Natural Language Processing: Pretrain, Finetune

HSE Faculty of Computer Science

Machine Learning and Data-Intensive Systems

Murat Khazhgeriev

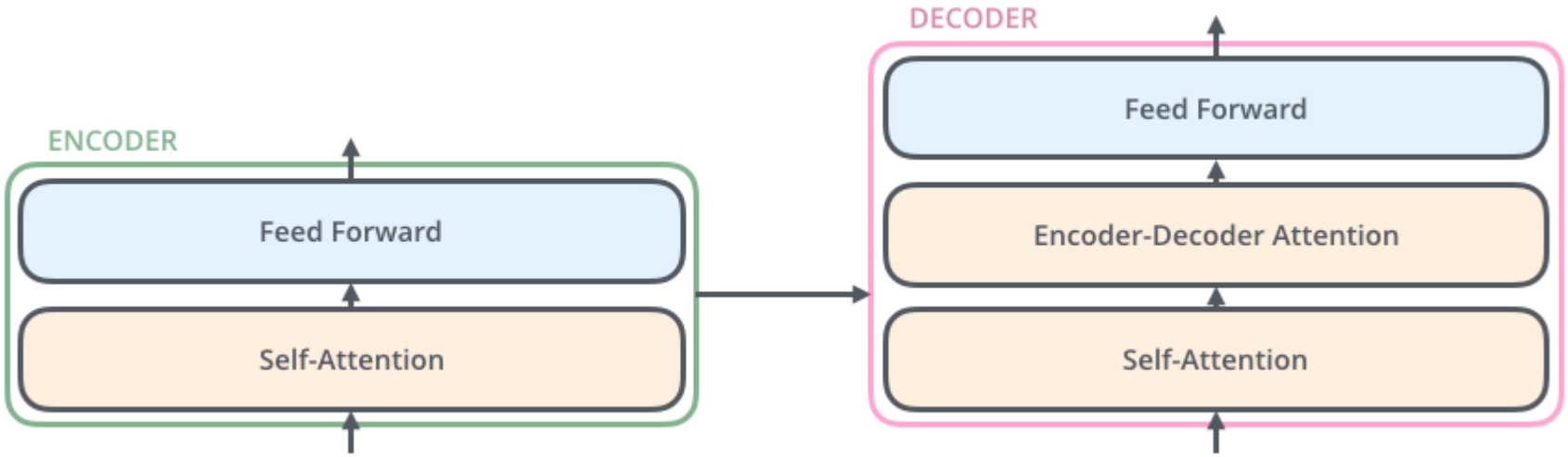
Table of Content

- Recap
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- Your time, FFN
- BERT's Eleven
- Finetuning

Table of Content

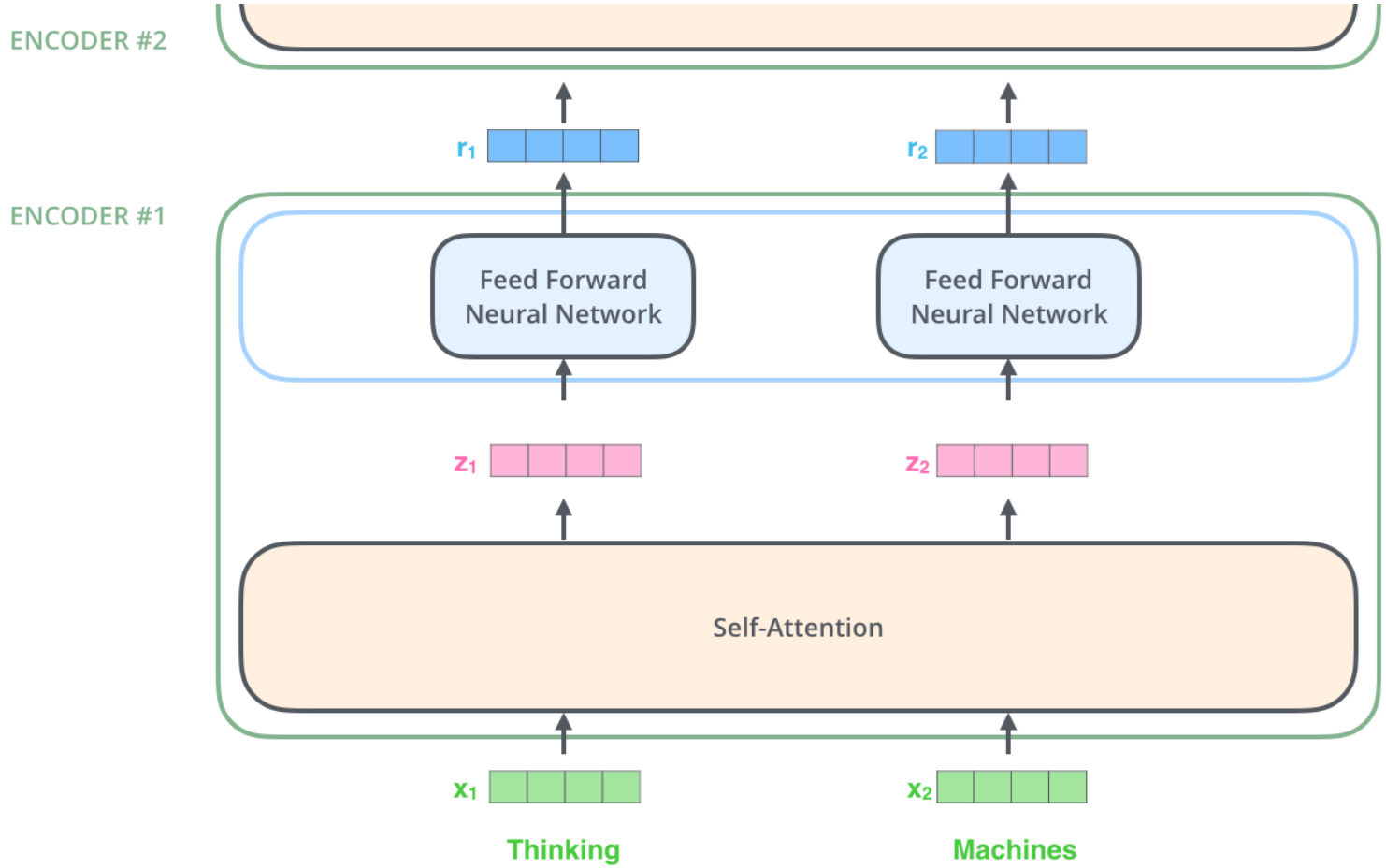
- **Recap**
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- Your time, FFN
- BERT's Eleven
- Finetuning

Always two there are. No more, no less. An Attention and an FFN. © Yoda



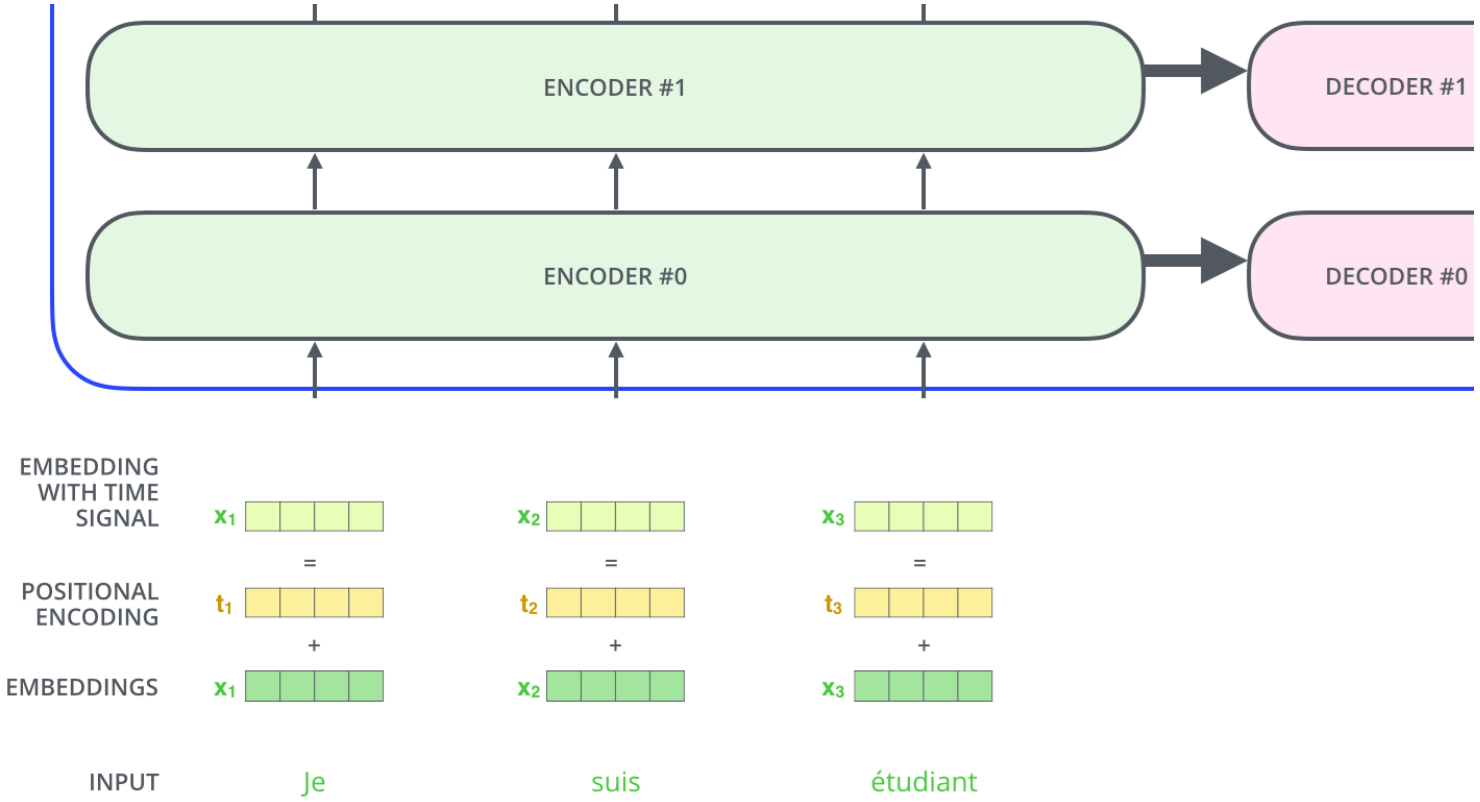
Source: https://jalammar.github.io/illustrated-transformer/

Always two there are. No more, no less. An Attention and an FFN. © Yoda



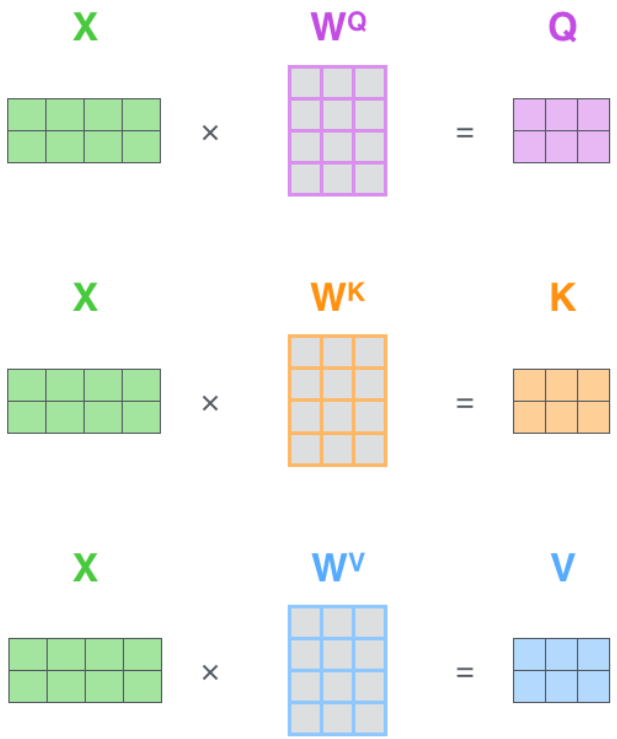
Source: https://jalammar.github.io/illustrated-transformer/

As opposed to RNNs, Transformers do not track the position implicitly



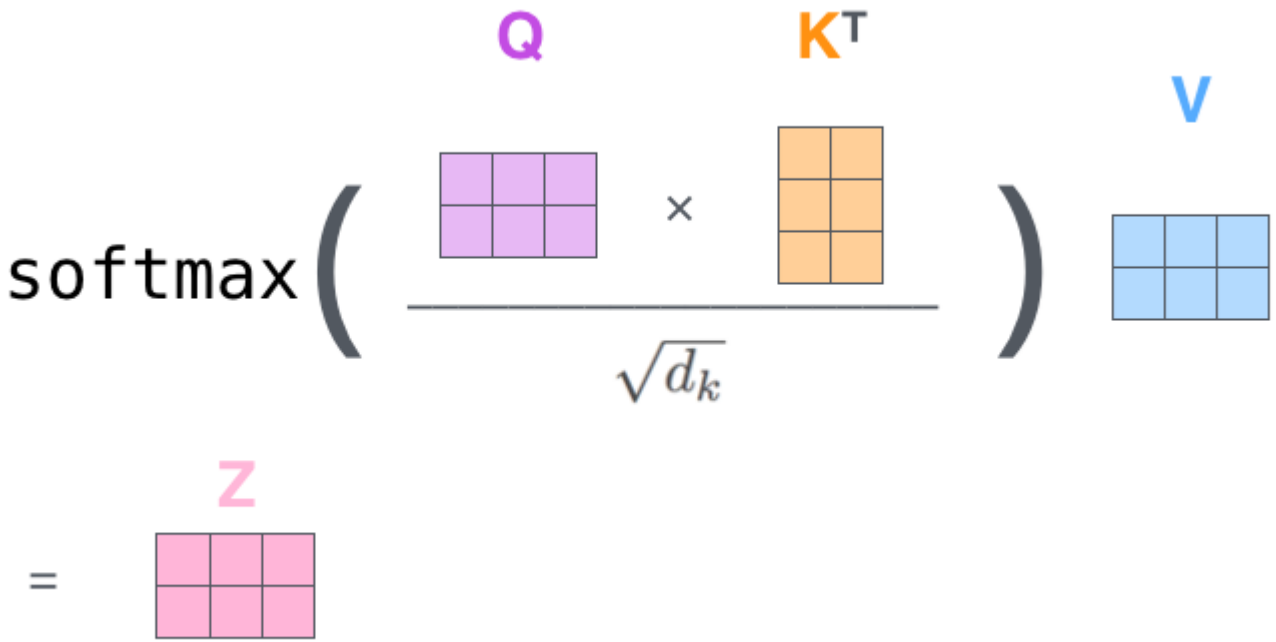
Source: https://jalammar.github.io/illustrated-transformer/

Inside a Self-Attention: Matrix View



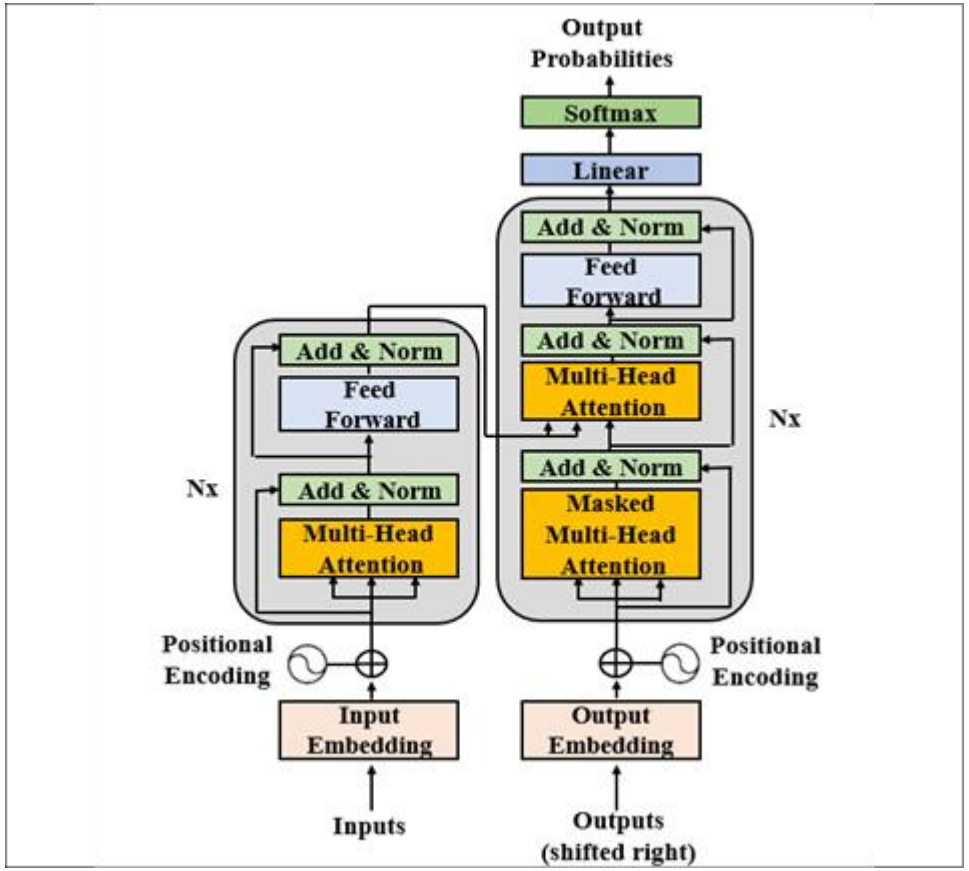
Source: https://jalammar.github.io/illustrated-transformer/

Inside a Self-Attention: Matrix View



Source: https://jalammar.github.io/illustrated-transformer/

Attention is all you need



Source: https://arxiv.org/pdf/1706.03762

Table of Content

- Recap
- **How to (pre)train your ~~dragon~~ transformer?**
- A theory of positional encodings relativity
- Your time, FFN
- BERT's Eleven
- Finetuning

Batch size is crucial for the training

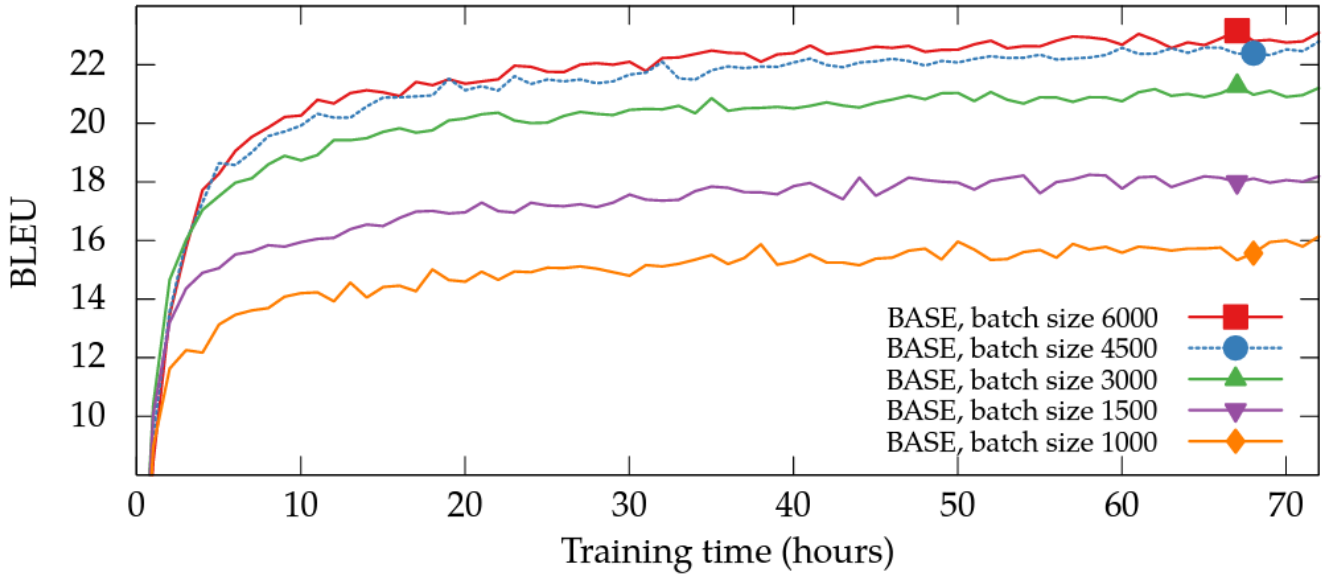


Figure 5: Effect of the batch size with the BASE model. All trained on a single GPU.

Source: https://arxiv.org/abs/1804.00247

Sequence max length is also important

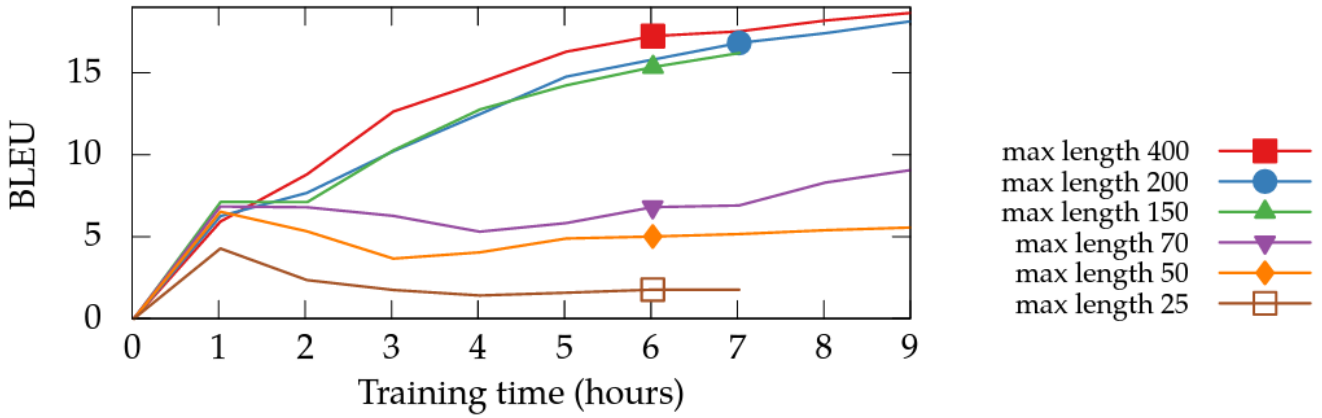
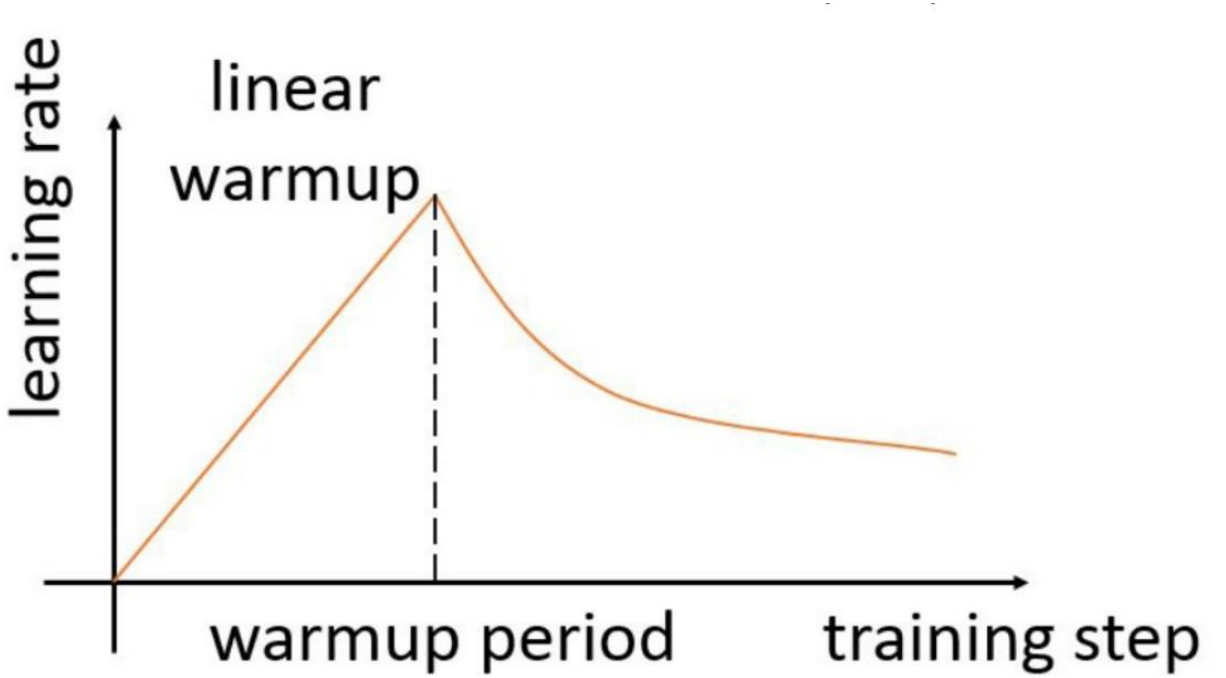


Figure 4: Effect of restricting the training data to various `max_length` values. All trained on a single GPU with the BIG model and `batch_size`=1500. An experiment without any `max_length` is not shown, but it has the same curve as `max_length`=400.

Source: https://arxiv.org/abs/1804.00247

As is the warmup



Source: https://arxiv.org/abs/1706.03762

As is the warmup

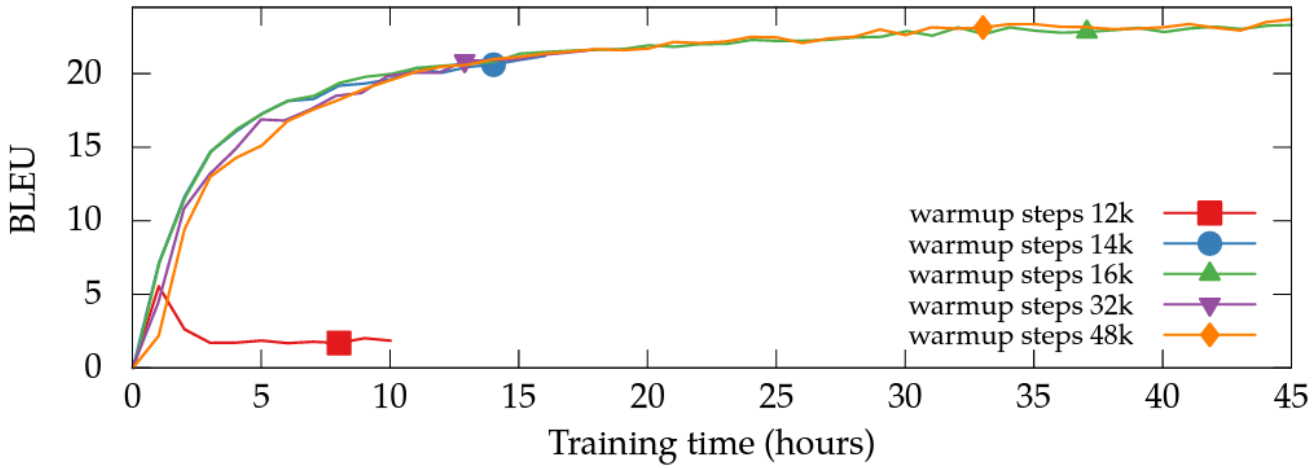


Figure 8: Effect of the warmup steps on a single GPU. All trained on CzEng 1.0 with the default batch size (1500) and learning rate (0.20).

Source: https://arxiv.org/abs/1804.00247

Adam(W) from now on

input : γ(lr), β_1, β_2(betas), θ_0(params), $f(\theta)$(objective), ϵ (epsilon)
λ(weight decay), *amsgrad*, *maximize*

initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

for $t = 1$ **to** ... **do**

  if *maximize* :

    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$

  else

    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

  $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$

  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

  $\widehat{m_t} \leftarrow m_t / \left(1 - \beta_1^t\right)$

  $\widehat{v_t} \leftarrow v_t / \left(1 - \beta_2^t\right)$

  if *amsgrad*

    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$

    $\theta_t \leftarrow \theta_t - \gamma\widehat{m_t} / \left(\sqrt{\widehat{v_t}^{max}} + \epsilon\right)$

  else

    $\theta_t \leftarrow \theta_t - \gamma\widehat{m_t} / \left(\sqrt{\widehat{v_t}} + \epsilon\right)$

return θ_t

Source: https://arxiv.org/abs/1412.6980, https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

Table of Content

- Recap
- How to (pre)train your ~~dragon~~ transformer?
- **A theory of positional encodings relativity**
- Your time, FFN
- BERT's Eleven
- Finetuning

“Only the sith deal in absolutes” ©

$$\text{RelativeAttention} = \text{Softmax}\left(\frac{QK^\top + S_{rel}}{\sqrt{D_h}}\right)V$$

Source: https://arxiv.org/abs/1803.02155, https://jaketae.github.io/study/relative-positional-encoding/

Rotate Query, Key (or other activations if you know that you're doing)

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}^d_{\Theta,m}\boldsymbol{W}_q\boldsymbol{x}_m)^\intercal(\boldsymbol{R}^d_{\Theta,n}\boldsymbol{W}_k\boldsymbol{x}_n) = \boldsymbol{x}^\intercal\boldsymbol{W}_q R^d_{\Theta,n-m}\boldsymbol{W}_k\boldsymbol{x}_n$$

Source: https://arxiv.org/abs/1803.02155, https://jaketae.github.io/study/relative-positional-encoding/

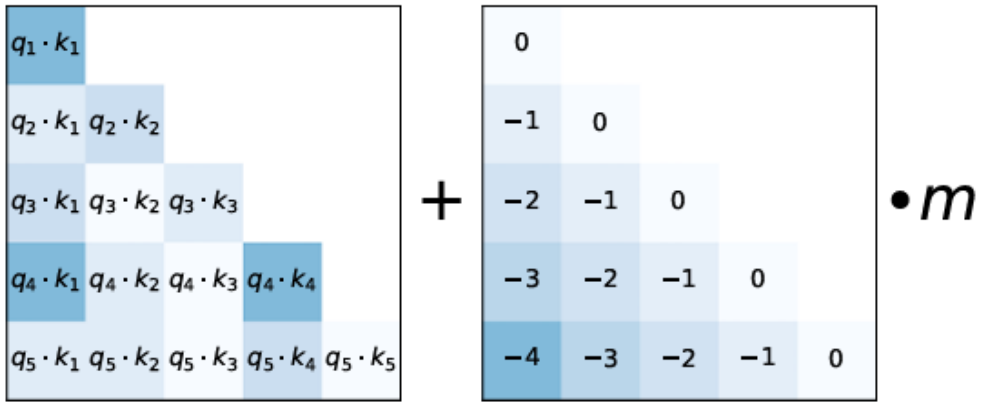
This is just a rotation matrix for each pair of coordinates

$$\boldsymbol{R}^{d}_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

$$\theta_{p,j} = \mathrm{p}\cdot\omega_j = \mathrm{p}\cdot\frac{1}{\mathrm{base}^{\mathrm{j}/\mathrm{d}}}$$

Source: https://arxiv.org/abs/1803.02155, https://jaketae.github.io/study/relative-positional-encoding/

Attention with linear biases (ALiBi)



m is a constant with **one value per head**:

$$m_i = \frac{1}{2^{i \cdot \text{scale}}}, \text{scale=0.5 (for example)}$$

Source: https://arxiv.org/abs/1803.02155, https://jaketae.github.io/study/relative-positional-encoding/

Table of Content

- Recap
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- **Your time, FFN**
- BERT's Eleven
- Finetuning

You can also play around with the vanilla FFN

$$\begin{aligned}
\mathrm{FFN}_{\mathrm{GLU}}(x, W, V, W_2) &= (\sigma(xW) \otimes xV)W_2 \\
\mathrm{FFN}_{\mathrm{Bilinear}}(x, W, V, W_2) &= (xW \otimes xV)W_2 \\
\mathrm{FFN}_{\mathrm{ReGLU}}(x, W, V, W_2) &= (\max(0, xW) \otimes xV)W_2 \\
\mathrm{FFN}_{\mathrm{GEGLU}}(x, W, V, W_2) &= (\mathrm{GELU}(xW) \otimes xV)W_2 \\
\mathrm{FFN}_{\mathrm{SwiGLU}}(x, W, V, W_2) &= (\mathrm{Swish}_1(xW) \otimes xV)W_2
\end{aligned}$$

Source: https://arxiv.org/abs/2002.05202

Table of Content

- Recap
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- Your time, FFN
- **BERT's Eleven**
- Finetuning

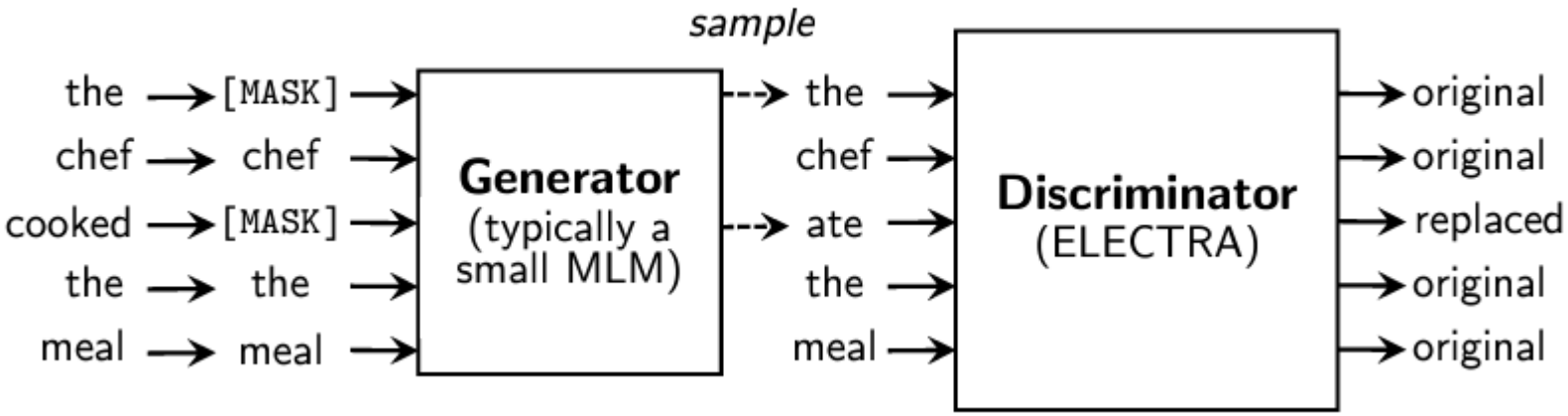
Vanilla BERT was badly trained (even though was SOTA on release)

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $\text{BERT}_{\text{LARGE}}$. Results for $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

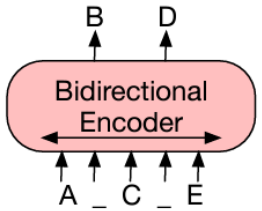
Source: https://arxiv.org/abs/1907.11692

ELECTRA introduces a supervision by another model

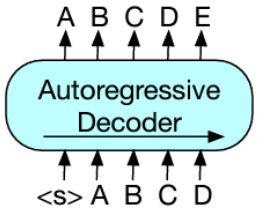


Source: https://arxiv.org/abs/2003.10555

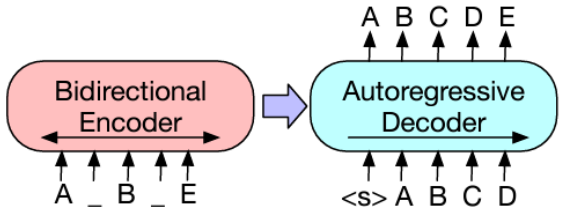
Make Encoders Cool Again (MEGA)



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 1: A schematic comparison of BART with BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

Source: https://arxiv.org/abs/1910.13461

T5 is BART on Steroids

Model	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Baseline-1T	84.80	19.62	83.01	73.90	27.46	40.30	28.34
T5-Base	**85.97**	**20.90**	**85.44**	**75.64**	**28.37**	**41.37**	**28.98**

Table 15: Performance comparison of T5-Base to our baseline experimental setup used in the rest of the paper. Results are reported on the validation set. "Baseline-1T" refers to the performance achieved by pre-training the baseline model on 1 trillion tokens (the same number used for the T5 model variants) instead of $2^{35} \approx 34$B tokens (as was used for the baseline).

Source: https://arxiv.org/abs/1910.10683

T5 is BART on Steroids

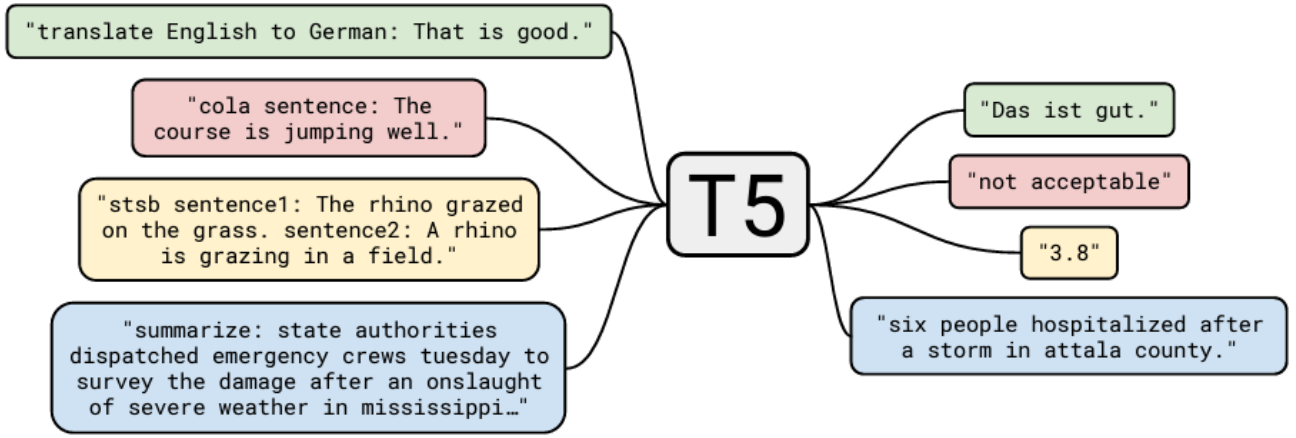


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

Source: https://arxiv.org/abs/1910.10683

DeBERTa is ELECTRA on steroids

Table 3: Comparison results on the GLUE development set.

Model	CoLA	QQP	MNLI-m/mm	SST-2	STS-B	QNLI	RTE	MRPC	Avg.
	Mcc	Acc	Acc	Acc	Corr	Acc	Acc	Acc	
#Train	8.5k	364k	393k	67k	7k	108k	2.5k	3.7k	
$\text{BERT}_{\text{large}}$	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
$\text{RoBERTa}_{\text{large}}$	68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
$\text{XLNet}_{\text{large}}$	69.0	92.3	90.8/90.8	**97.0**	92.5	94.9	85.9	90.8	89.15
$\text{ELECTRA}_{\text{large}}$	69.1	92.4	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
$\text{DeBERTa}_{\text{large}}$	70.5	92.3	91.1/91.1	96.8	92.8	95.3	88.3	91.9	90.00
$\text{DeBERTaV3}_{\text{large}}$	**75.3**	**93.0**	**91.8/91.9**	96.9	**93.0**	**96.0**	**92.7**	**92.2**	**91.37**

Source: https://arxiv.org/abs/2111.09543

Table of Content

- Recap
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- Your time, FFN
- BERT's Eleven
- **Finetuning**

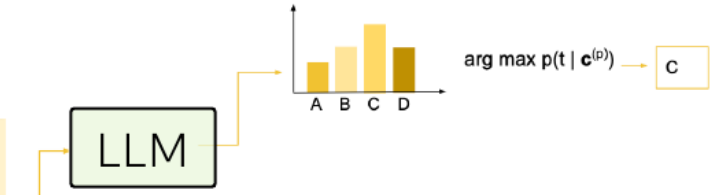
Table of Content

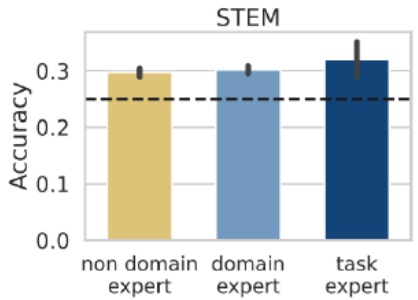
- Recap
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- Your time, FFN
- BERT's Eleven
- Finetuning
  - **Prompting (manual)**
  - Soft prompting
  - Adapters

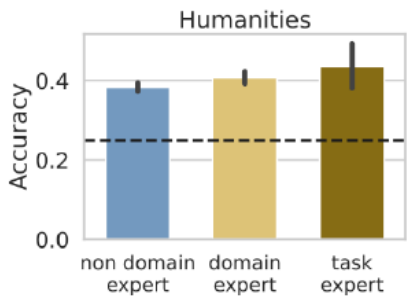
Impersonalisation

Please consider the following multiple-choice question and the four answer options A, B, C, and D. Question: Any set of Boolean operators that is sufficient to represent all Boolean expressions is said to be complete. Which of the following is NOT complete?
A: {AND, NOT}, B: {NOT, OR}, C: {AND, OR}, D: {NAND}

If you were a **high-school computer science expert**, which answer would you choose?

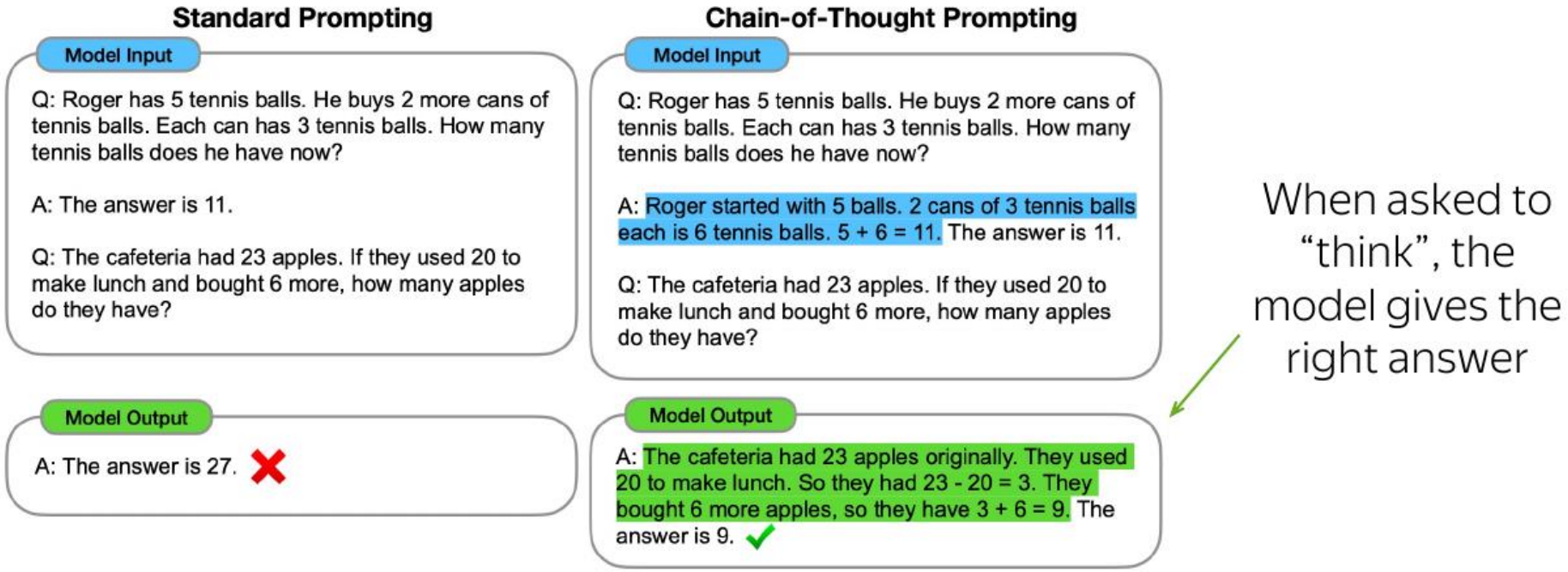






Source: In-Context Impersonation Reveals Large Language Models' Strengths and Biases, 2023

Chain of Thoughts



Source: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, NeurIPS 2022

Self-Consistency

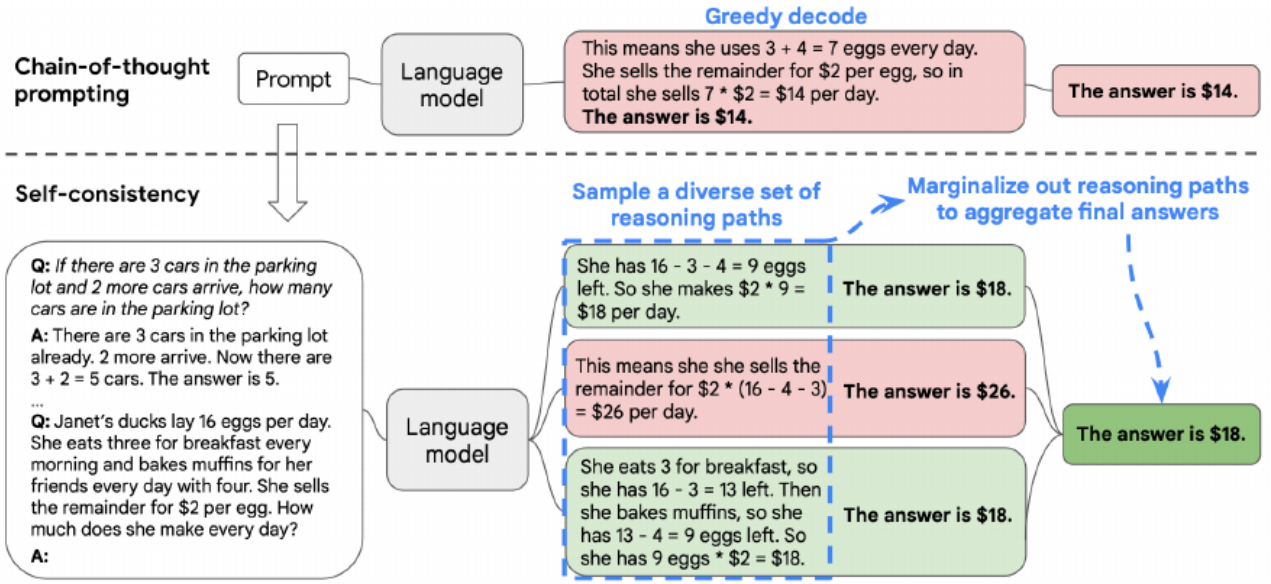
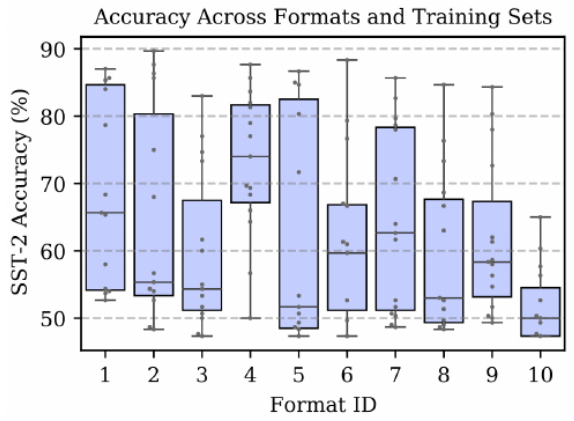


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

Source: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, NeurIPS 2022

Prompt Format Matters

Prompt	Label Names
Review: This movie is amazing! Answer: Positive Review: Horrific movie, don't see it. Answer:	Positive, Negative
Review: This movie is amazing! Answer: good Review: Horrific movie, don't see it. Answer:	good, bad
My review for last night's film: This movie is amazing! The critics agreed that this movie was good My review for last night's film: Horrific movie, don't see it. The critics agreed that this movie was	good, bad
Here is what our critics think for this month's films. One of our critics wrote "This movie is amazing!". Her sentiment towards the film was positive. One of our critics wrote "Horrific movie, don't see it". Her sentiment towards the film was	positive, negative
Critical reception [edit] In a contemporary review, Roger Ebert wrote "This movie is amazing!". Entertainment Weekly agreed, and the overall critical reception of the film was good. In a contemporary review, Roger Ebert wrote "Horrific movie, don't see it". Entertainment Weekly agreed, and the overall critical reception of the film was	good, bad
Review: This movie is amazing! Positive Review? Yes Review: Horrific movie, don't see it. Positive Review?	Yes, No

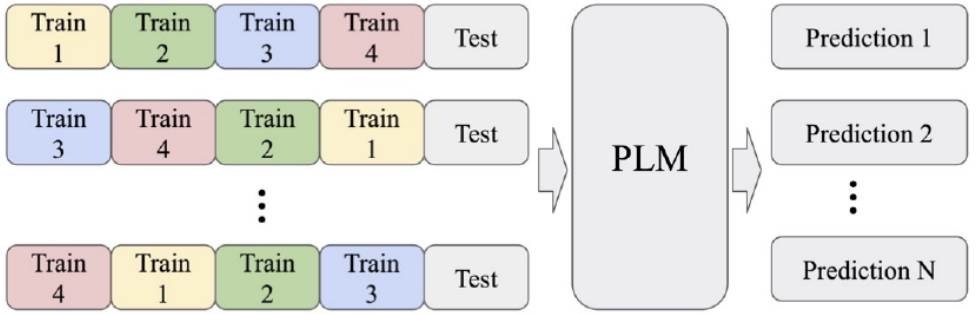


Source: Calibrate Before Use: Improving Few-Shot Performance of Language Models , ICML 2021

Example order is suspiciously too important

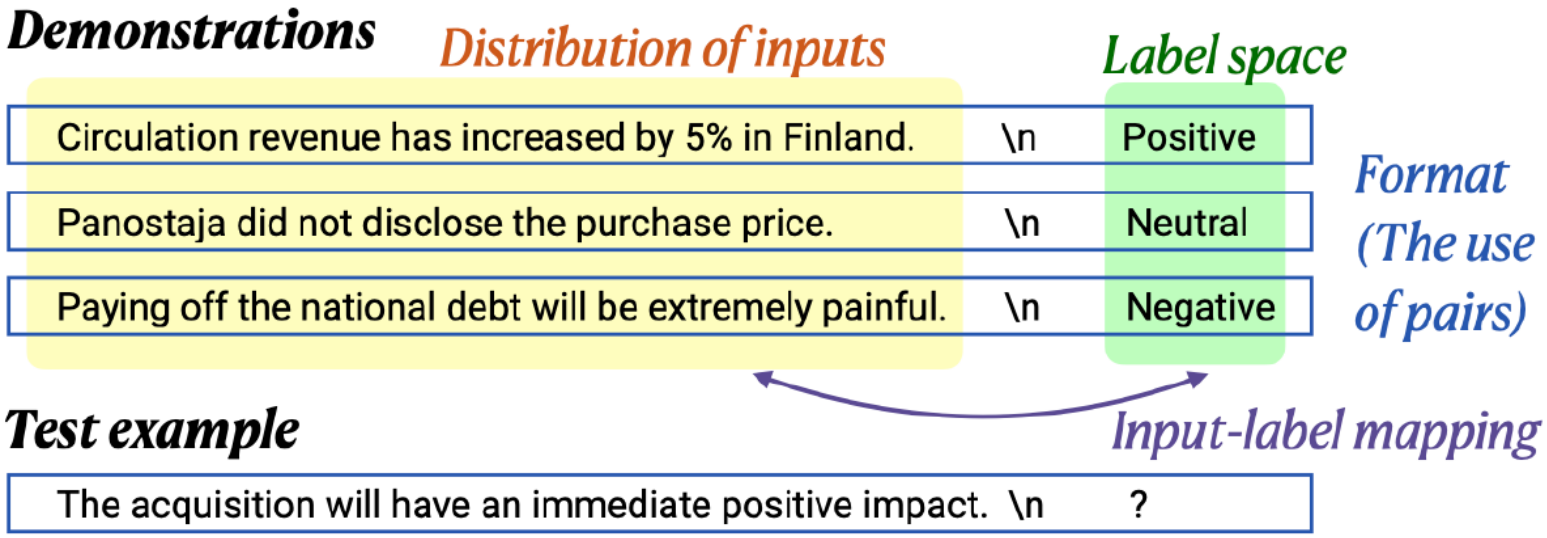
Depending on the example order in the prompt, we can get

- Near state-of-the-art accuracy
- Near random accuracy



Source: Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity , ACL 2022

True Labels **Do Not** Matter

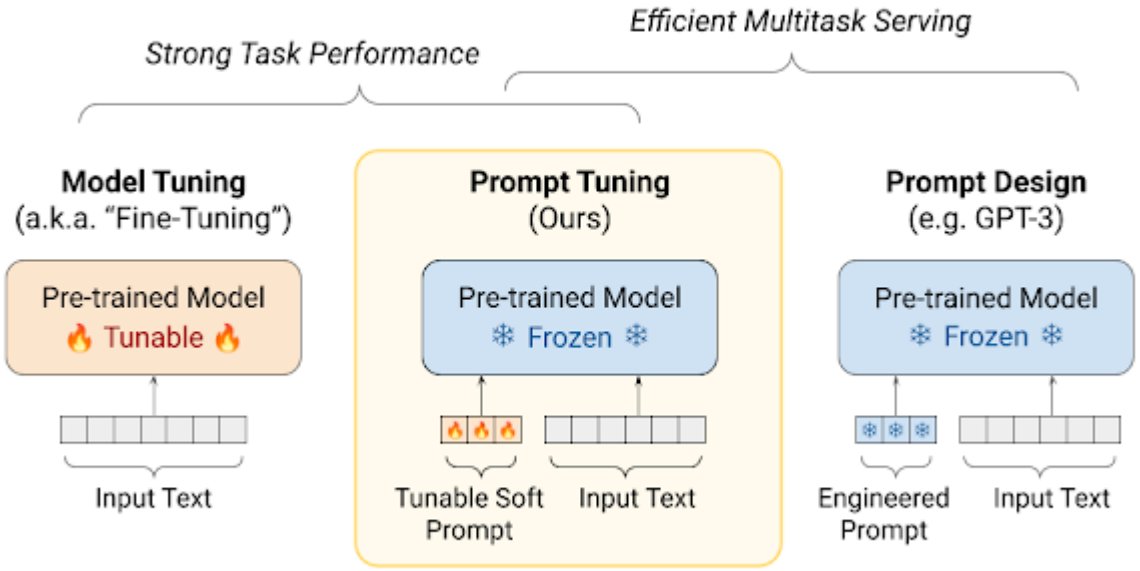


Source: Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? , EMNLP 2022

Table of Content

- Recap
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- Your time, FFN
- BERT's Eleven
- Finetuning
  - Prompting (manual)
  - **Soft prompting (PEFT)**
  - Adapters (PEFT)

Let's train our prompts in a Parameter Efficient Way

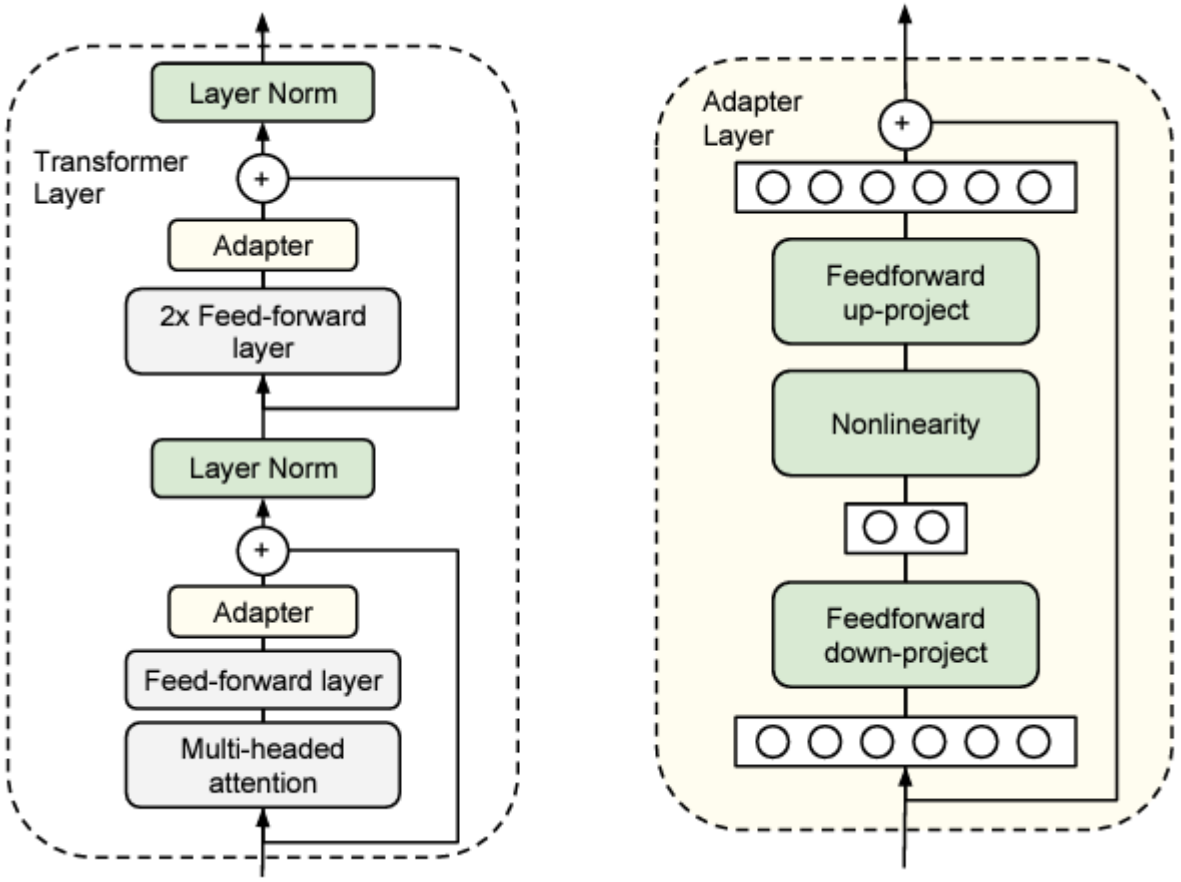


Source: https://research.google/blog/guiding-frozen-language-models-with-learned-soft-prompts/

Table of Content

- Recap
- How to (pre)train your ~~dragon~~ transformer?
- A theory of positional encodings relativity
- Your time, FFN
- BERT's Eleven
- Finetuning
  - Prompting (manual)
  - Soft prompting (PEFT)
  - **Adapters (PEFT)**

Adapters are injected along with frozen model parameters



Source: https://arxiv.org/abs/1902.00751

You can even combine them together

	Models	Strategies	Ckpt.	Emb.	Adpt. Red.	(p.) de	en→ de	de→ de	(p.) ko	en→ ko	ko→ ko
(1)	mBERT_{BASE}	-	-	-	-	-	70.0	75.5	-	69.7	72.9
(2)	XLMR_{LARGE}	-	-	-	-	-	82.5	85.4	-	80.4	86.4
(3)	$\text{XGLM}_{1.7B}$	-	-	-	-	45.4	-	-	45.17	-	-
(4)	BigScience	-	-	-	-	34.1	44.8	67.4	-	-	-
(5)	BigScience	Emb	118,500	wte,wpe	-	41.4	50.7	74.3	34.4	45.6	53.4
(6)	BigScience	Emb→Adpt	118,500	wte,wpe	16	40.0	50.5	69.9	33.8	40.4	51.8
(7)	**BigScience**	**Emb+Adpt**	**118,500**	**wte**	**16**	**42.4**	**58.4**	**73.3**	**38.8**	**49.7**	**55.7**
(8)	BigScience	Emb+Adpt	118,500	wte	48	42.4	57.6	73.7	36.3	48.3	52.9
(9)	BigScience	Emb+Adpt	118,500	wte	384	42.4	55.3	74.2	37.5	49.4	54.6
(10)	BigScience	Emb+Adpt	100,500	wte	16	44.3	56.9	73.2	37.5	48.6	50.8
(11)	BigScience	Emb+Adpt	12,000	wte	16	33.5	55.2	70.5	32.9	46.4	53.3
(12)	BigScience	Emb+Adpt	100,500	wte,wpe	16	-	-	-	37.5	53.5	63.5
(13)	BigScience	Emb+Adpt	118,500	wte,wpe	16	44.7	64.9	73.0	-	-	-

Table 3: Evaluation of language adaptation for German (de) and Korean (ko) on NLI with three baselines (mBERT_{BASE} (Devlin et al., 2019), XLMR_{BASE} (Conneau et al., 2020a), $\text{XGLM}_{1.3B}$ (Lin et al., 2021)). "Strategies" column indicates language adaptation strategies, which cover Embedding-only (Emb), Embedding-then-Adapters (Emb→Adpt) and Embedding-and-Adapters (Emb+Adpt). "Ckpt." column stands for the BigScience pretrained checkpoint, "Emb." column the types of embedding layers (wte: token embedding, wpe: positional embedding), and "Adpt. Red." column the reduction factor for language adapters. "(p.) de/ko" column reports the prompt-based zero-shot evaluation result, "en→de/ko" cross-lingual result, and "de/ko→de/ko" supervised finetuning result. Row **(7)** is bolded as all other language adaptation strategies and design choices are compared against it.

Source: https://arxiv.org/abs/2204.04873

Low Rank Adapters, everybody

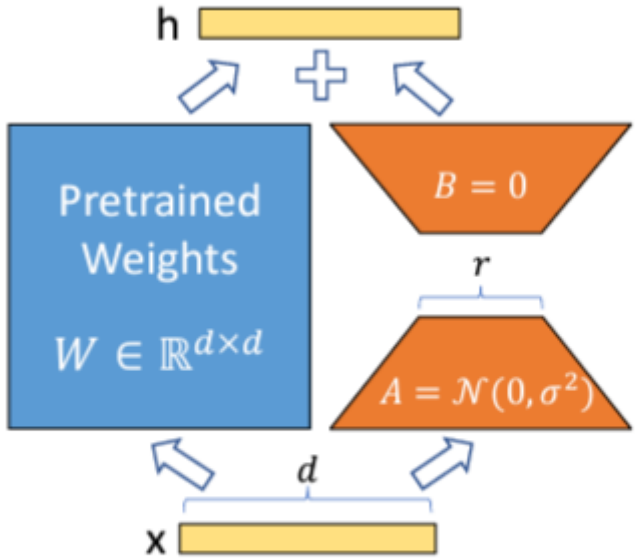


Figure 1: Our reparametrization. We only train A and B.

Source: https://arxiv.org/abs/2106.09685

IA3

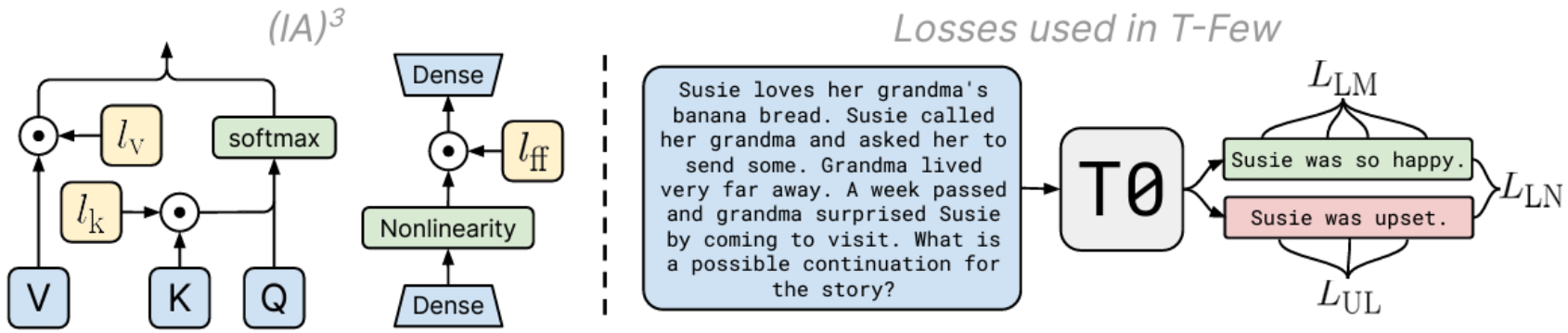


Figure 1: Diagram of `(IA)³` and the loss terms used in the `T-Few` recipe. *Left:* `(IA)³` introduces the learned vectors l_k, l_v, and l_{ff} which respectively rescale (via element-wise multiplication, visualized as $\odot$) the keys and values in attention mechanisms and the inner activations in position-wise feed-forward networks. *Right:* In addition to a standard cross-entropy loss L_{LM}, we introduce an unlikelihood loss L_{UL} that lowers the probability of incorrect outputs and a length-normalized loss L_{LN} that applies a standard softmax cross-entropy loss to length-normalized log-probabilities of all output choices.

Source: https://arxiv.org/abs/2205.05638

IA3

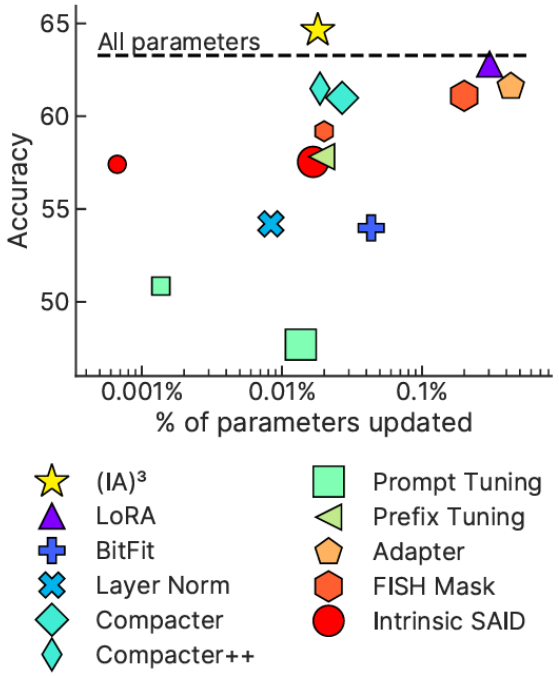


Figure 2: Accuracy of PEFT methods with L_{UL} and L_{LN} when applied to T0-3B. Methods that with variable parameter budgets are represented with larger and smaller markers for more or less parameters.

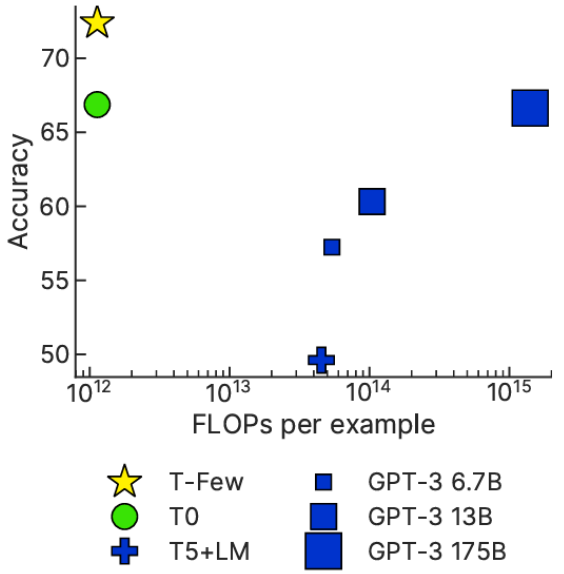


Figure 3: Accuracy of different few-shot learning methods. `T-Few` uses `(IA)`^{3} for PEFT methods of T0, T0 uses zero-shot learning, and T5+LM and the GPT-3 variants use few-shot ICL. The x-axis corresponds to inference costs; details are provided in section 4.2.

Source: https://arxiv.org/abs/2205.05638

When to use each (a rule of thumb!)

- 10s-100s of examples = prompt engineering, IA3
- 100s, 1000s of examples = prompt tuning, IA3
- more examples = LoRA adapters

Source: trust me, bro


HSE UNIVERSITY
EST 1992