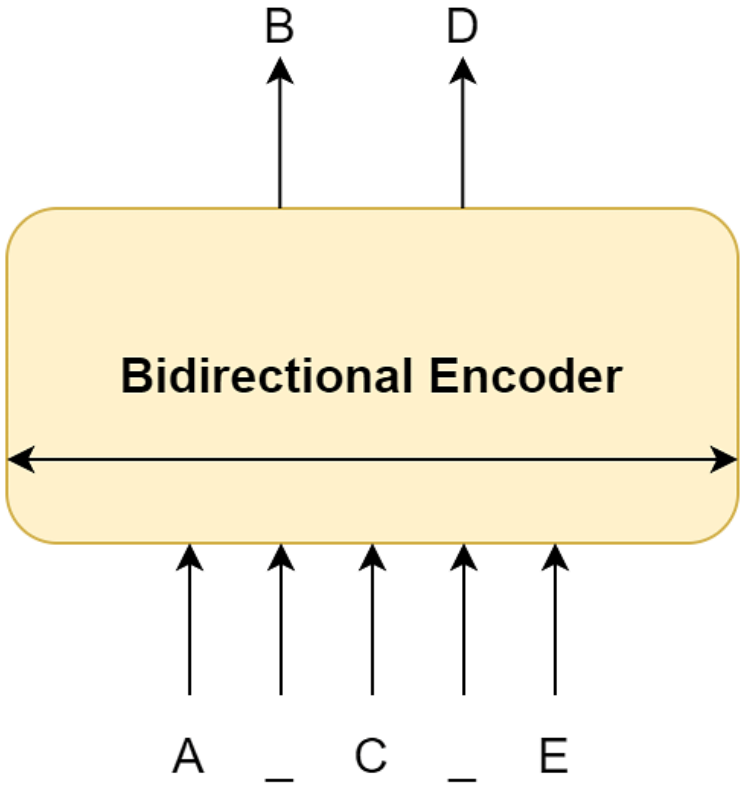
Full Model: Machine Translation

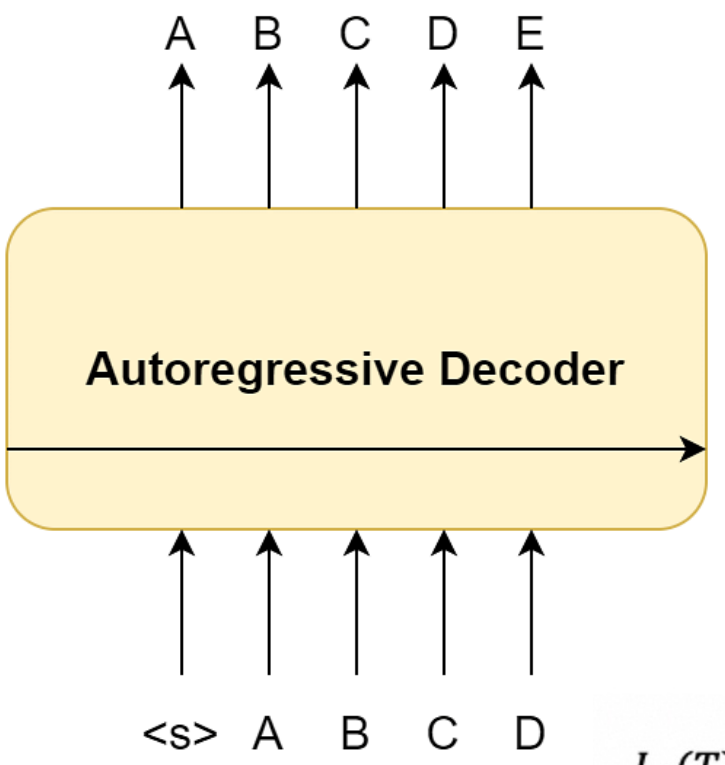
- The encoder-decoder attention is just like self attention, except it uses K, V from the top of encoder output, and its own Q



Encoder vs Decoder

- Encoders (e.g. BERT) learn representations of input (by predicting masked input) – bi-directional
- Decoders (e.g. GPT-3) are autoregressive decoders: uni-directional

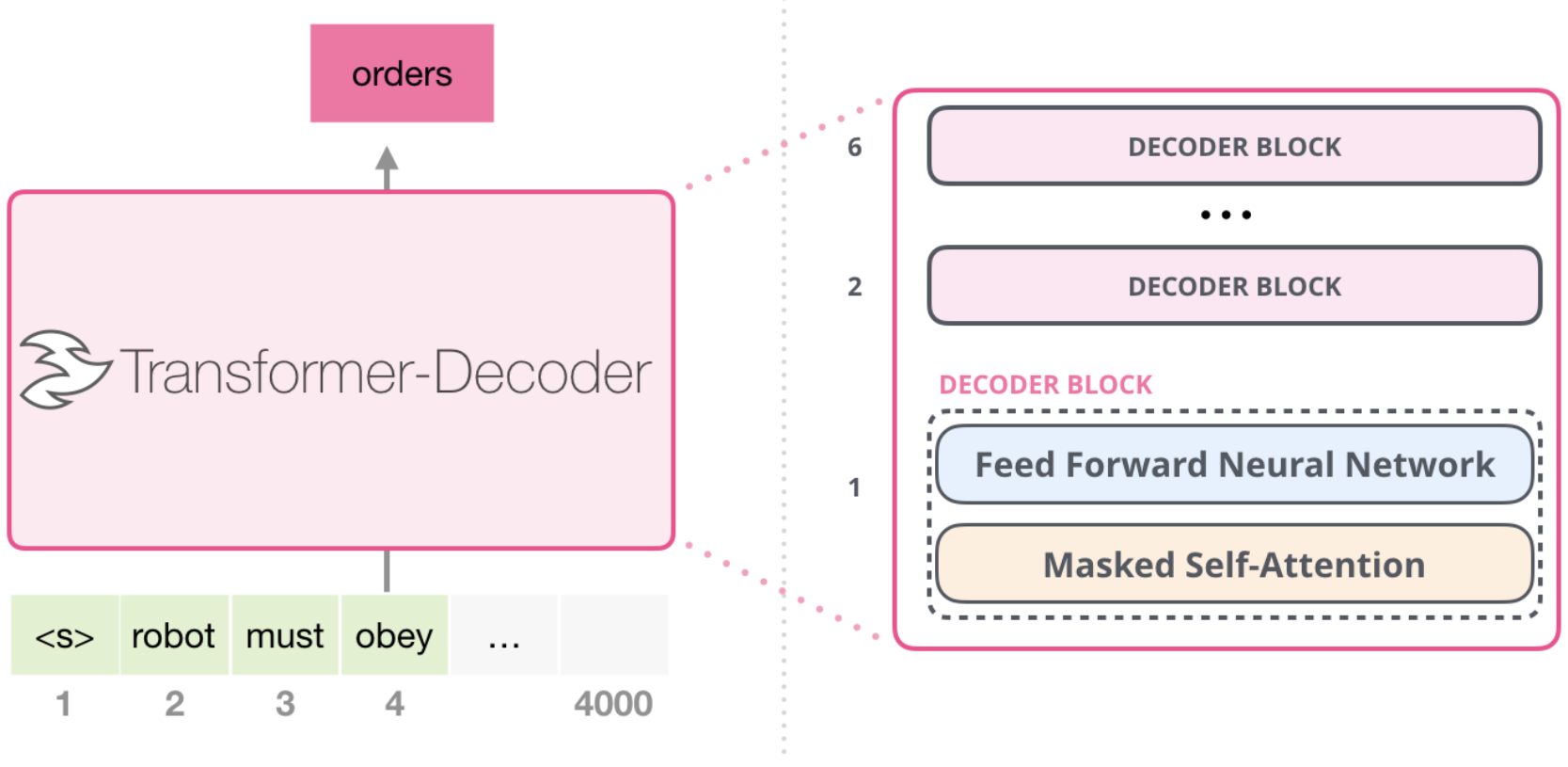




$$L_1(T) = \sum_i log\, P(t_i|t_{i-k}, \ldots, t_{i-1}; \theta)$$

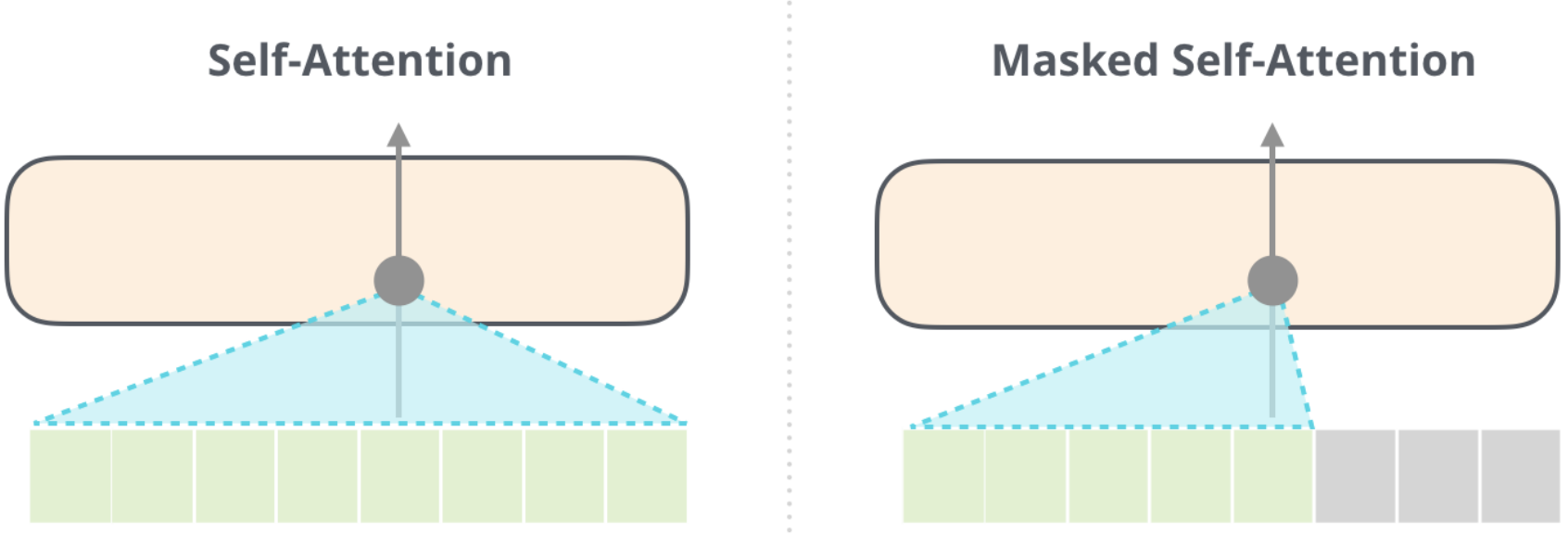
Decoder-based Only: GPT

- Predict the next token in a sequence
- Rely on masked self-attention
  - In decoder, the input is "incomplete" when calculating self-attention
  - The solution is to set future unknown values with "-inf".



Decoder-based Only: GPT

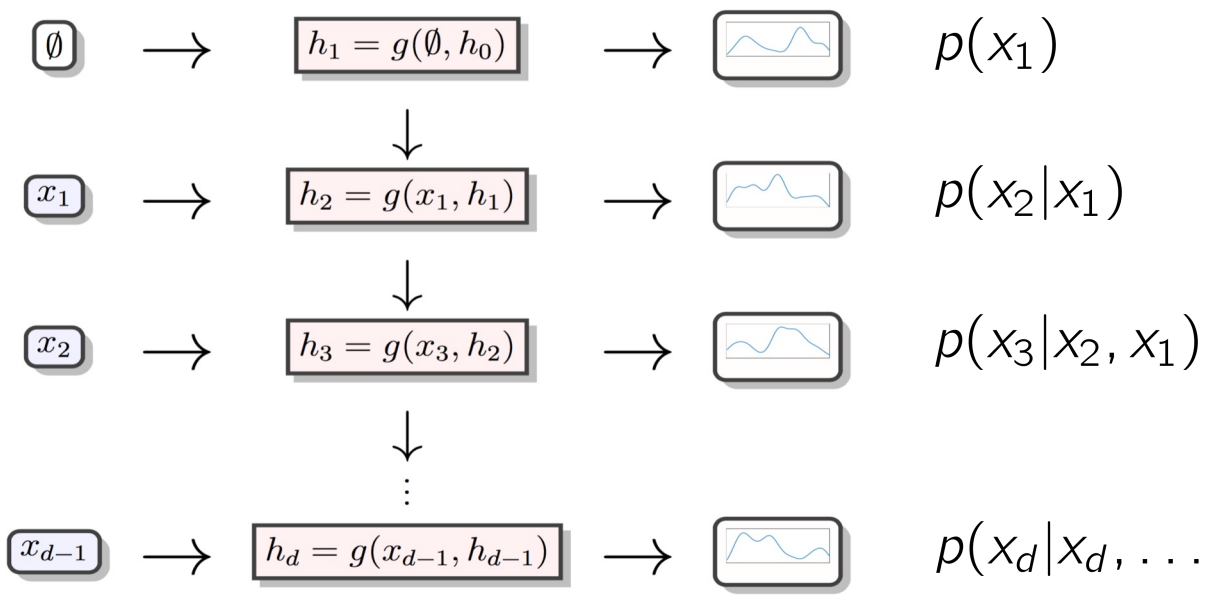
- Masked Self-Attention (to compute more efficiently)



Autoregressive Models

- Density Estimation by Autoregression

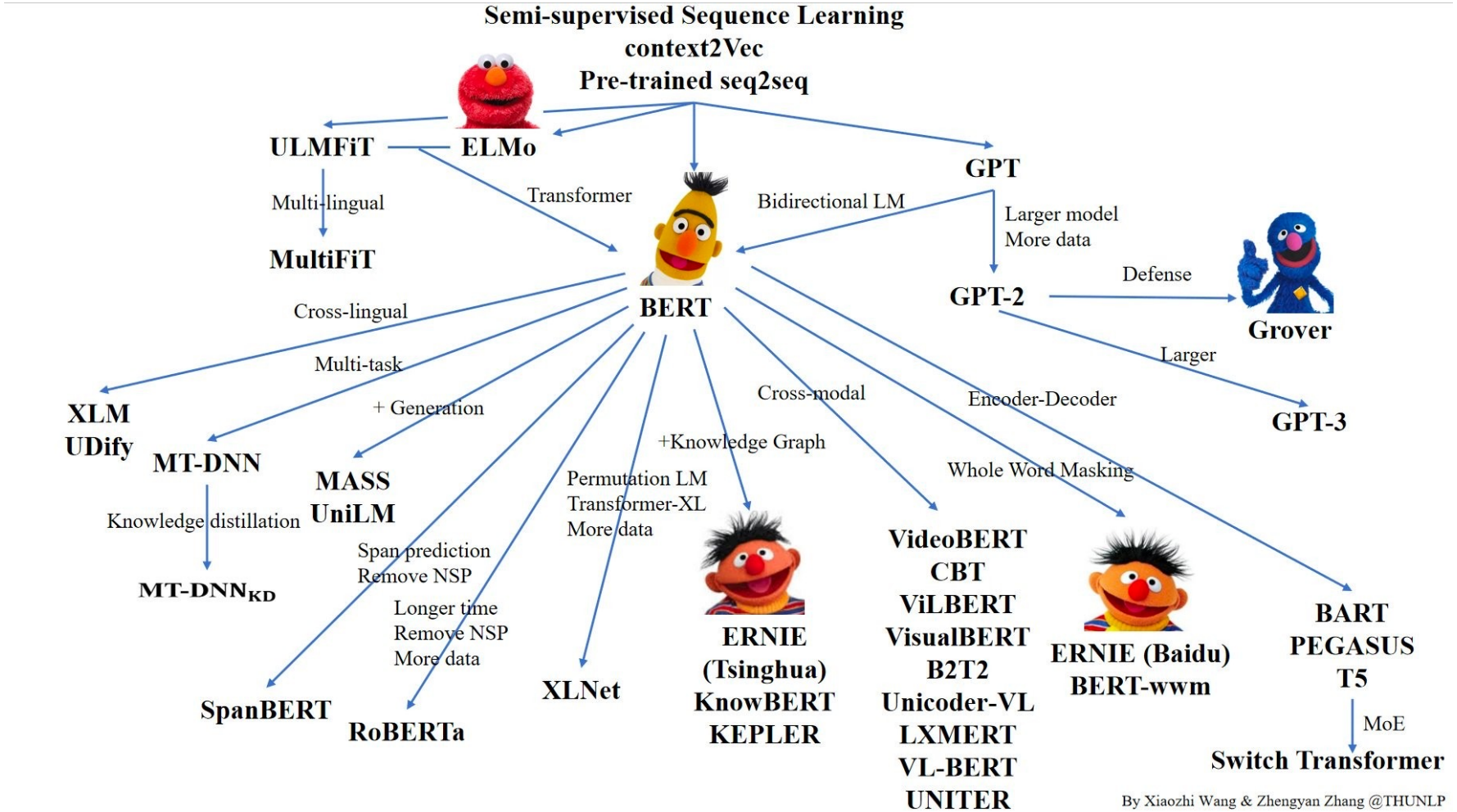
$$p(x_1, \ldots, x_d) = \prod_{i=1}^{d} p(x_i | x_{i-1}, \ldots, x_1) \approx \prod_{i=1}^{d} p(x_i | g(x_{i-1}, \ldots, x_1))$$



Each conditional can be a deep Transformer model

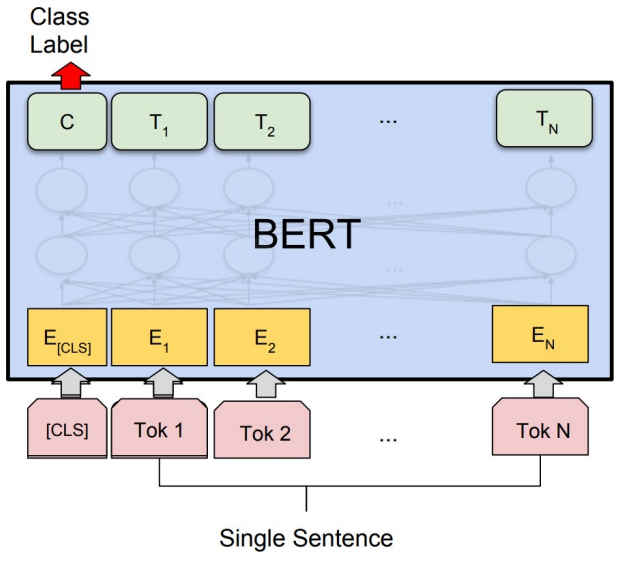
NADE (Uria 2013), MADE (Germain 2017), MAF (Papamakarios 2017), PixelCNN (van den Oord, et al, 2016)

Pre-Trained Large Language Models (LLMs)

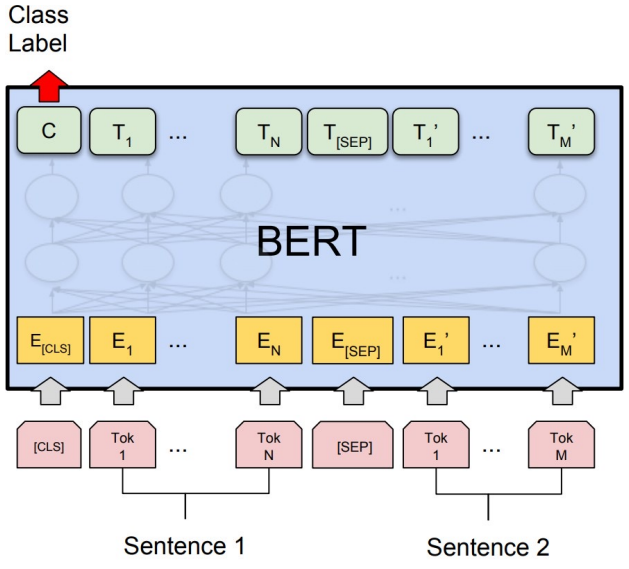


(by Sep 2019)

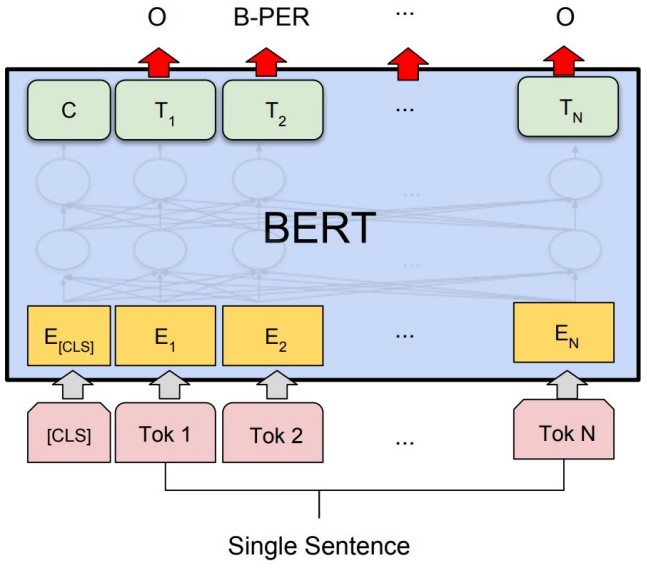
BERT Model



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



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



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

- CLS token is used to provide classification decisions
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

GPT/GPT2

- Train a single unidirectional transformer LM on long contexts

- GPT2: trained on 40GB of text collected from upvoted links from reddit

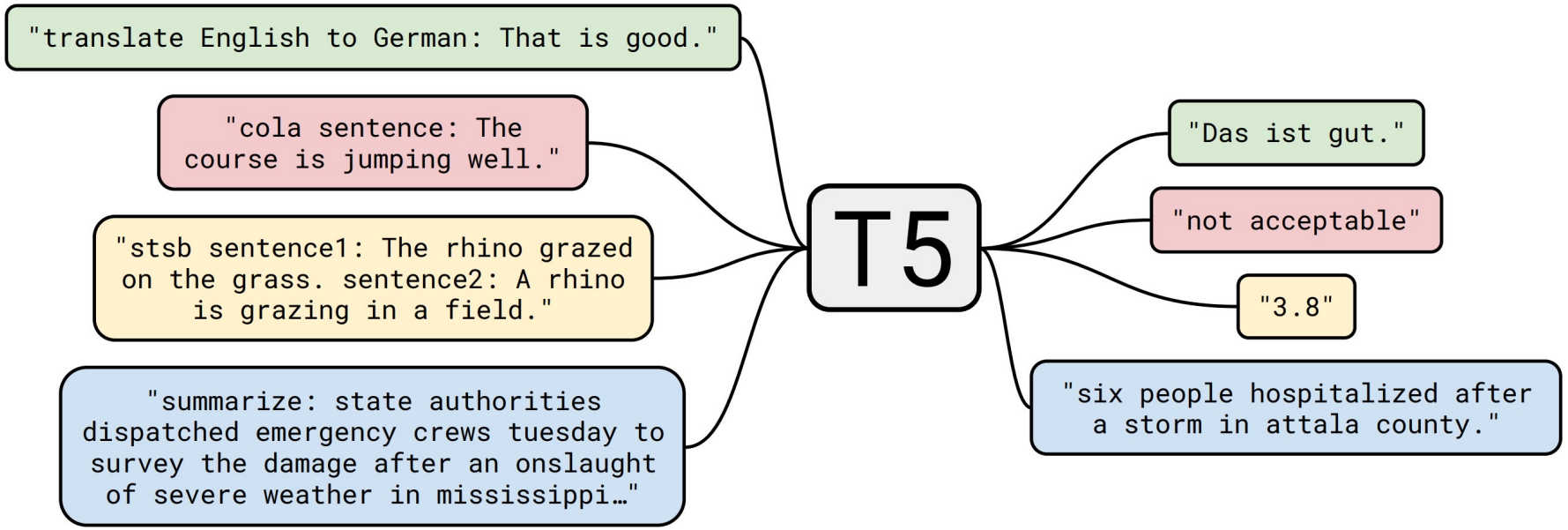
- 1.5B parameters — by far the largest of these models trained as of March 2019

- Because it's a language model, we can generate from it

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

T5

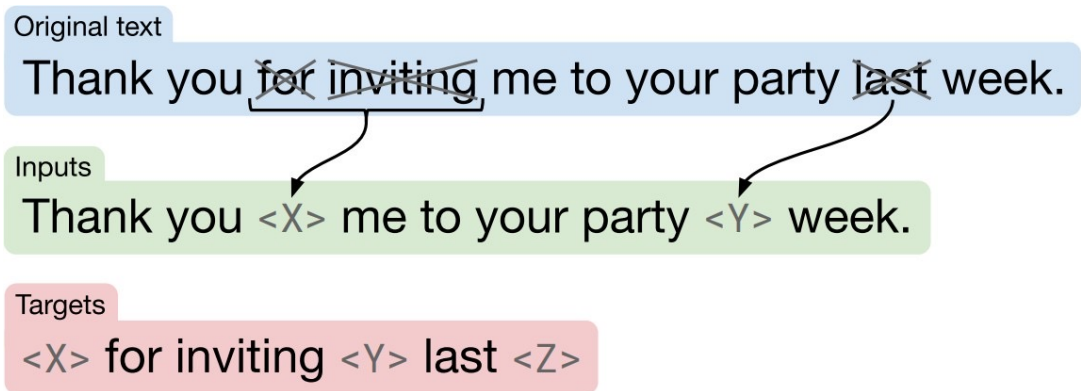
- Frame many problems as sequence-to-sequence ones:



Raffel et al. (2019)

T5

- Pre-training: similar denoising scheme to BERT



Raffel et al. (2019)

T5

- Colossal Cleaned Common Crawl: 750 GB of text
- We still haven't hit the limit of bigger data being useful for pre- training:

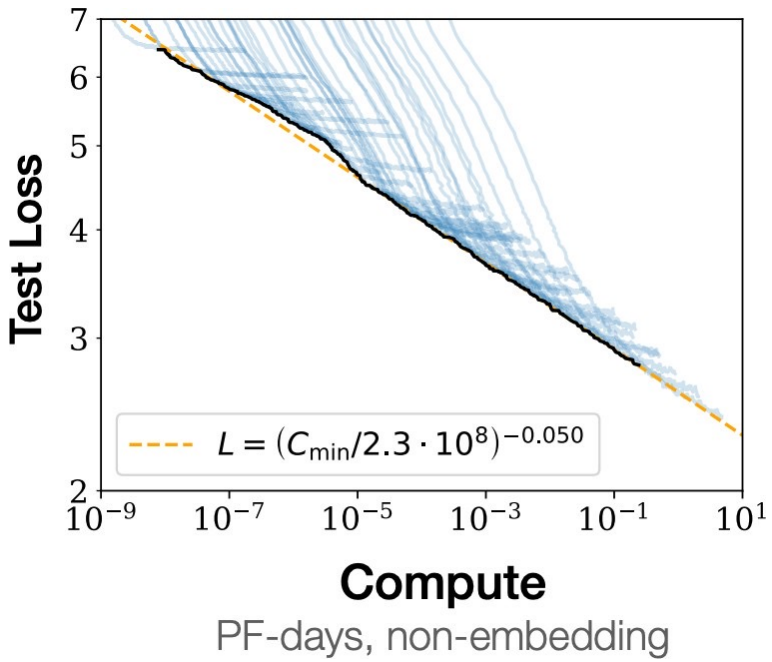
we see stronger MT results from the biggest data

Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full dataset	0	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

Raffel et al. (2019)

Scaling Laws

- Each model is a different-sized LM (GPT-style)
- With more compute, larger models get further down the loss “frontier”
- Building a bigger model (increasing compute) will decrease test loss!



1 petaflop/s-day is equivalent to 8 V100 GPUs at full efficiency of a day

Kaplan et al. (2020)

Scaling Laws

- The scaling laws suggest how to set model size, dataset size, and training time for big datasets

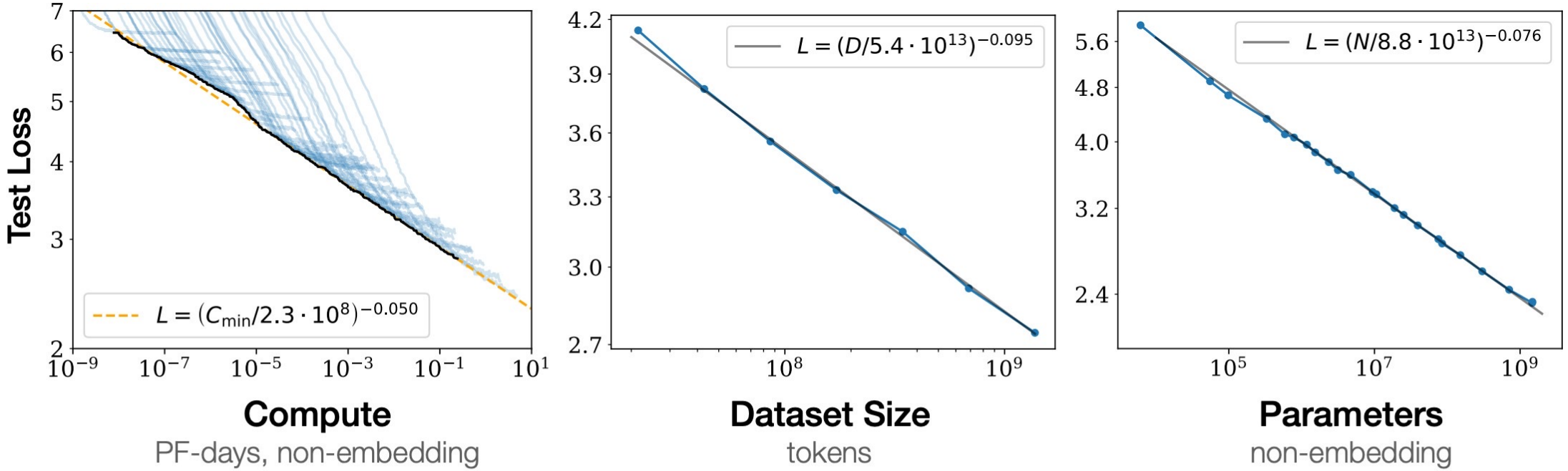


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

GPT3

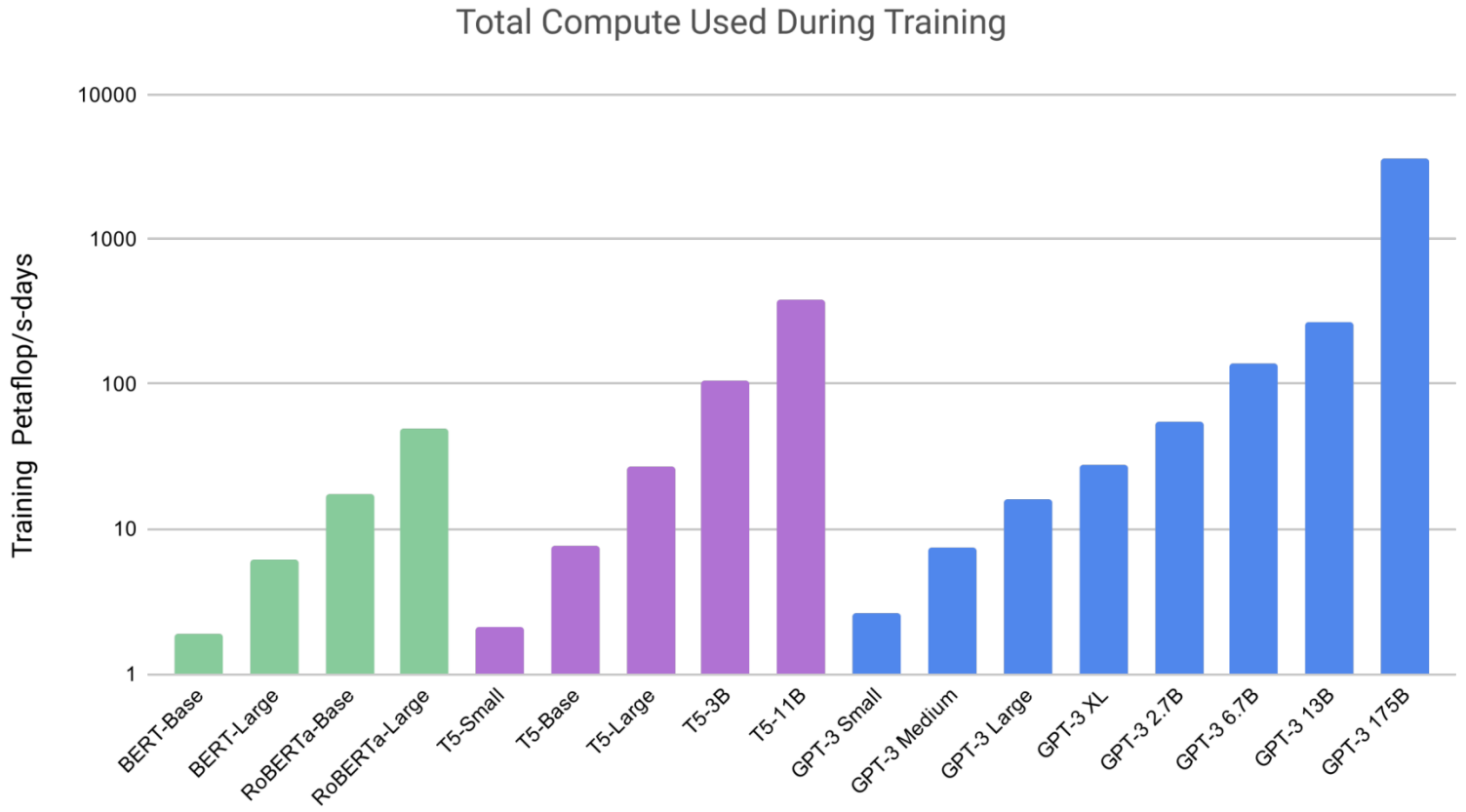
- GPT-2 but even larger: 1.3B -> 175B parameter models

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}

- Trained on 570GB of Common Crawl
- 175B parameter model’s parameters alone take >400GB to store (4 bytes per param). Trained in parallel on a “high bandwidth cluster provided by Microsoft”

Pre-training Cost

- Trained on Microsoft Azure, estimated to cost ~$5-10M (1000x BERT-large)



- petaflop/s-day is equivalent to 8 V100 GPUs at full efficiency of a day

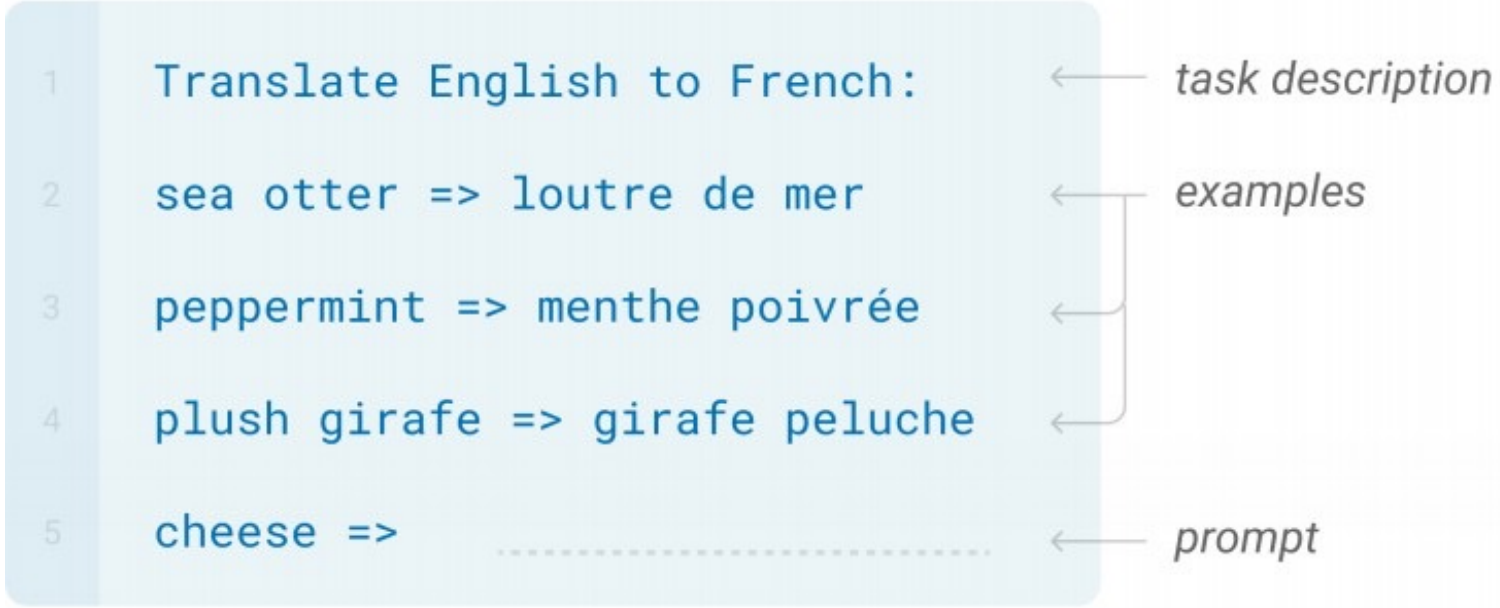
GPT3

- The standard way of training models like GPT

GPT3: Few-Shot Learning

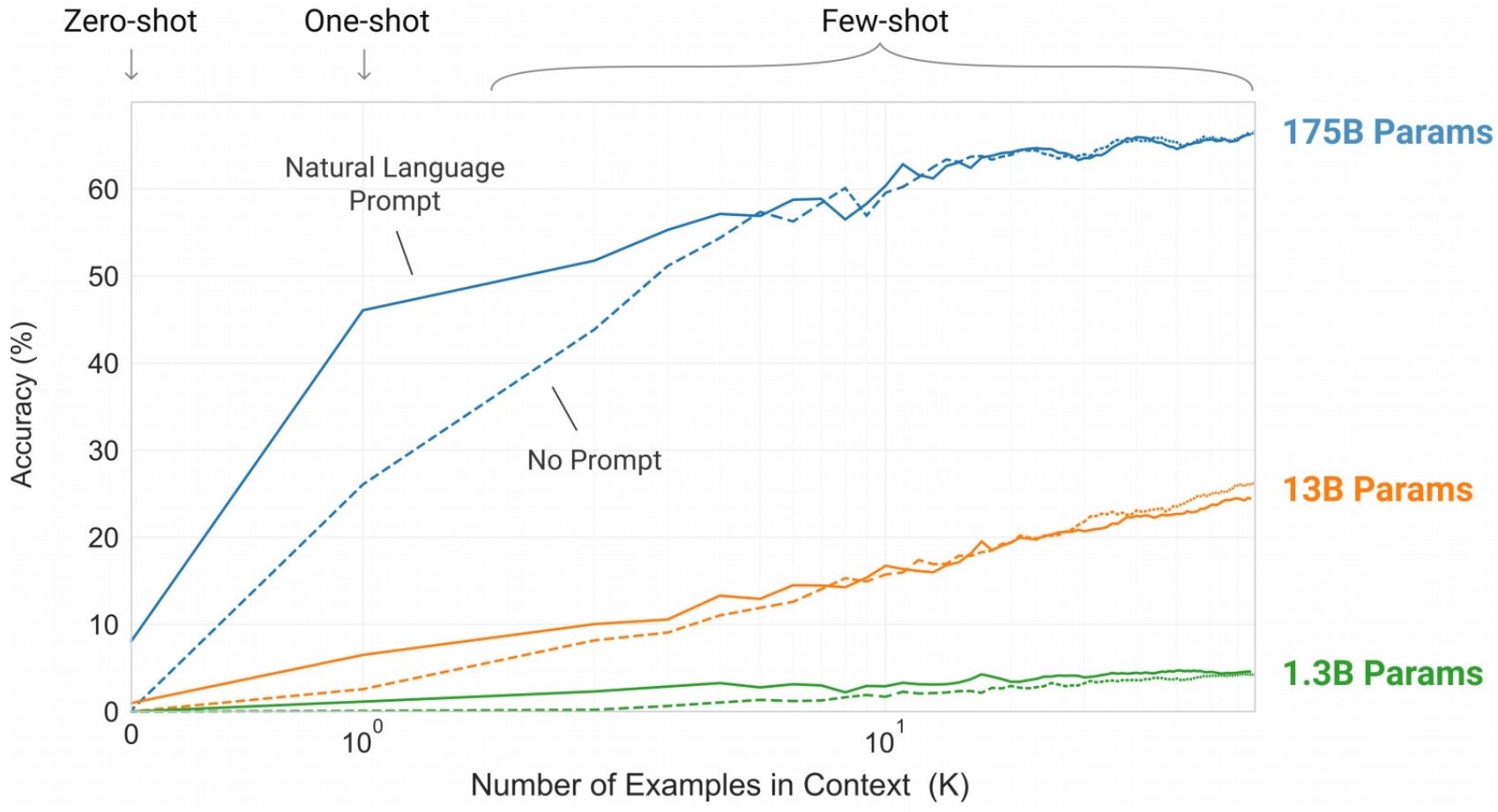
Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



GPT3: Few-Shot Learning

- **Key observation:** few-shot learning only works with the very largest models!



PALM: Google's LLM

Google AI Blog

Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance

Monday, April 4, 2022

Posted by Sharan Narang and Aakanksha Chowdhery, Software Engineers, Google Research

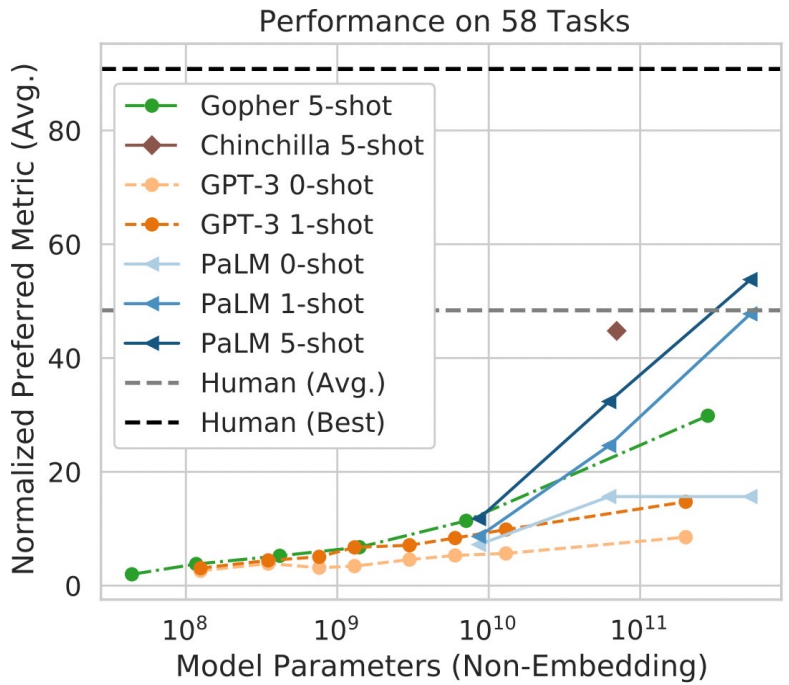
In recent years, large neural networks trained for language understanding and generation have achieved impressive results across a wide range of tasks. GPT-3 first showed that large language models (LLMs) can be used for *few-shot learning* and can achieve impressive results without large-scale task-specific data collection or model parameter updating. More recent LLMs, such as GLaM, LaMDA, Gopher, and Megatron-Turing NLG, achieved state-of-the-art few-shot results on many tasks by scaling model size, using sparsely activated modules, and training on larger datasets from more diverse sources. Yet much work remains in understanding the capabilities that emerge with few-shot learning as we push the limits of model scale.

Last year Google Research announced our vision for Pathways, a single model that could generalize across domains and tasks while being highly efficient. An important milestone toward realizing this vision was to develop the new Pathways system to orchestrate distributed computation for accelerators. In "PaLM: Scaling Language Modeling with Pathways", we introduce the Pathways Language Model (PaLM), a 540-billion parameter, dense decoder-only Transformer model trained with the Pathways system, which enabled us to efficiently train a single model across

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

PALM: Google's LLM

- 540 billion parameter model created by Google (not publicly available)
- Trained on 780 billion tokens, 6144 TPU v4 chips using Pathways to work across multiple TPU Pods).
- PALM2 is out – powers BARD



Total dataset size = 780 billion tokens	
Data source	Proportion of data
Social media conversations (multilingual)	50%
Filtered webpages (multilingual)	27%
Books (English)	13%
GitHub (code)	5%
Wikipedia (multilingual)	4%
News (English)	1%

Chowdhery et al. (2022)

Chat GPT: with RLHF

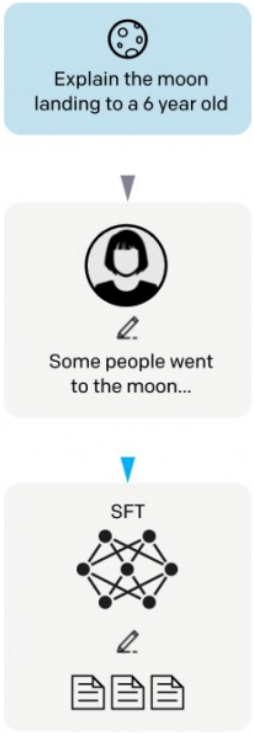
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



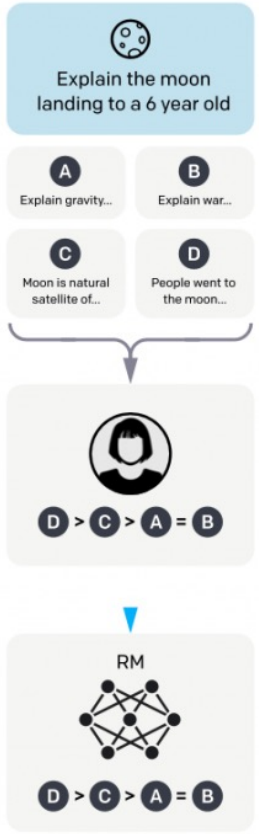
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

iq.opengenus.org

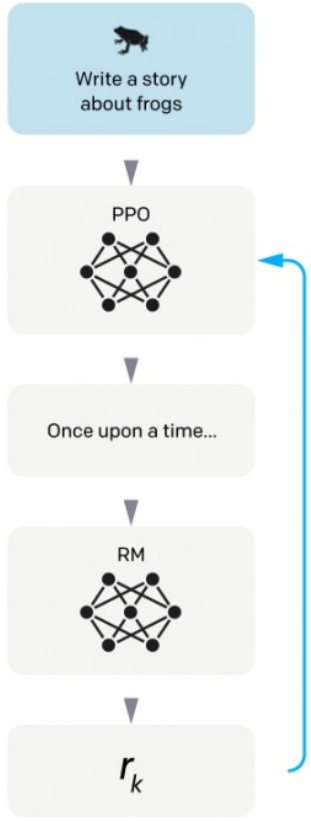
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Cost

- GPT-3: estimated to be $4.6M. This cost has a large carbon footprint
  - Carbon footprint: equivalent to driving 700,000 km by car (source: Anthropocene magazine)
  - (Counterpoints: GPT-3 isn't trained frequently, equivalent to 100 people traveling 7000 km for a conference, can use renewables)

BERT-Base pre-training: carbon emissions roughly on the same order as a single passenger on a flight from NY to San Francisco