JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Transformer Language Models

CSCI 601-471/671 (NLP: Self-Supervised Models)

https://self-supervised.cs.jhu.edu/sp2024/

Language Models: A History

- Probabilistic n-gram models of text generation [Jelinek+ 1980's, …]
  - Applications: Speech Recognition, Machine Translation

- Word representation learning [Brown 1992, …]
  - Brown, LSA, Word2Vec, Glove …

- Statistical or shallow neural LMs (late 90's – mid 00's) [Bengio+ 2001, …]

- Pre-training deep neural language models (2017's onward):
  - Many models based on: Self-Attention

RNNs, Back to the Cons

- While RNNs in theory can represent long sequences, they quickly forget portions of the input.
- Vanishing/exploding gradients
- Difficult to parallelize
- The alternative solution we will see: Transformers!

Chapter Plan

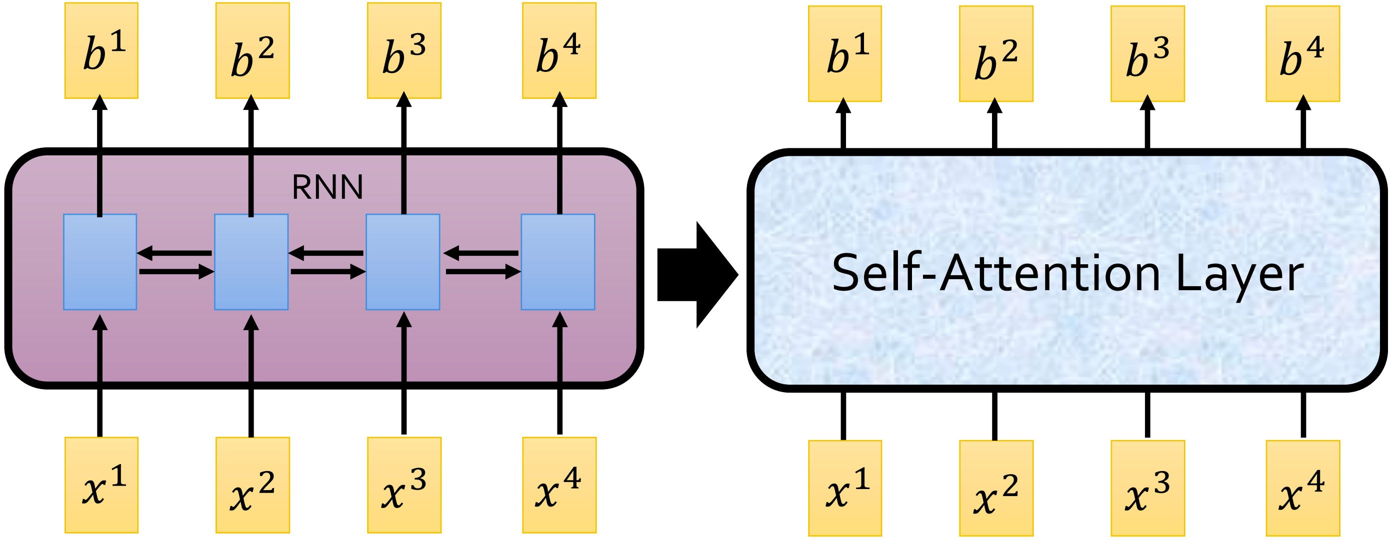
1. Self-Attention: how it works
2. Transformer architecture
3. Transformer-based families of Language Models
4. Practical hacks and variants
5. Various objective functions

Chapter goal----

Self-Attention

Self-Attention

- b^i is obtained based on the whole input sequence.
- can be parallelly computed.

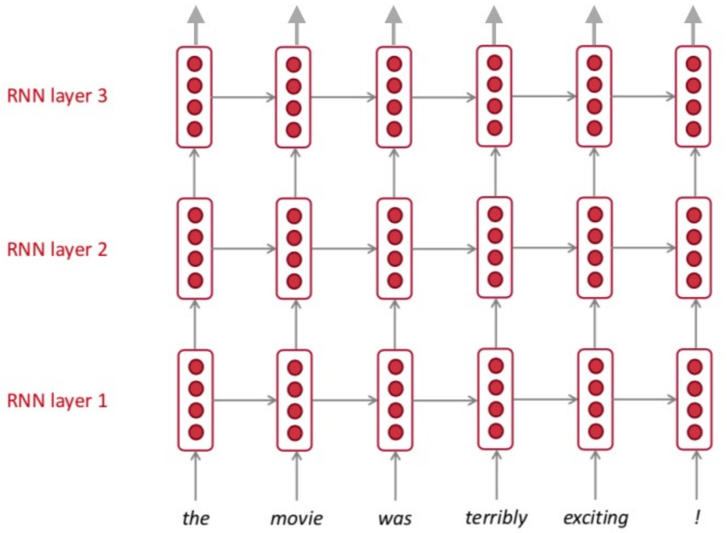


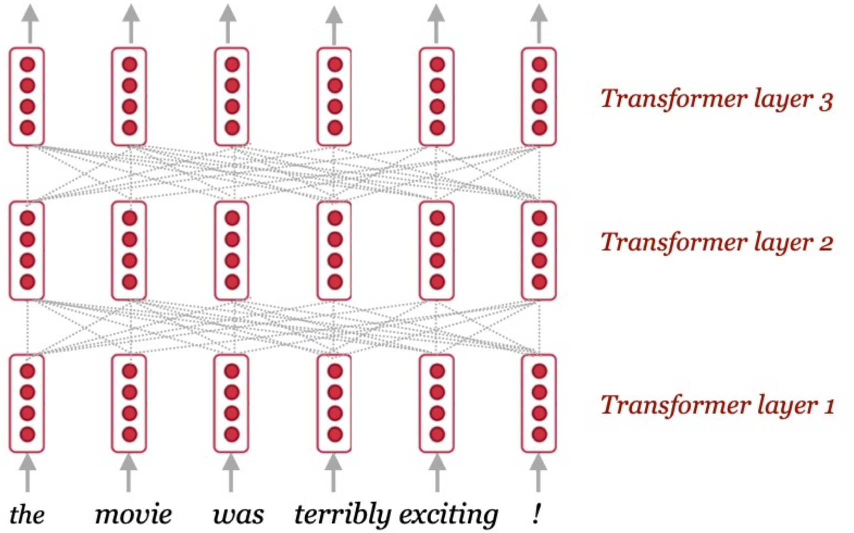
Idea: replace any thing done by RNN with self-attention.

"Neural machine translation by jointly learning to align and translate" Bahdanau etl. 2014;
"Attention is All You Need" Vaswani et al. 2017

[adopted from Hung-yi Lee]

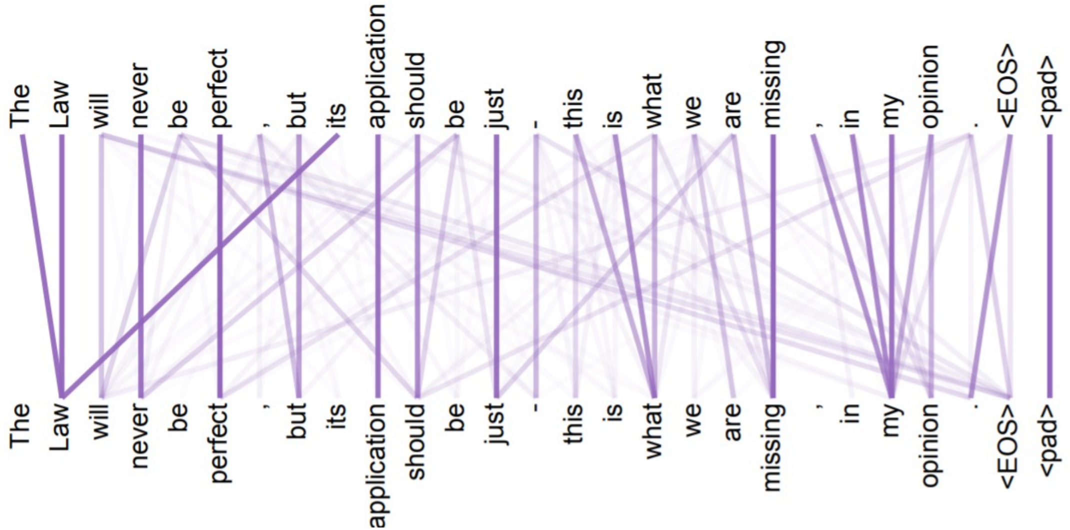
RNN vs Transformer





Attention

- Core idea: build a mechanism to focus (“attend”) on a particular part of the context.



[Attention Is All You Need, Vaswani et al. 2017]

Defining Self-Attention

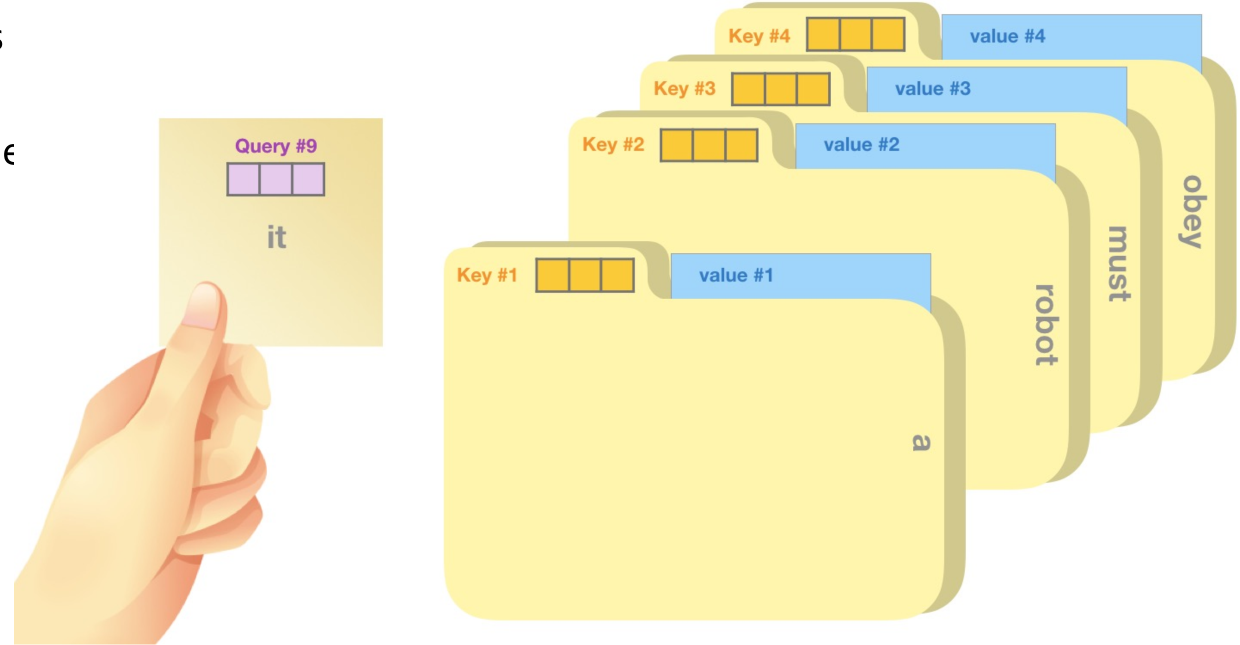
- Terminology:
  - Query: to match others
  - Key: to be matched
  - Value: information to be extracted

[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

Defining Self-Attention

An analogy ....

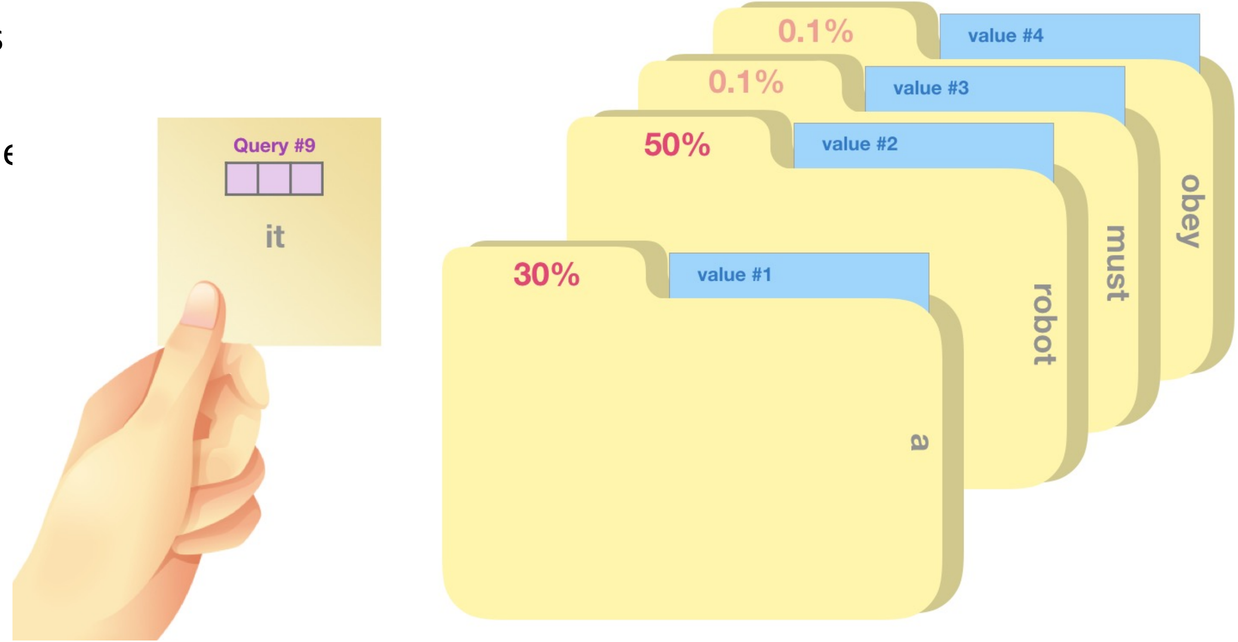
- Terminology:
  - Query: to match others
  - Key: to be matched
  - Value: information to be



[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

Defining Self-Attention

- Terminology:
  - Query: to match others
  - Key: to be matched
  - Value: information to be



[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

q: query (to match others)

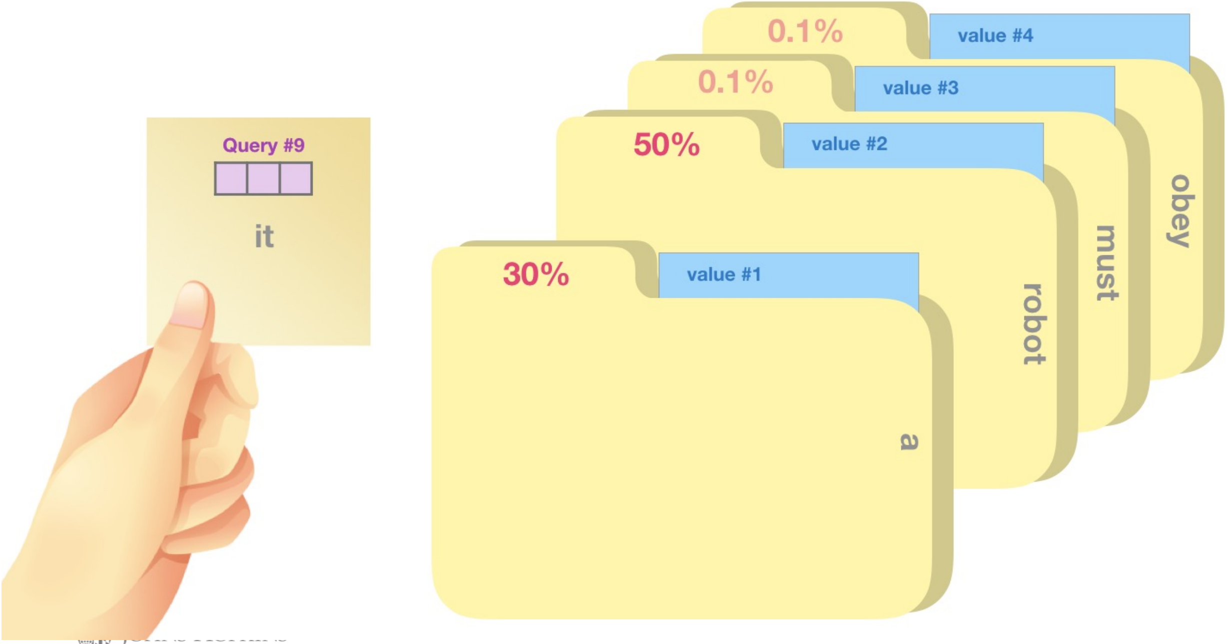
$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

$$v_i = W^v x_i$$



q: query (to match others)

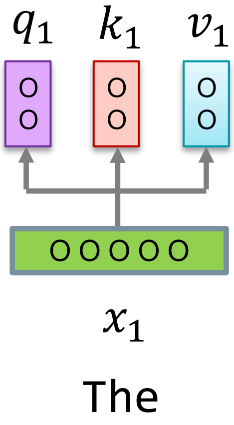
$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

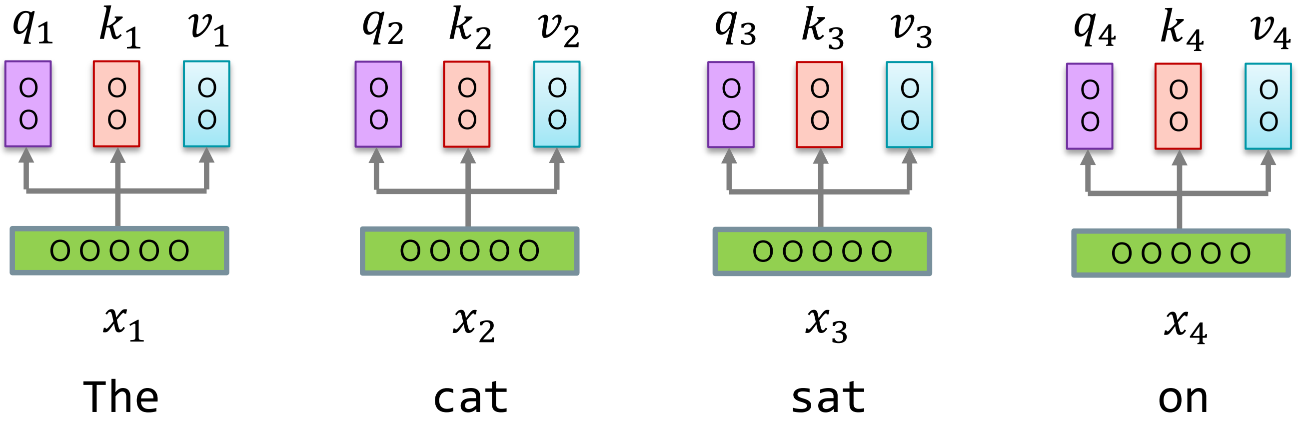
$$v_i = W^v x_i$$



q: query (to match others)
$$q_i = W^q x_i$$

k: key (to be matched)
$$k_i = W^k x_i$$

v: value (information to be extracted)
$$v_i = W^v x_i$$



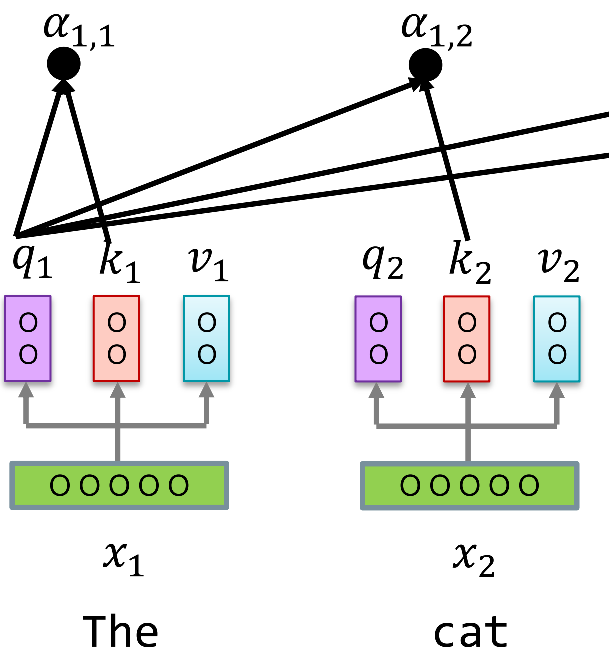
$$\alpha_{1,i} = \underbrace{q^1 \cdot k^i / \sqrt{d}}_{\text{Scaled dot product}}$$

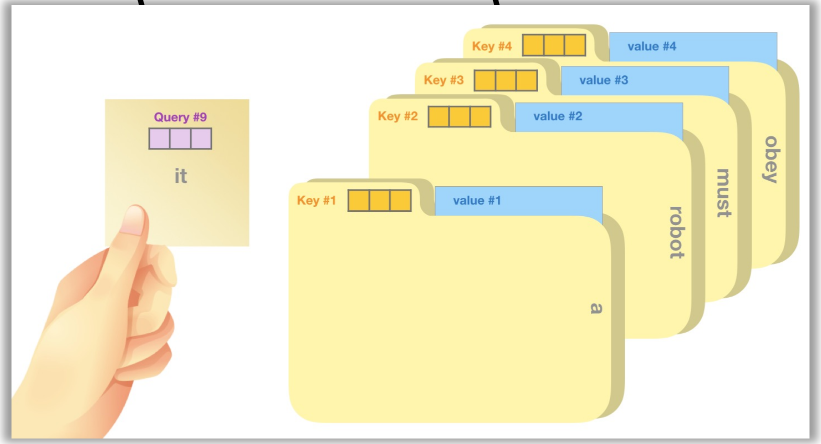
q: query (to match others)

k: key (to be matched)

v: value (information to be extracted)

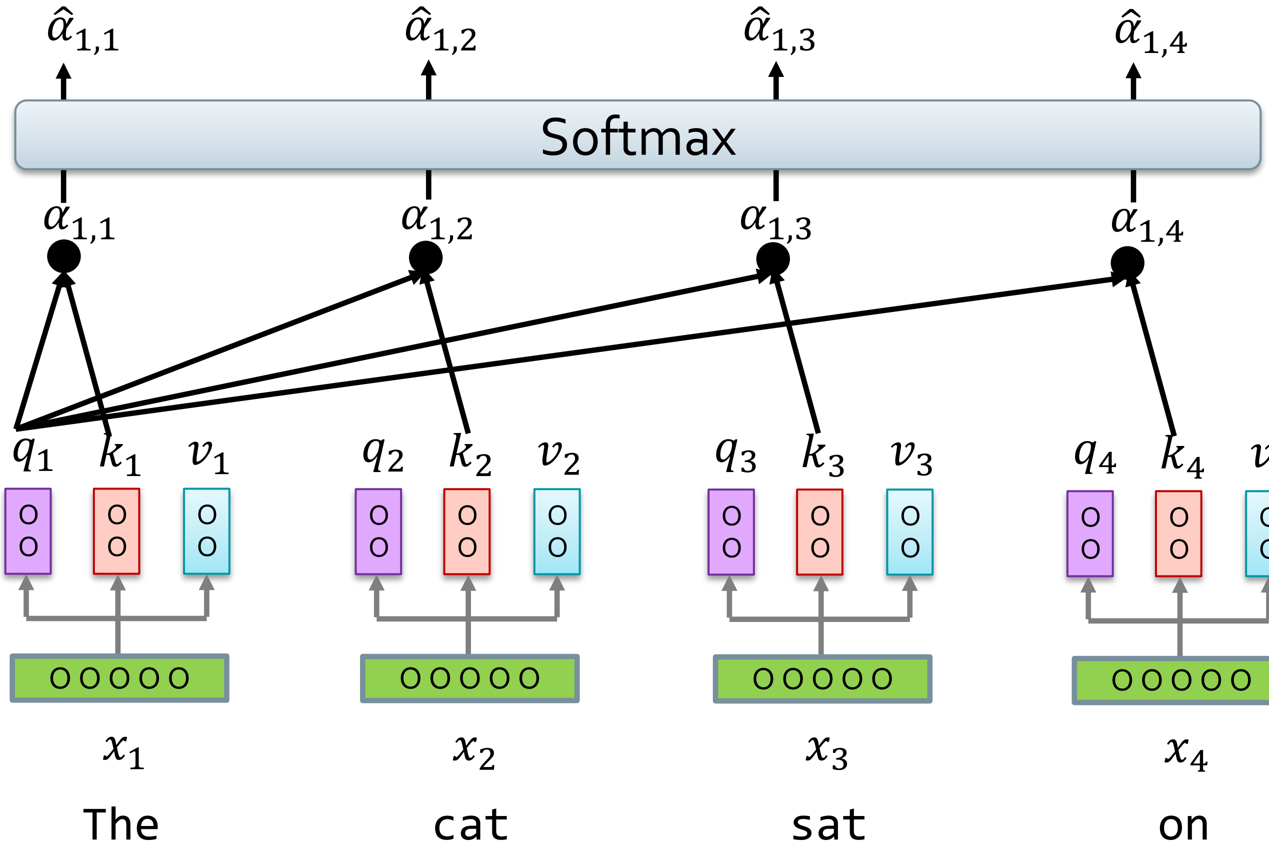
How much should "The" attend to other positions?

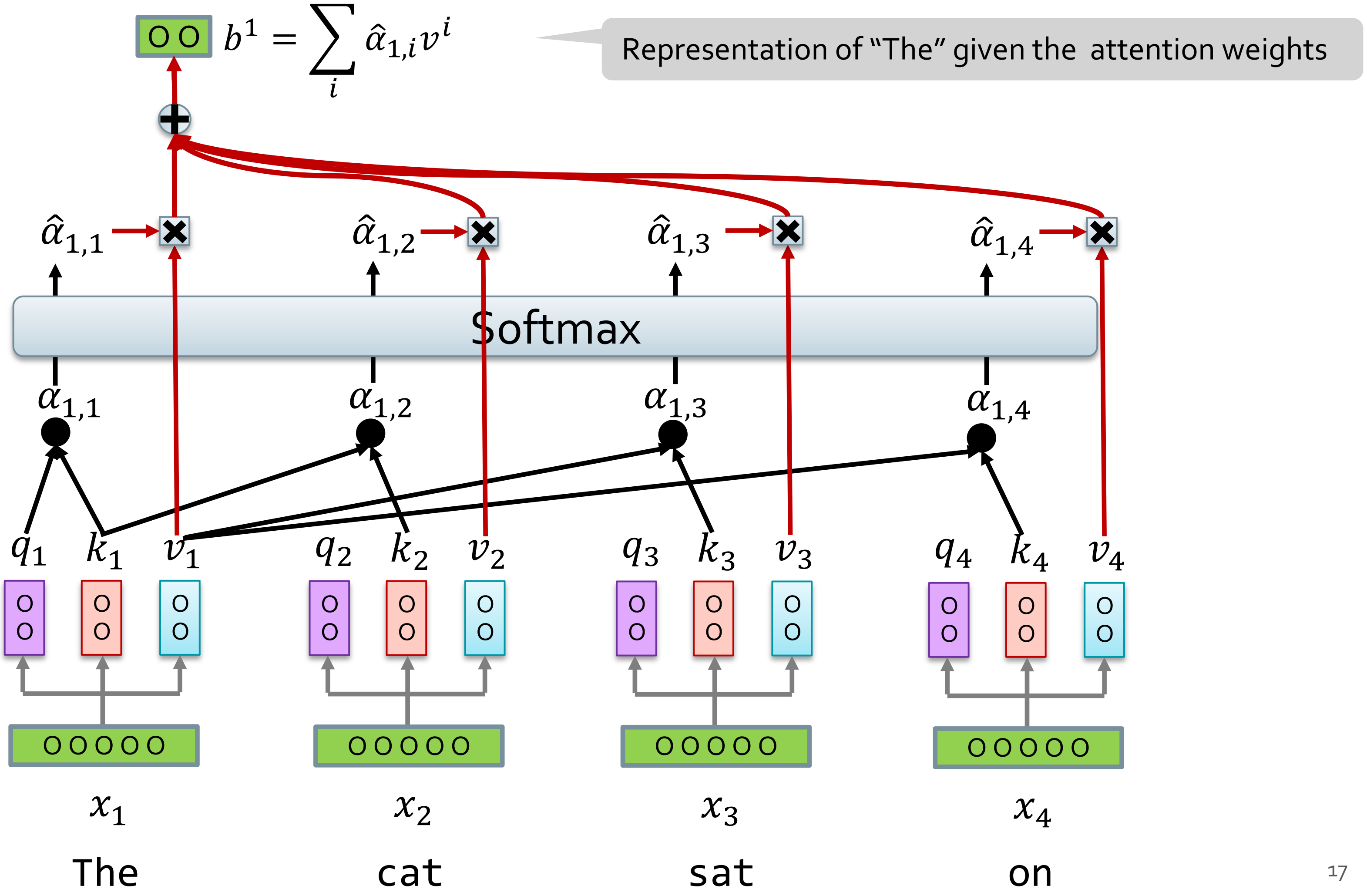




$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

How much should "The" attend to other positions?





$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
Representation of "The" given the attention weights
$\hat{\alpha}_{1,1}$
$\hat{\alpha}_{1,2}$
$\hat{\alpha}_{1,3}$
$\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
x_1
x_2
x_3
x_4
The
cat
sat
on

Self-Attention

- Can write it in matrix form:
- Given input $\mathbf{x}$:

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

hardmaru
@hardmaru

The most important formula in deep learning after 2018

Self-Attention

What is self-attention? Self-attention calculates a weighted average of feature representations with the weight proportional to a similarity score between pairs of representations. Formally, an input sequence of n tokens of dimensions d, $X \in \mathbf{R}^{n \times d}$, is projected using three matrices $W_Q \in \mathbf{R}^{d \times d_q}$, $W_K \in \mathbf{R}^{d \times d_k}$, and $W_V \in \mathbf{R}^{d \times d_v}$ to extract feature representations Q, K, and V, referred to as query, key, and value respectively with $d_k = d_q$. The outputs Q, K, V are computed as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{1}$$

So, self-attention can be written as,

$$S = D(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V, \tag{2}$$

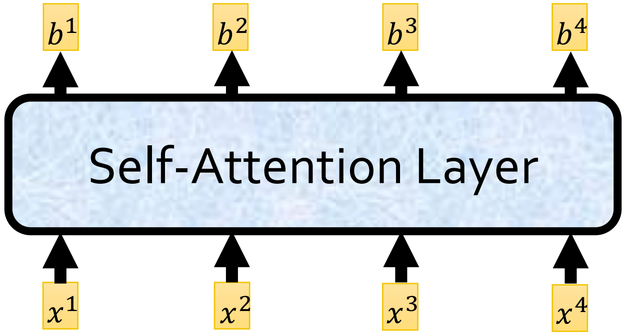
where softmax denotes a *row-wise* softmax normalization function. Thus, each element in S depends on all other elements in the same row.

9:08 PM · Feb 9, 2021 · Twitter Web App

553 Retweets **42** Quote Tweets **3,338** Likes

Self-Attention: Back to Big Picture

- Attention is a powerful mechanism to create context-aware representations
- A way to focus on select parts of the input



- Better at maintaining long-distance dependencies in the context.

[Attention Is All You Need, Vaswani et al. 2017]

Properties of Self-Attention

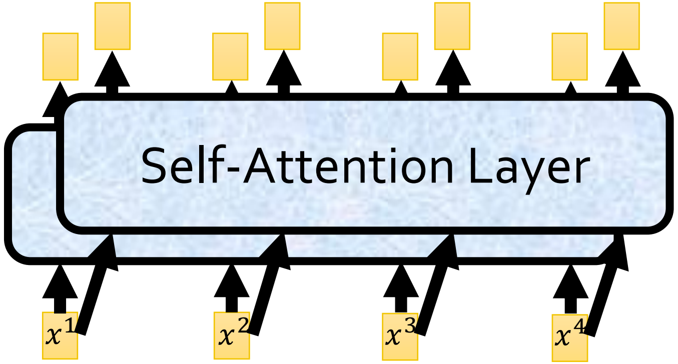
Layer Type	Complexity per Layer	Sequential Operations
Self-Attention	$O(n^2 \cdot d)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$

- n = sequence length, d = hidden dimension
- Quadratic complexity, but:
  - O(1) sequential operations (not linear like in RNN)

- Efficient implementations

Multi-Headed Self-Attention

- Multiple parallel attention layers is quite common.
  - Each attention layer has its own parameters.
  - Concatenate the results and run them through a linear projection.



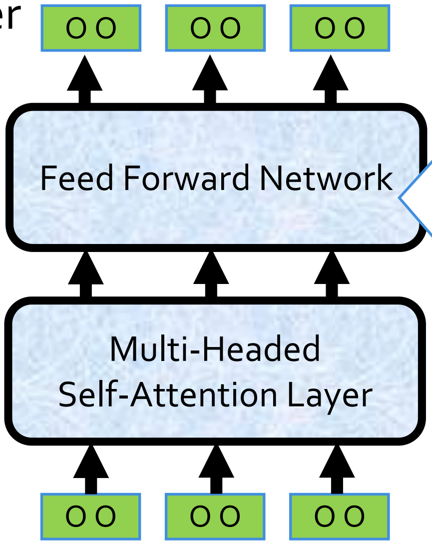
[Attention Is All You Need, Vaswani et al. 2017]

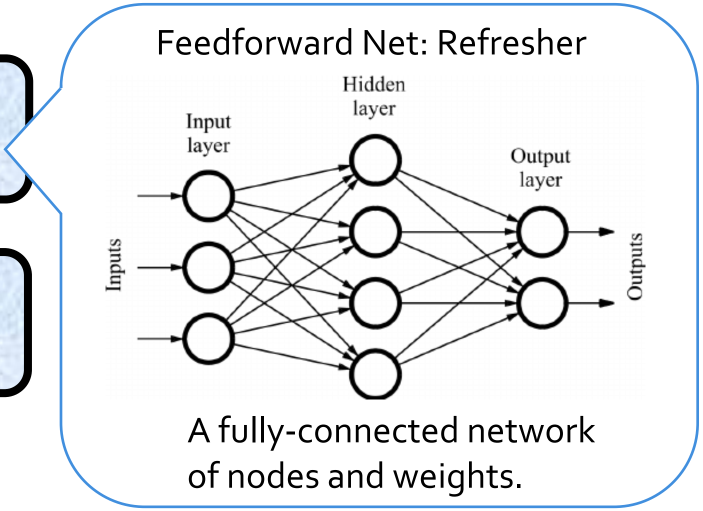
24

Combine with FFN

- Add a feed-forward network on top it to add more expressivity.
  - This allows the model to apply another transformation to the contextual representations (or "post-process" them).
  - Usually, the dimensionality of the hidden feedforward layer is 2-8 times larger than the input dimension.

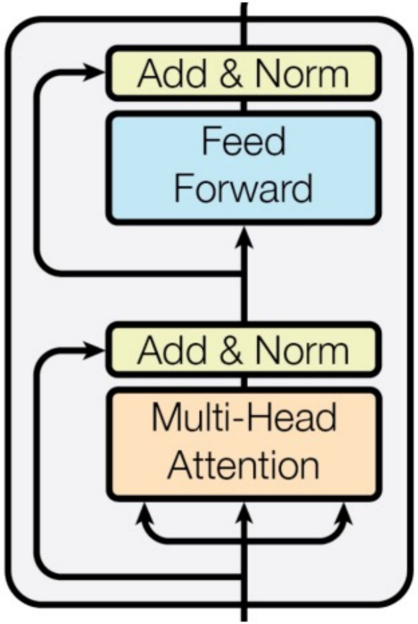
$$\mathrm{FFN}(\mathbf{x}) = f(cW_1 + b_1)W_2 + b_2$$





How Do We Prevent Vanishing Gradients?

- Residual connections let the model “skip” layers
  - These connections are particularly useful for training deep networks
- Use layer normalization to stabilize the network and allow for proper gradient flow



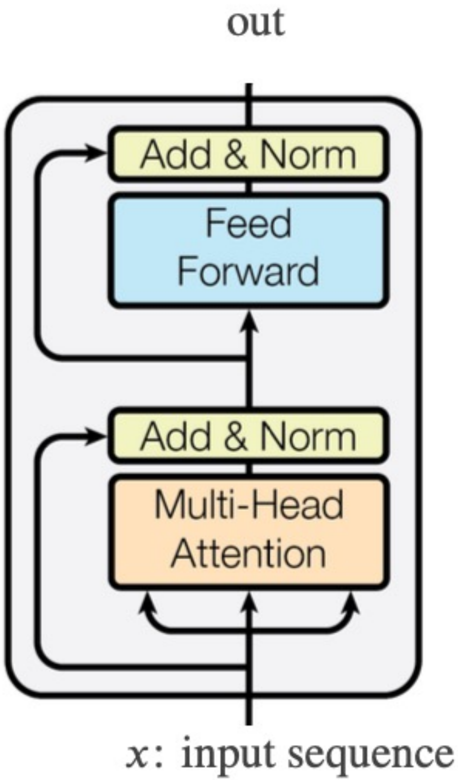
[Attention Is All You Need, Vaswani et al. 2017]

Putting it Together: Self-Attention Block

Given input $\mathbf{x}$:

$$\text{out} = LN(\tilde{\boldsymbol{c}} + \boldsymbol{c}')$$
$$\tilde{\boldsymbol{c}} = \text{FFN}(\boldsymbol{c}') = f(\boldsymbol{c}'W_1 + b_1)W_2 + b_2$$

$$\boldsymbol{c}' = LN(\boldsymbol{c} + \boldsymbol{x})$$
$$\boldsymbol{c} = \text{MultiHeadedAttention}(\boldsymbol{x}; \mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v)$$



[Attention Is All You Need, Vaswani et al. 2017]

Summary: Self-Attention Block

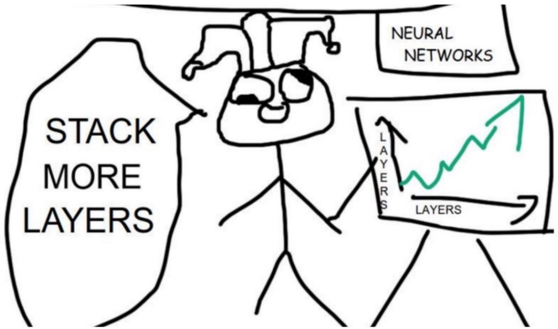
- **Self-Attention:** A critical building block of modern language models.
  - The idea is to compose meanings of words weighted according some similarity notion.

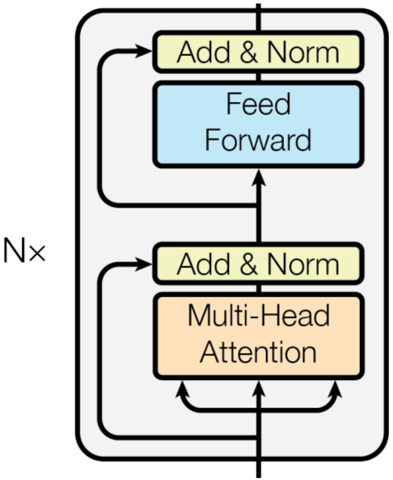
- **Next:** We will combine self-attention blocks to build various architectures known as Transformer.

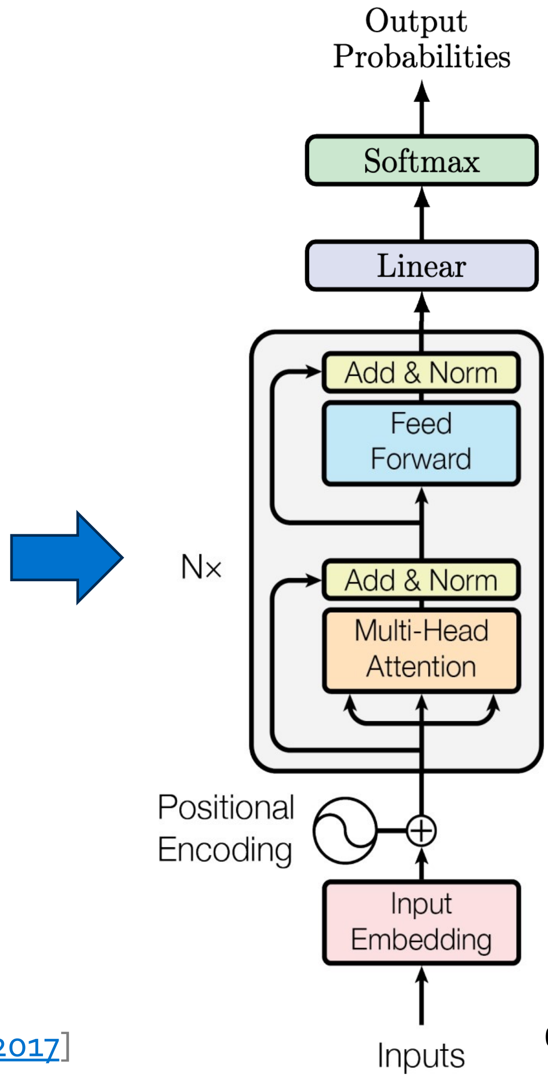
Transformer

How Do We Make it Deep?

- Stack more layers!







[Attention Is All You Need, Vaswani et al. 2017]

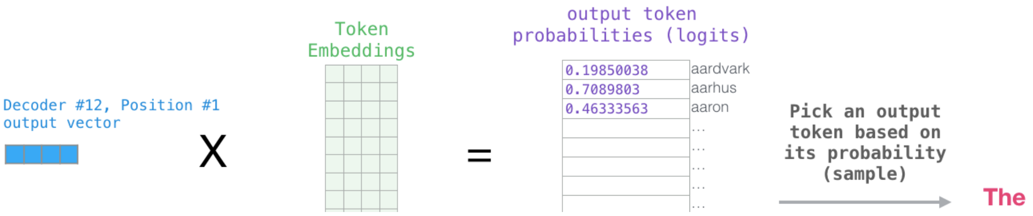
From Representations to Prediction

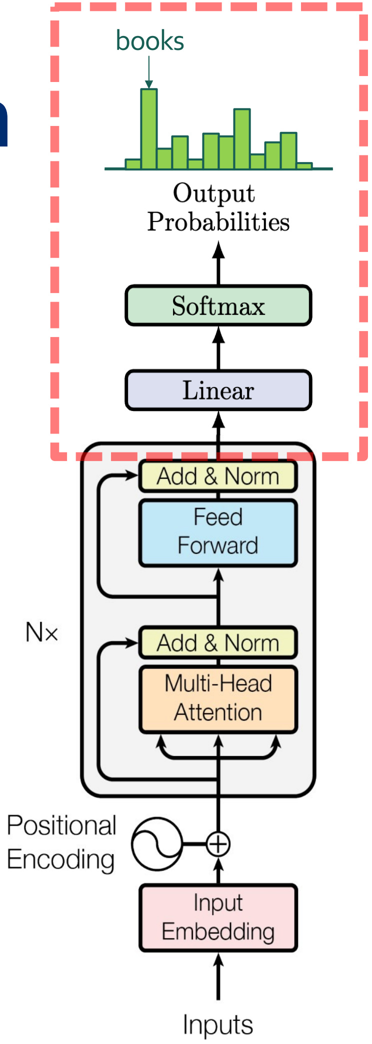
- To perform prediction, add a classification head on top of the final layer of the transformer.
- This can be per token (Language modeling)
- Or can be for the entire sequence (only one token)

$$\text{out} \in \mathbb{R}^{S \times d} \quad \text{(S: Sequence length)}$$

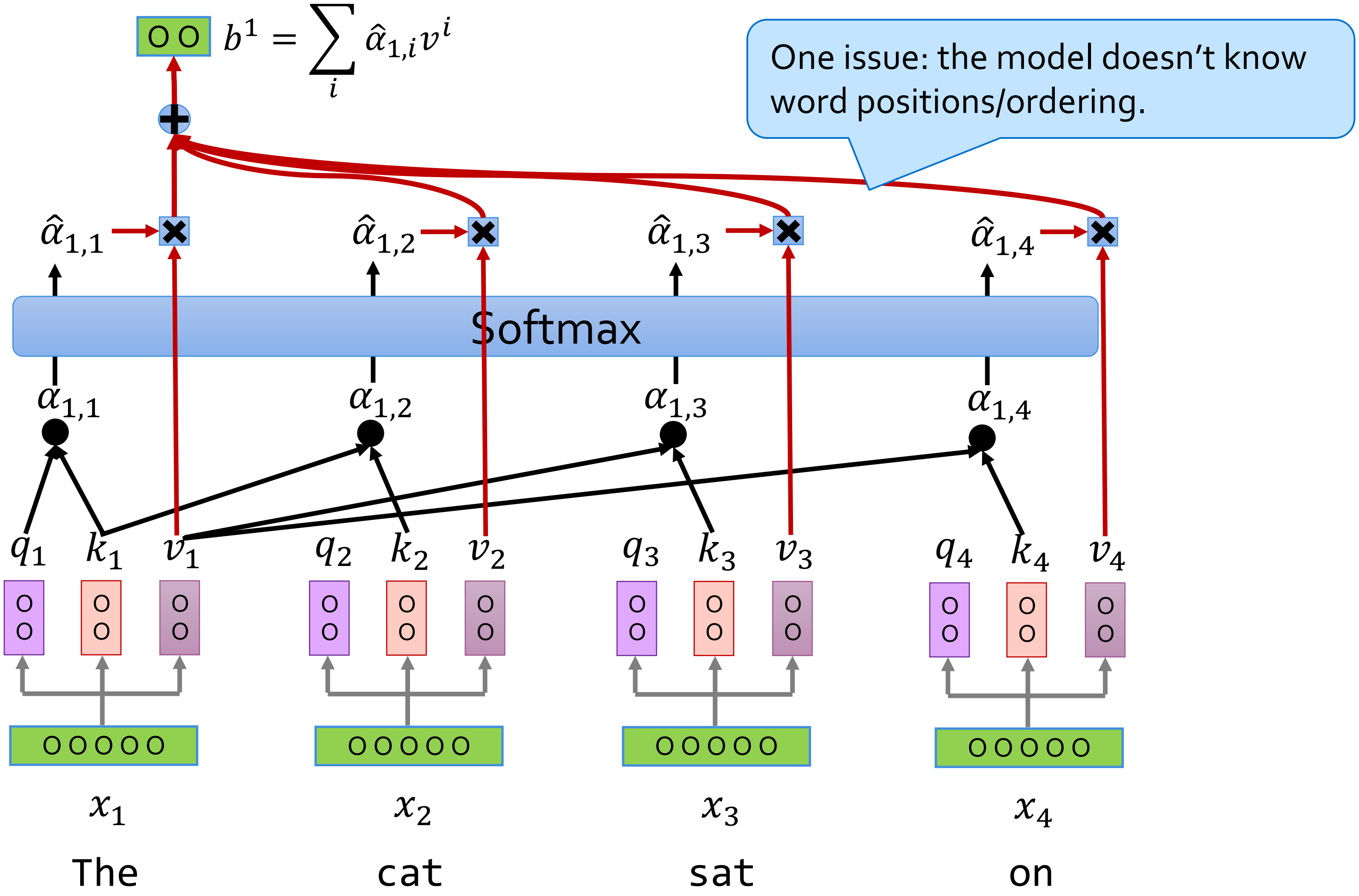
$$\text{logits} = \text{Linear}_{(d,\,V)}(out) = f(out \,.\, W_V) \in \mathbb{R}^{S \times V}$$

$$\text{probabilies} = \text{softmax}(\text{logits}) \in \mathbb{R}^{S \times V}$$

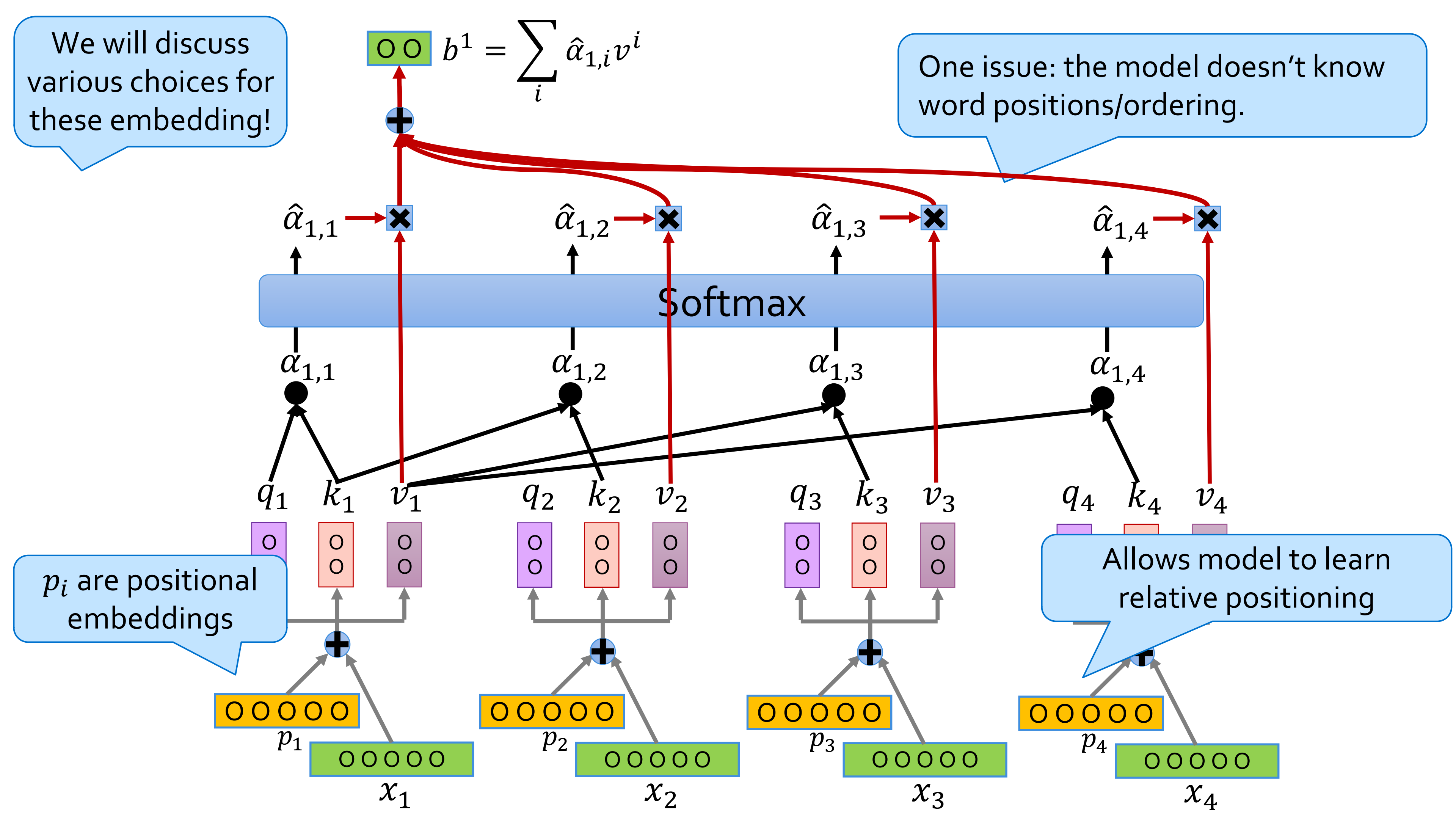




One last wrinkle though …



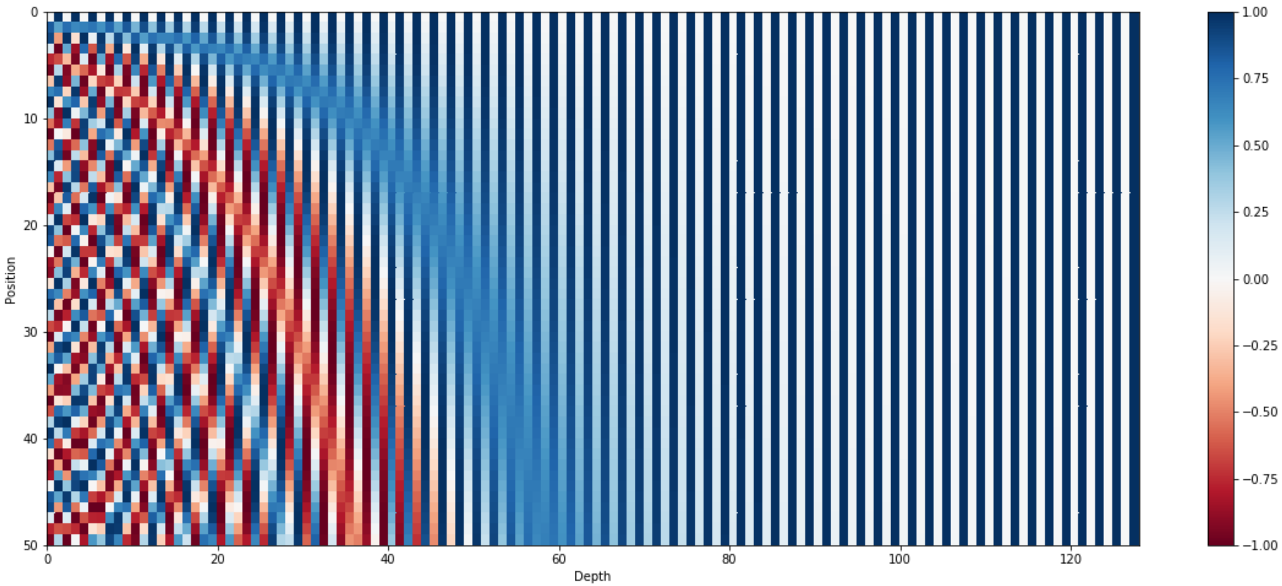
One issue: the model doesn't know word positions/ordering.
$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
$\hat{\alpha}_{1,1}$ $\hat{\alpha}_{1,2}$ $\hat{\alpha}_{1,3}$ $\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$ $\alpha_{1,2}$ $\alpha_{1,3}$ $\alpha_{1,4}$
q_1 k_1 v_1 q_2 k_2 v_2 q_3 k_3 v_3 q_4 k_4 v_4
x_1 x_2 x_3 x_4
The cat sat on

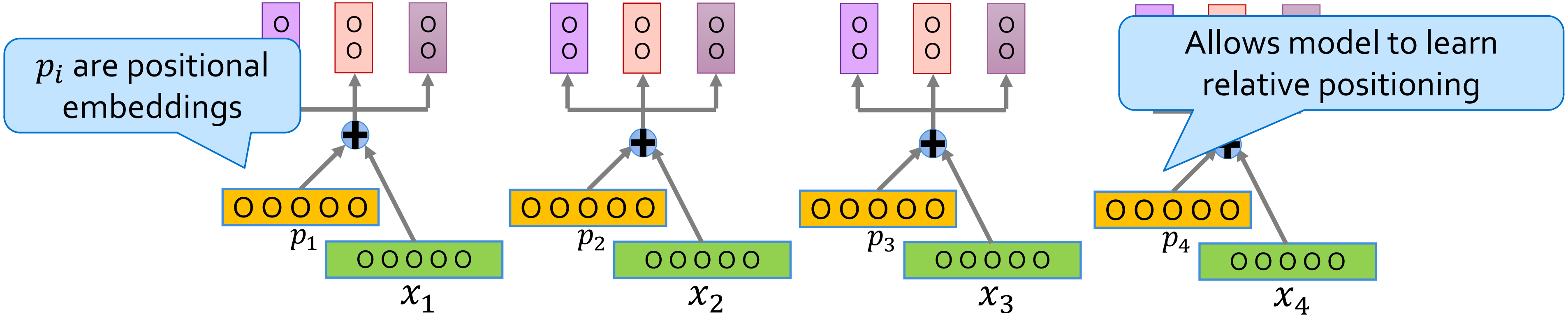


We will discuss various choices for these embedding!
$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
One issue: the model doesn't know word positions/ordering.
$\hat{\alpha}_{1,1}$ $\hat{\alpha}_{1,2}$ $\hat{\alpha}_{1,3}$ $\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$ $\alpha_{1,2}$ $\alpha_{1,3}$ $\alpha_{1,4}$
q_1 k_1 v_1 q_2 k_2 v_2 q_3 k_3 v_3 q_4 k_4 v_4
p_i are positional embeddings
Allows model to learn relative positioning
p_1 p_2 p_3 p_4
x_1 x_2 x_3 x_4

An approach:

Sine/Cosine encoding

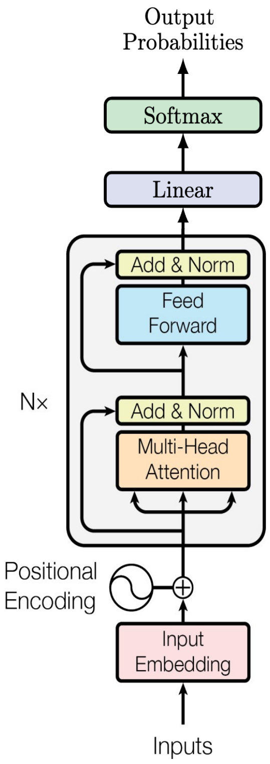




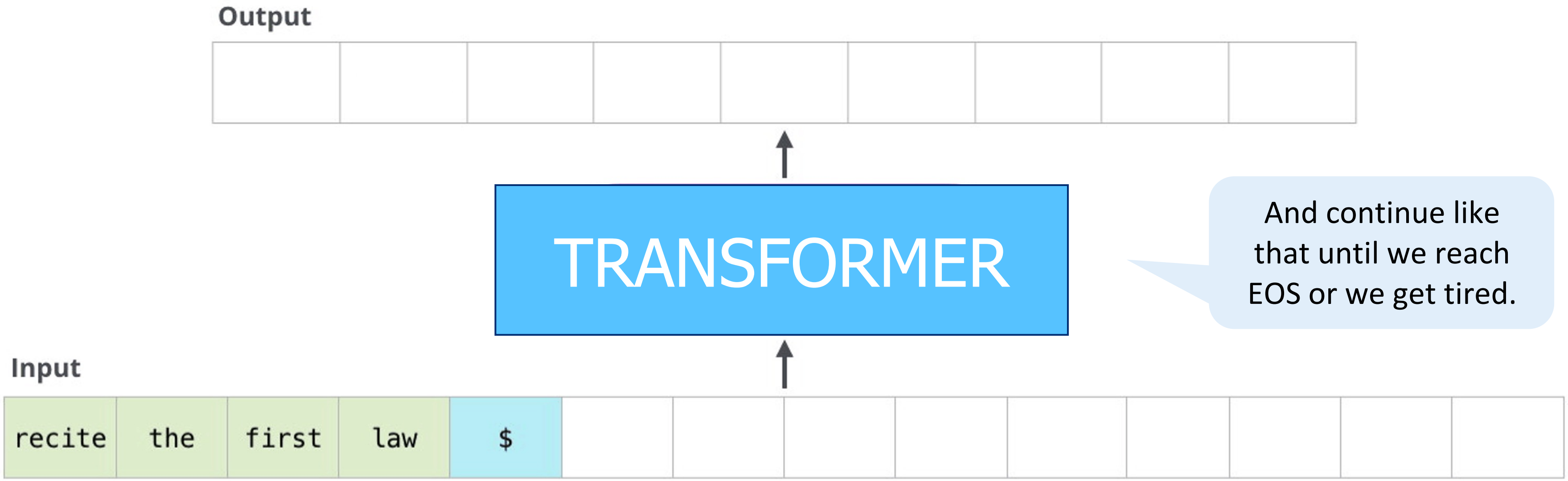
The Transformer Stack in PyTorch

```python
class Block(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.ln_1 = LayerNorm(config.n_embd, bias=config.bias)
        self.attn = CausalSelfAttention(config)
        self.ln_2 = LayerNorm(config.n_embd, bias=config.bias)
        self.mlp = MLP(config)

    def forward(self, x):
        x = x + self.attn(self.ln_1(x))
        x = x + self.mlp(self.ln_2(x))
        return x
```

```python
self.transformer = nn.ModuleDict(
    dict(
        wte=nn.Embedding(config.vocab_size, config.n_embd),
        wpe=nn.Embedding(config.block_size, config.n_embd),
        drop=nn.Dropout(config.dropout),
        h=nn.ModuleList([Block(config) for _ in range(config.n_layer)]),
        ln_f=LayerNorm(config.n_embd, bias=config.bias),
    )
)
self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)
```



Transformer-based Language Modeling

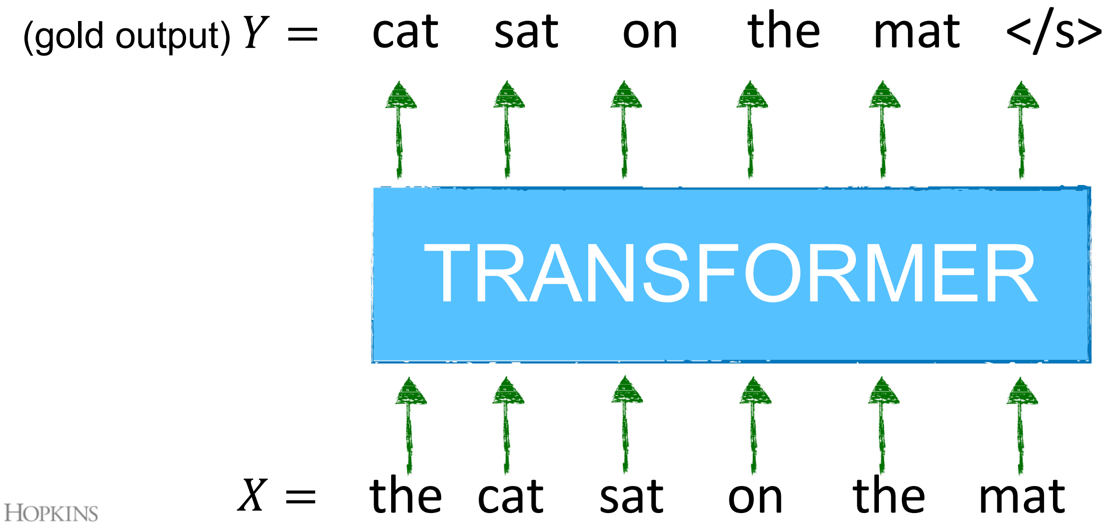


And continue like that until we reach EOS or we get tired.

Image by http://jalammar.github.io/illustrated-gpt2/

Training a Transformer Language Model

- **Goal:** Train a Transformer for language modeling (i.e., predicting the next word).
- **Approach:** Train it so that each position is predictor of the next (right) token.
  - We just shift the input to right by one, and use as labels



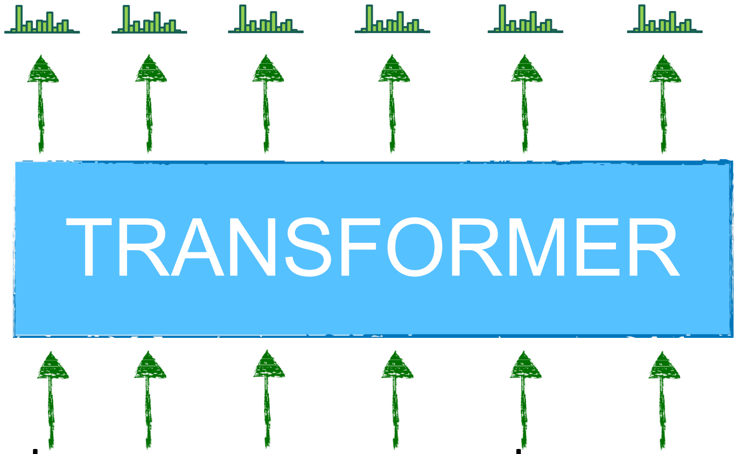

```
X = text[:, :-1]
Y = text[:, 1:]
```

[Slide credit: Arman Cohan]

Training a Transformer Language Model

- For each position, compute their corresponding **distribution** over the whole vocab.

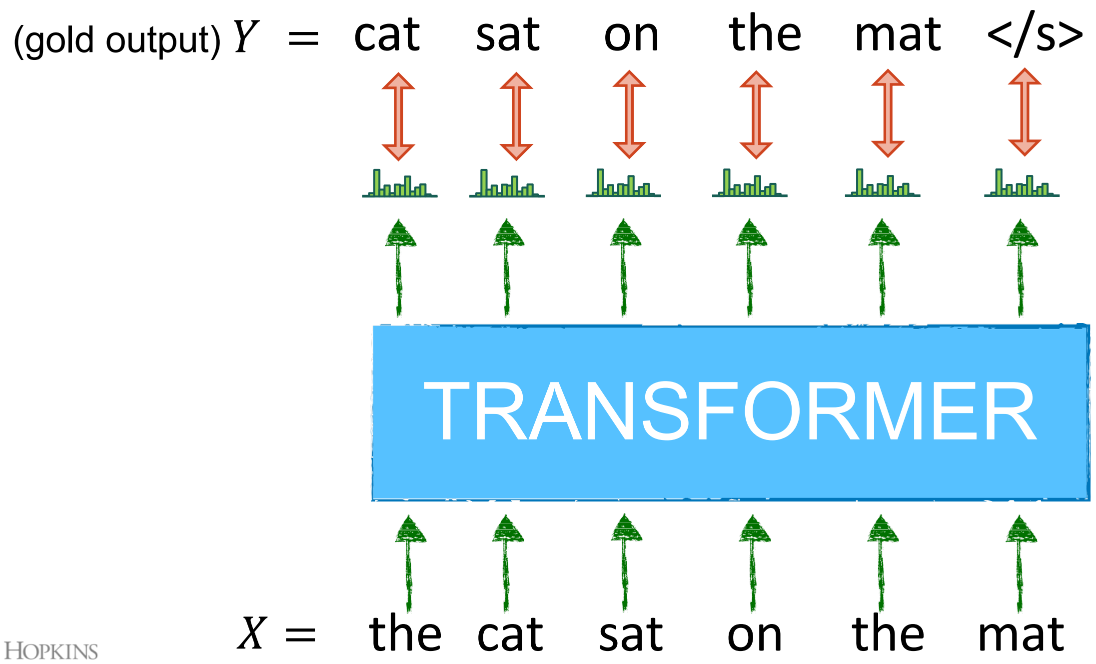
(gold output) Y = cat sat on the mat </s>



X = the cat sat on the mat

Training a Transformer Language Model

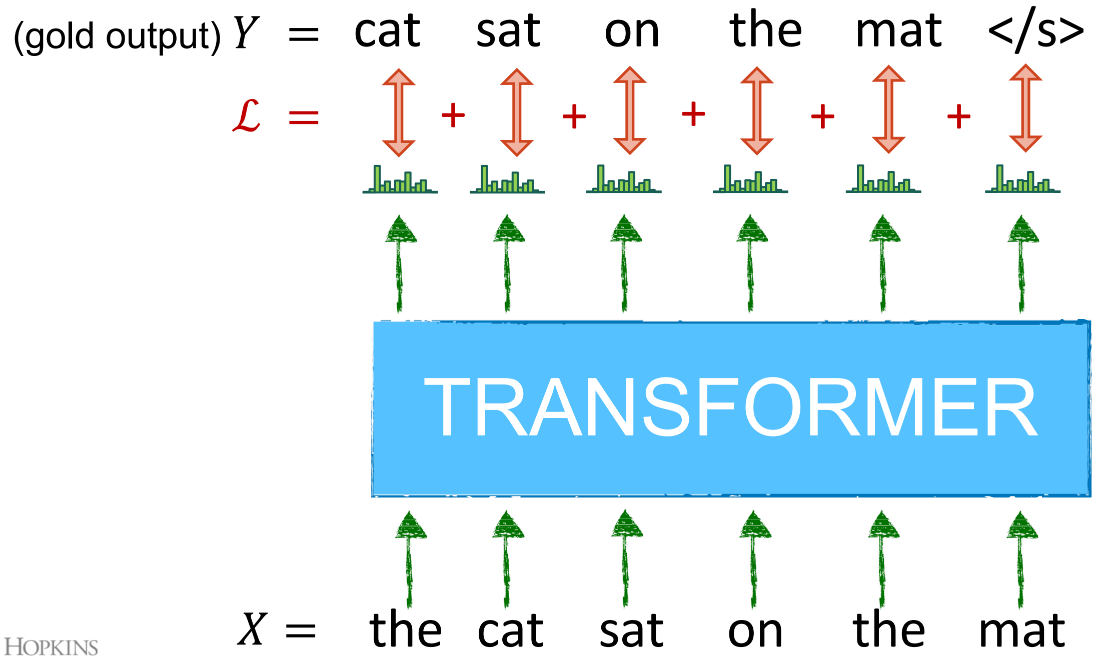
- For each position, compute the **loss** between the distribution and the gold output label.



Johns Hopkins Whiting School of Engineering

Training a Transformer Language Model

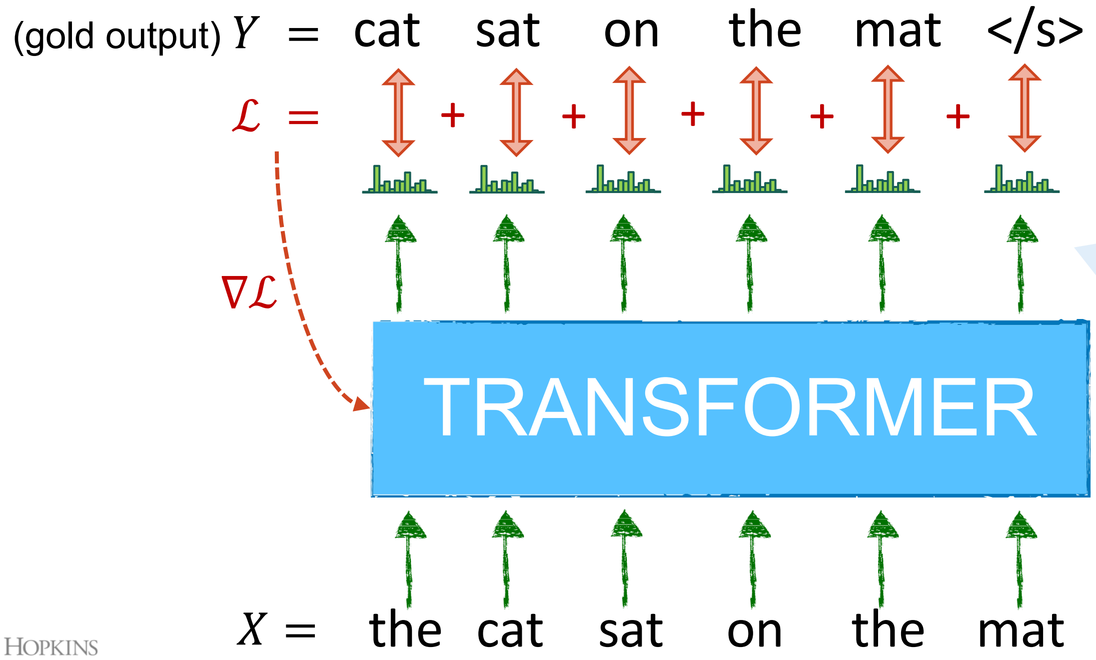
- Sum the position-wise loss values to a obtain a **global loss**.



JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Training a Transformer Language Model

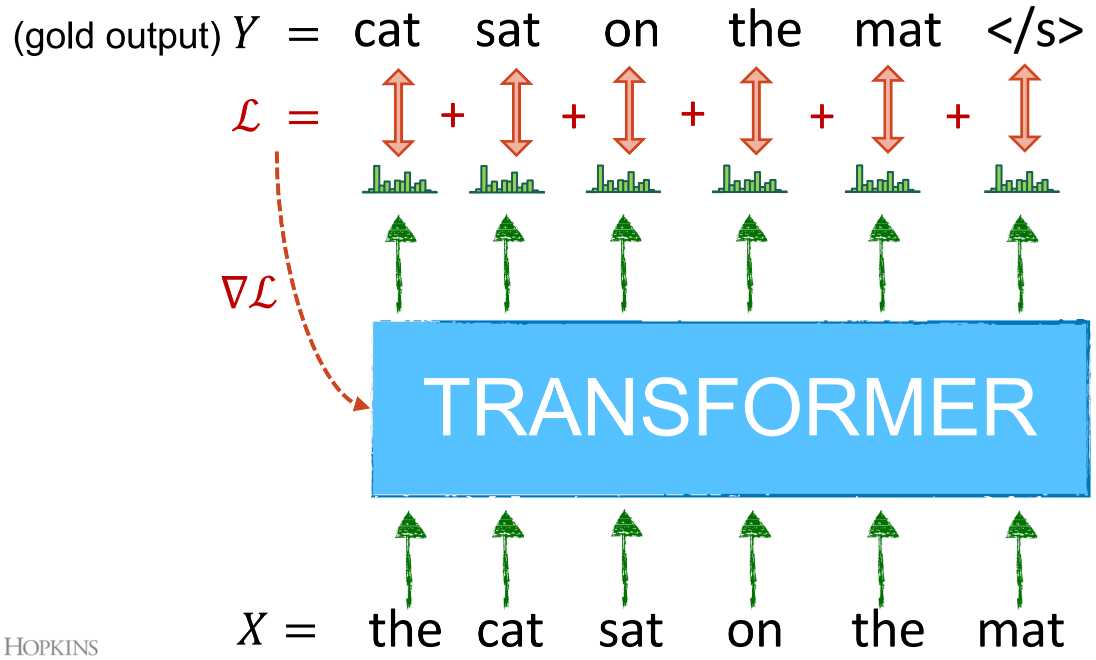
- Using this loss, do **Backprop** and **update** the Transformer parameters.



Well, this is not quite right 🤡 …
what is the problem with this?

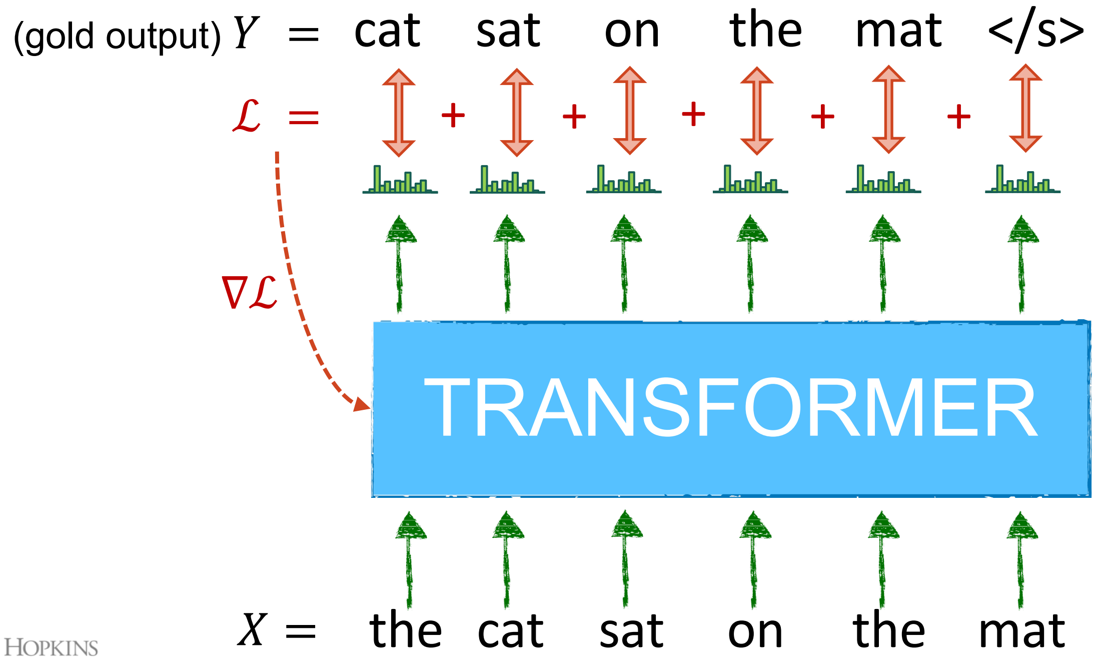
Training a Transformer Language Model

- The model would solve the task by copying the next token to output (data leakage).
  - Does not learn anything useful



Training a Transformer Language Model

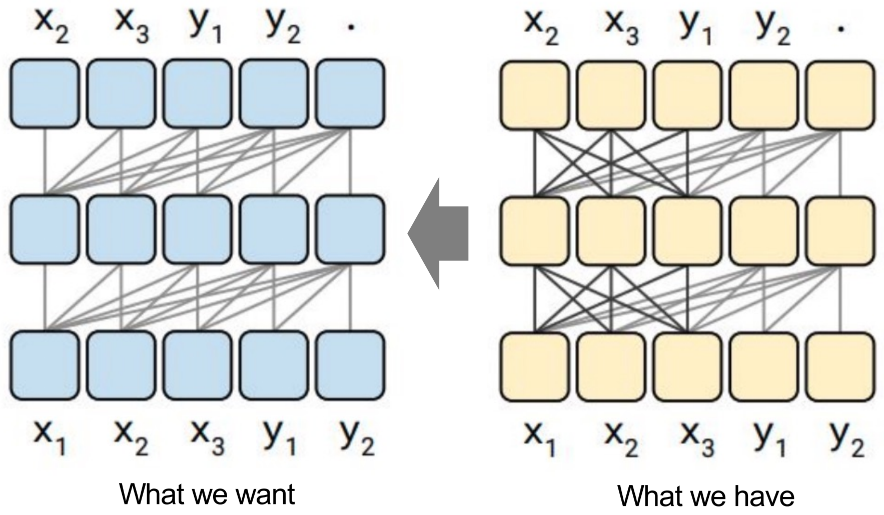
- We need to **prevent information leakage** from future tokens! How?



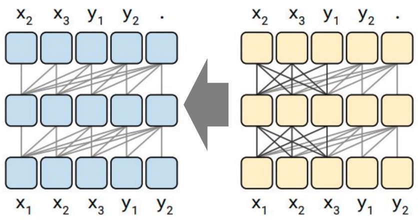
Attention mask

Attention raw scores

	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11



Attention mask



Attention raw scores

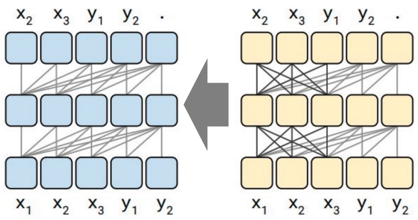
	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11

Attention mask

	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
1	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
2	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
3	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
4	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf
5	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Slide credit: Arman Cohan

Attention mask



Attention raw scores

	1	2	3	4	5	6	7	8	9	10		
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11

X

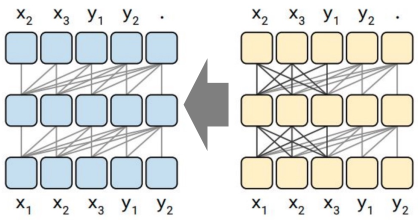
Attention mask

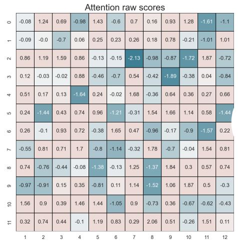
	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
1	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
2	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
3	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
4	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf
5	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

0.30 0.25 0.20 0.15 0.10 0.05 0.00

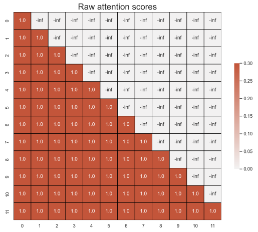
Note matrix multiplication is quite fast in GPUs.

Attention mask

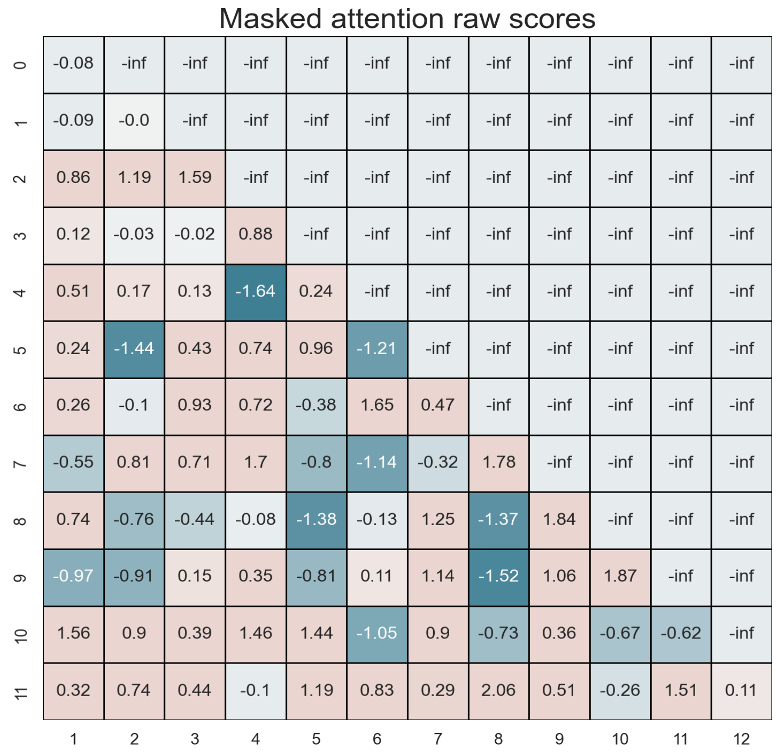




X

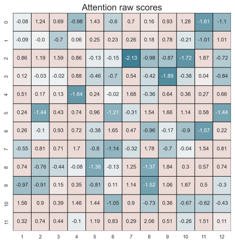


=

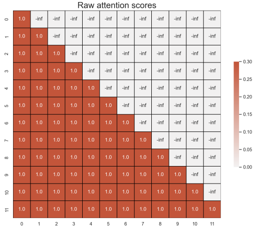


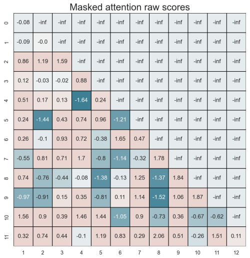
Attention mask

The effect is more than just pruning out some of the wirings in self-attention block.

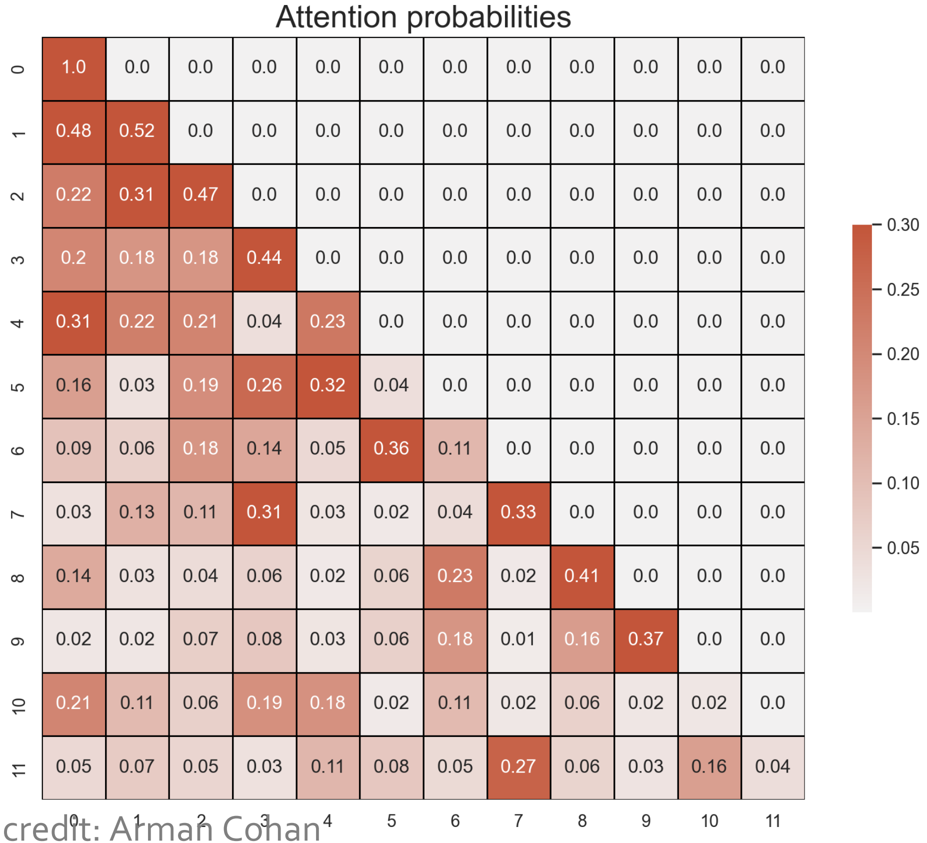


X





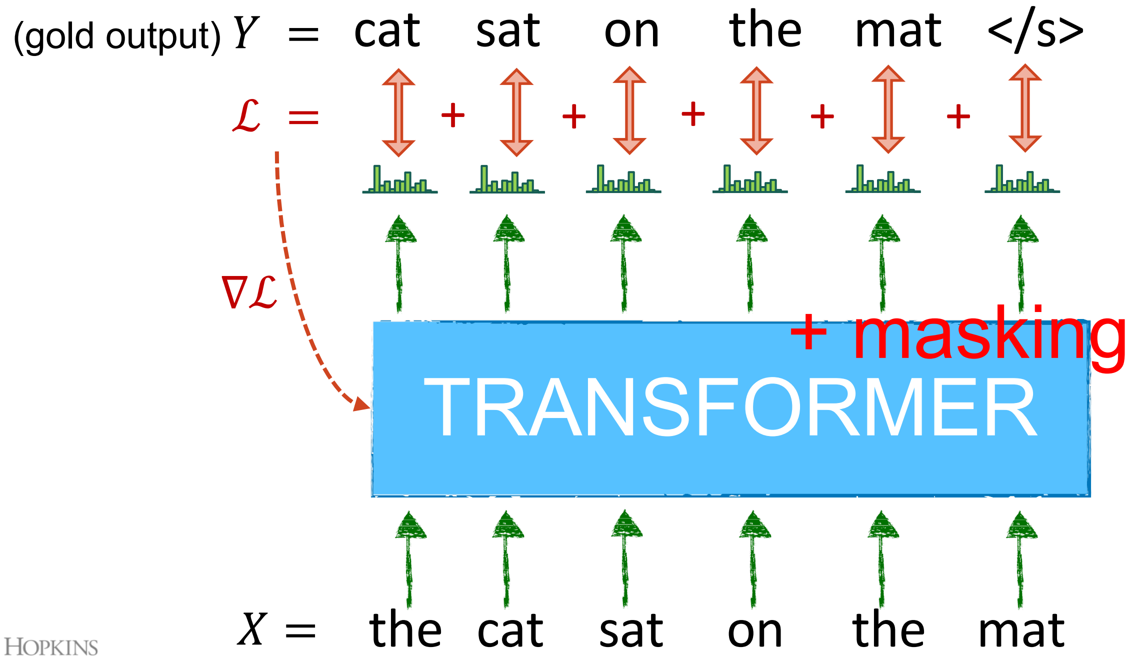
softmax



Slide credit: Arman Cohan

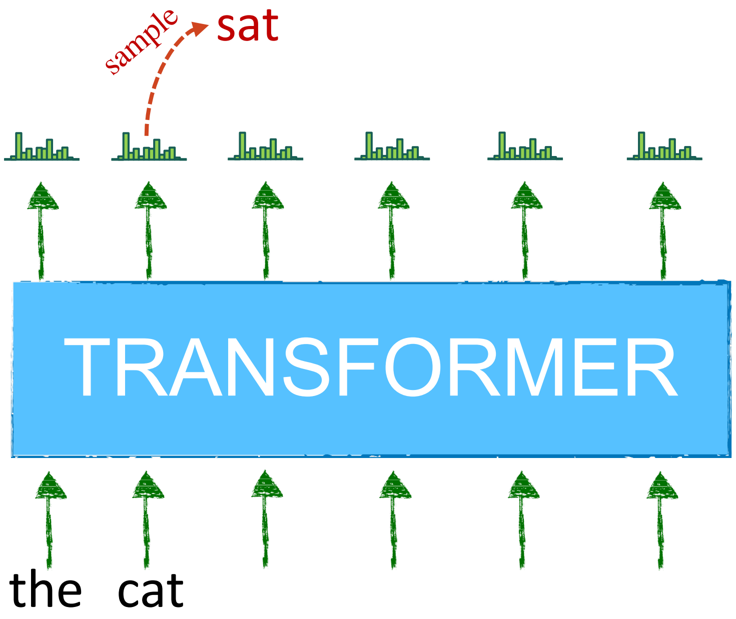
Training a Transformer Language Model

- We need to **prevent information leakage** from future tokens! How?



How to use the model to generate text?

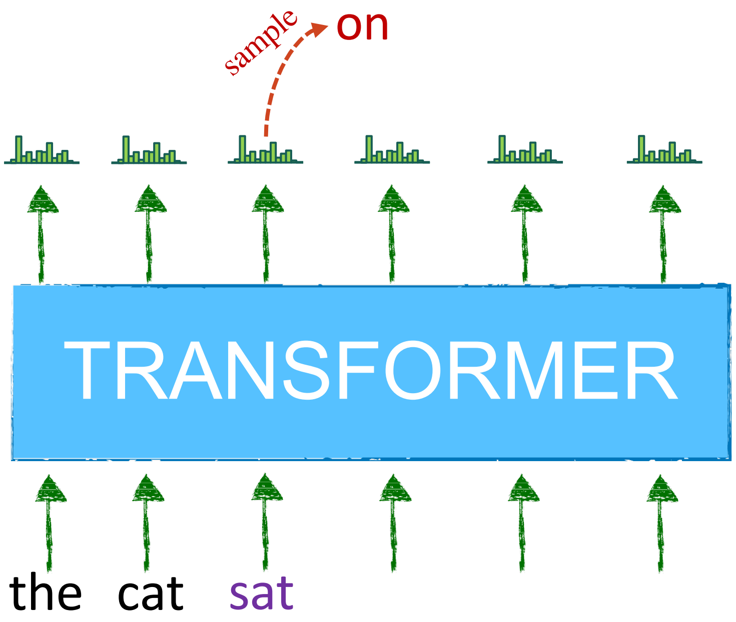
- Use the output of previous step as input to the next step repeatedly



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

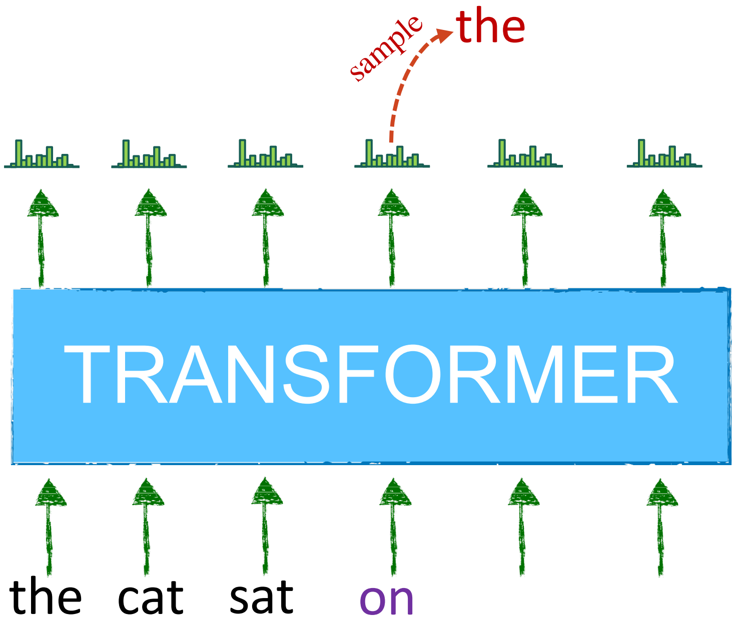
The probabilities get revised upon adding a new token to the input.



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

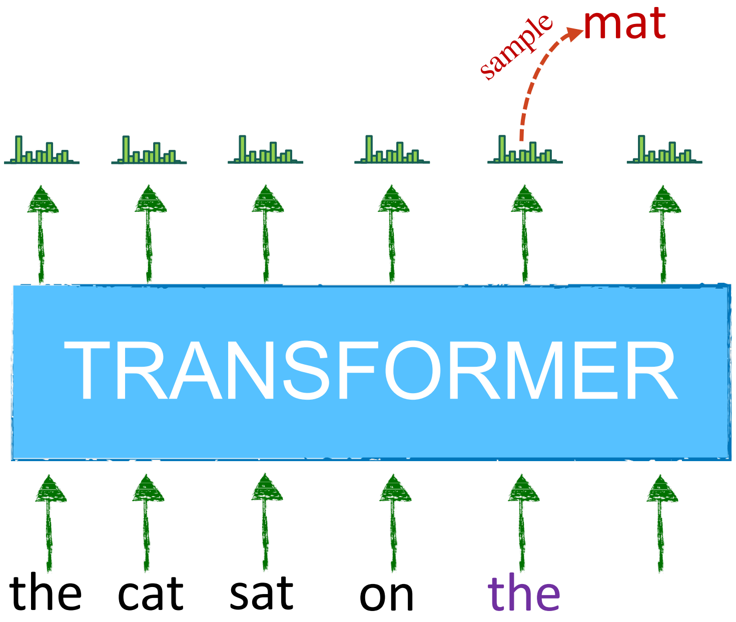
The probabilities get revised upon adding a new token to the input.



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

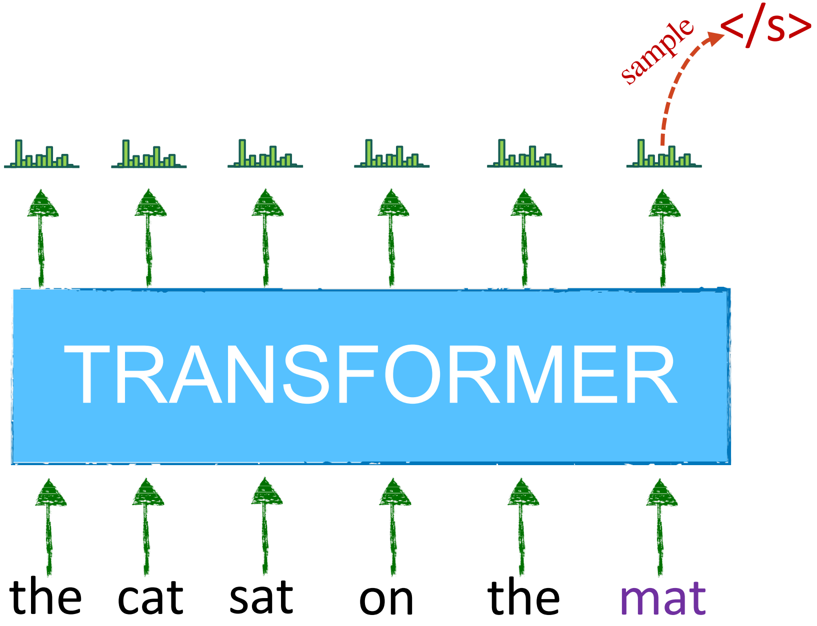
The probabilities get revised upon adding a new token to the input.



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly

The probabilities get revised upon adding a new token to the input.

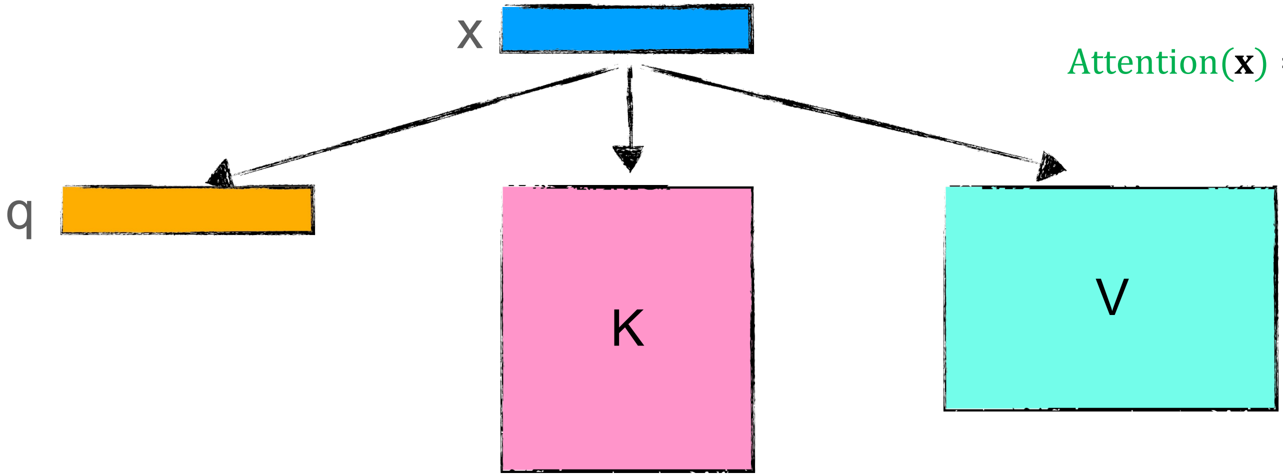


An important efficiency consideration about decoding!

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

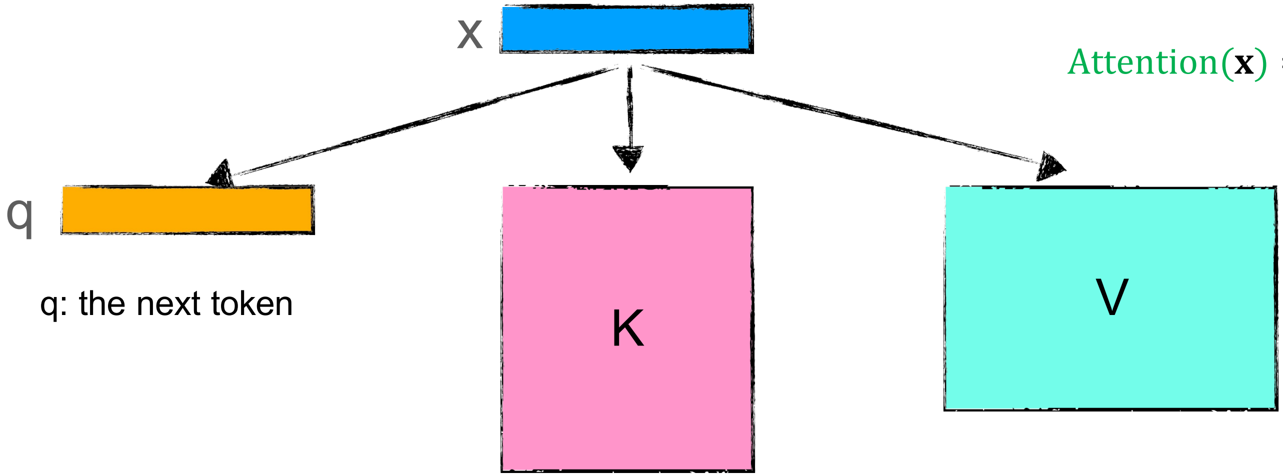


[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



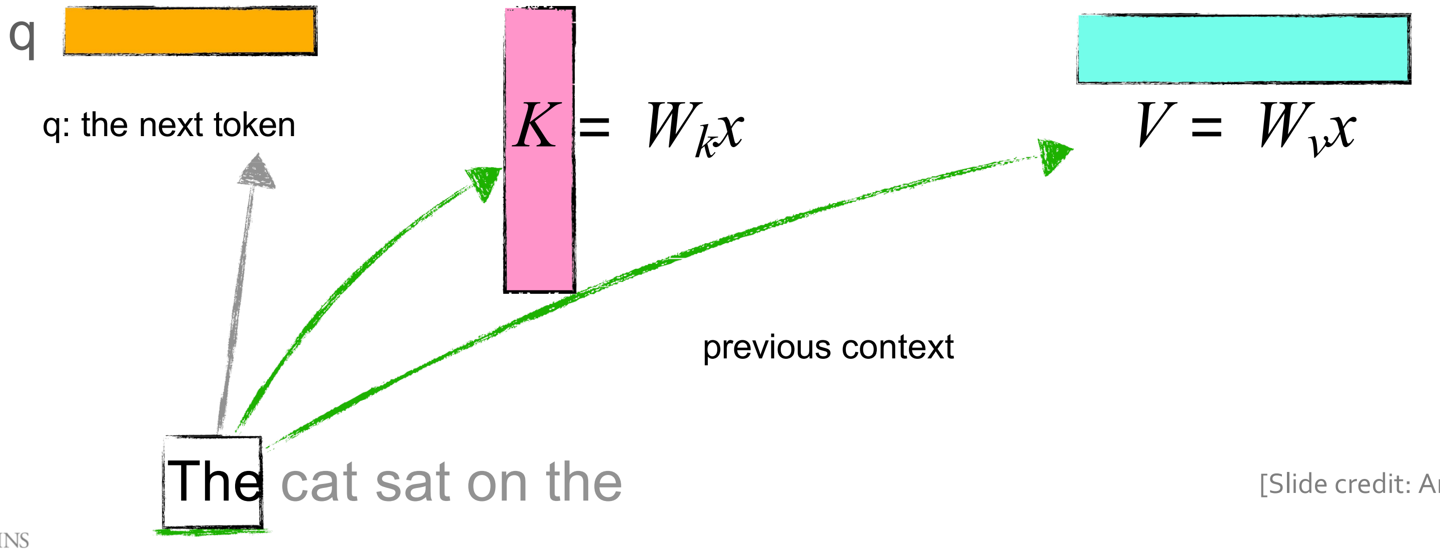
q: the next token

previous context

[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

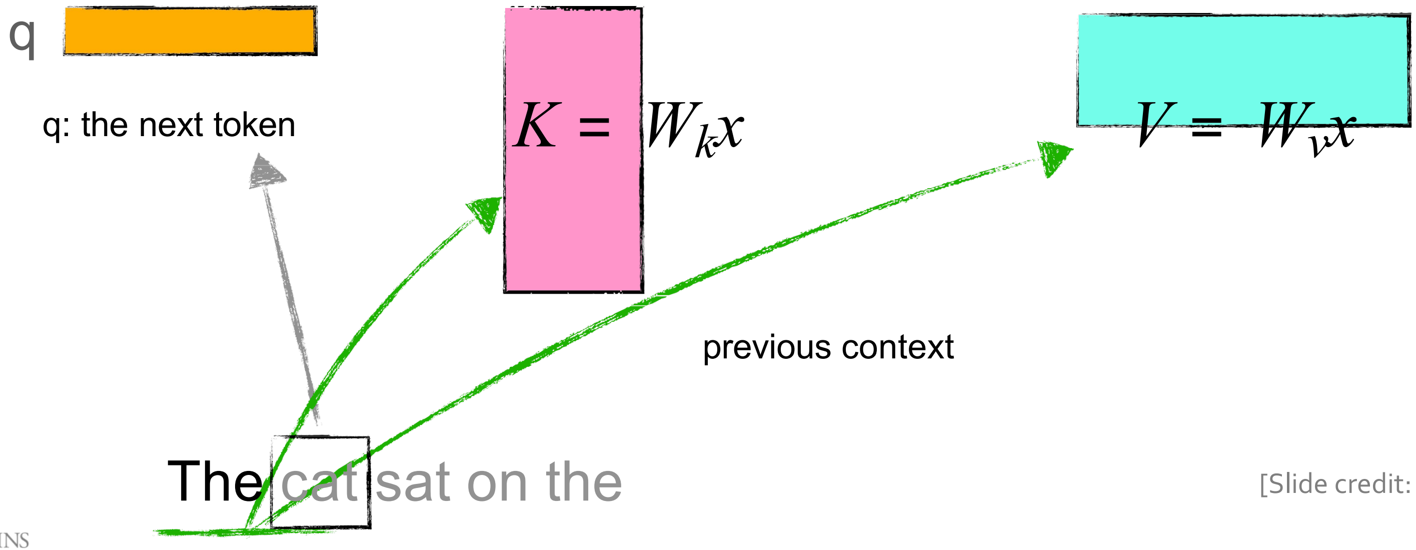


[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

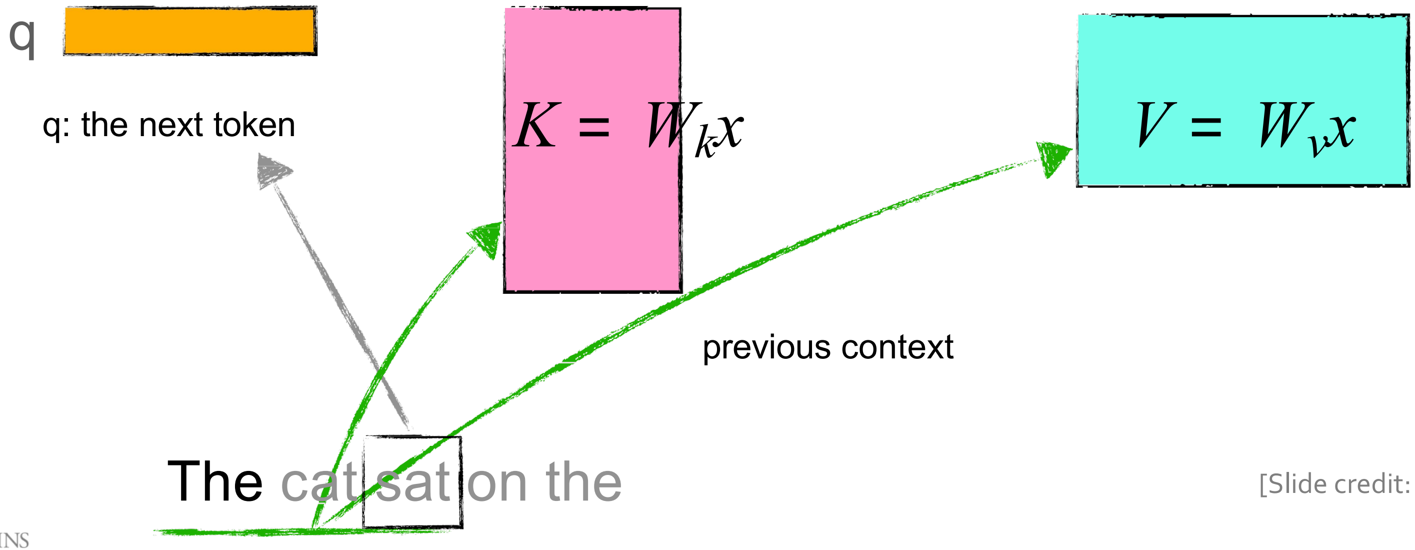


[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

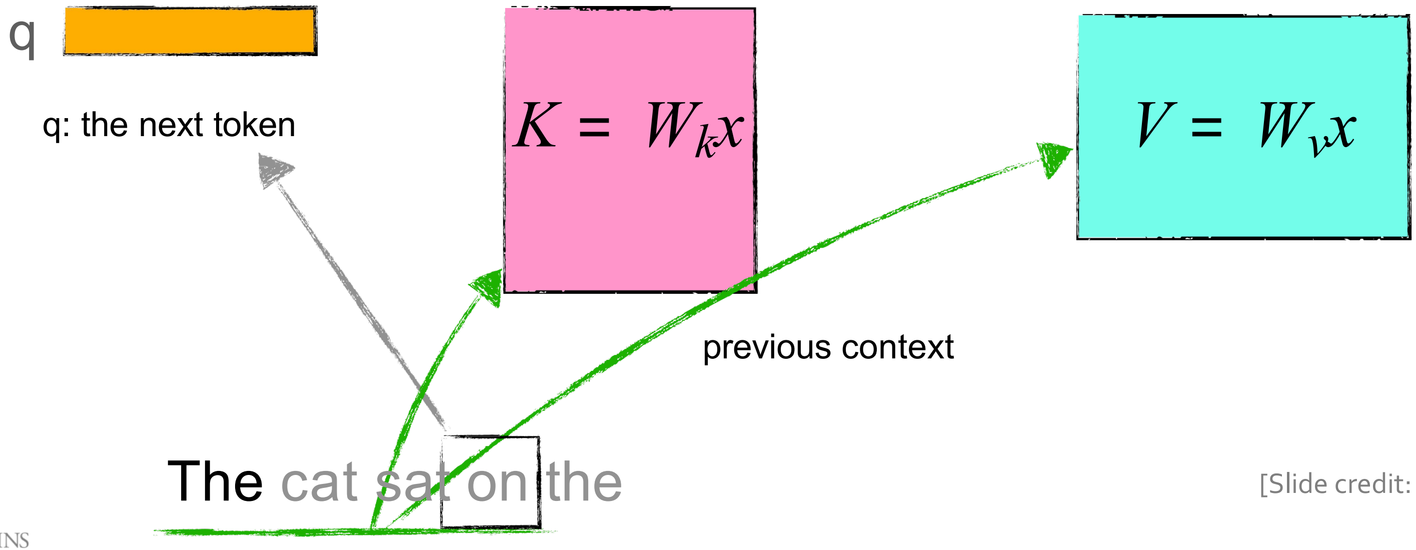


[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

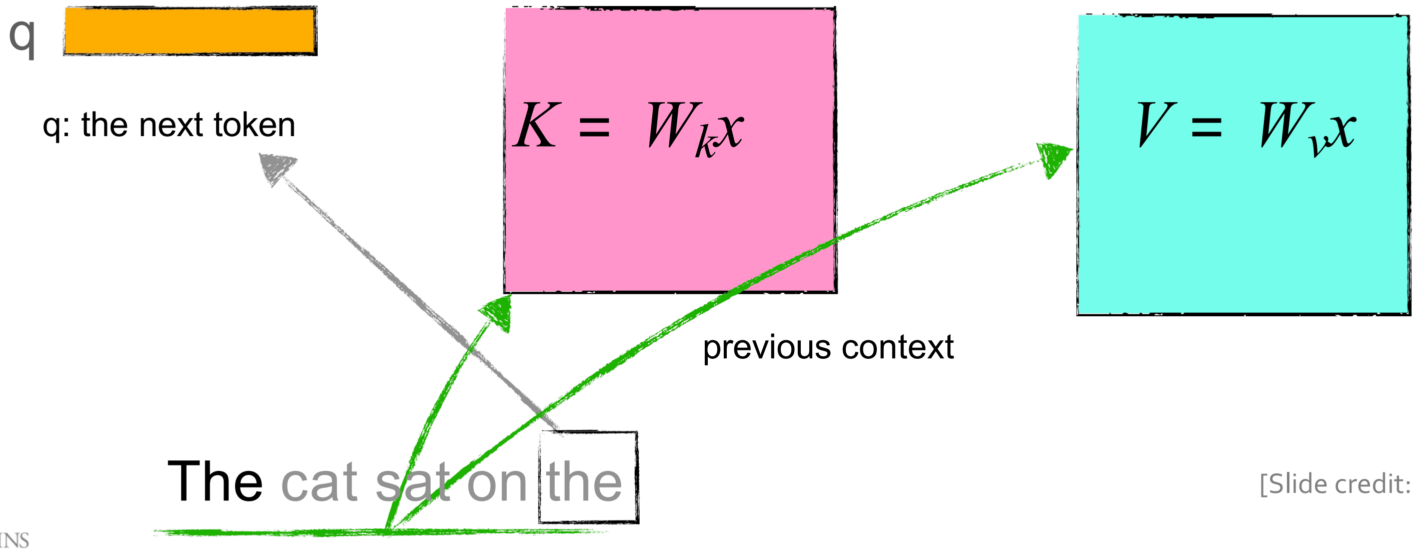
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

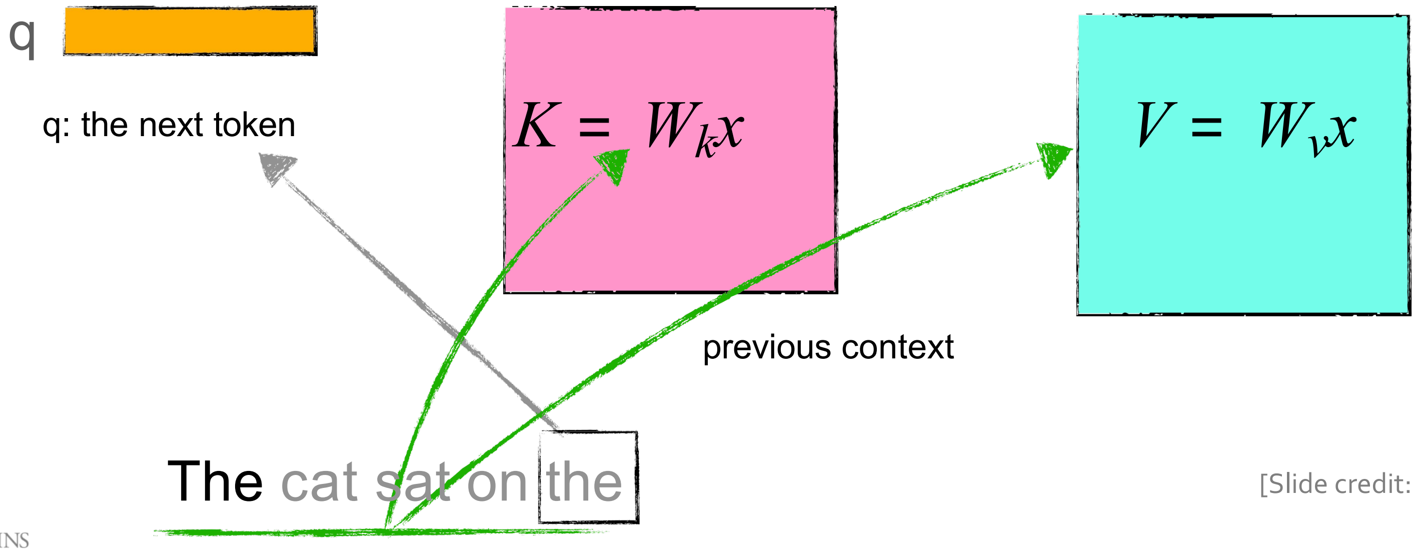


[Slide credit: Arman Cohan]

Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

- We are computing the Keys and Values many times!
  - Let's reduce redundancy! 😤



[Slide credit: Arman Cohan]

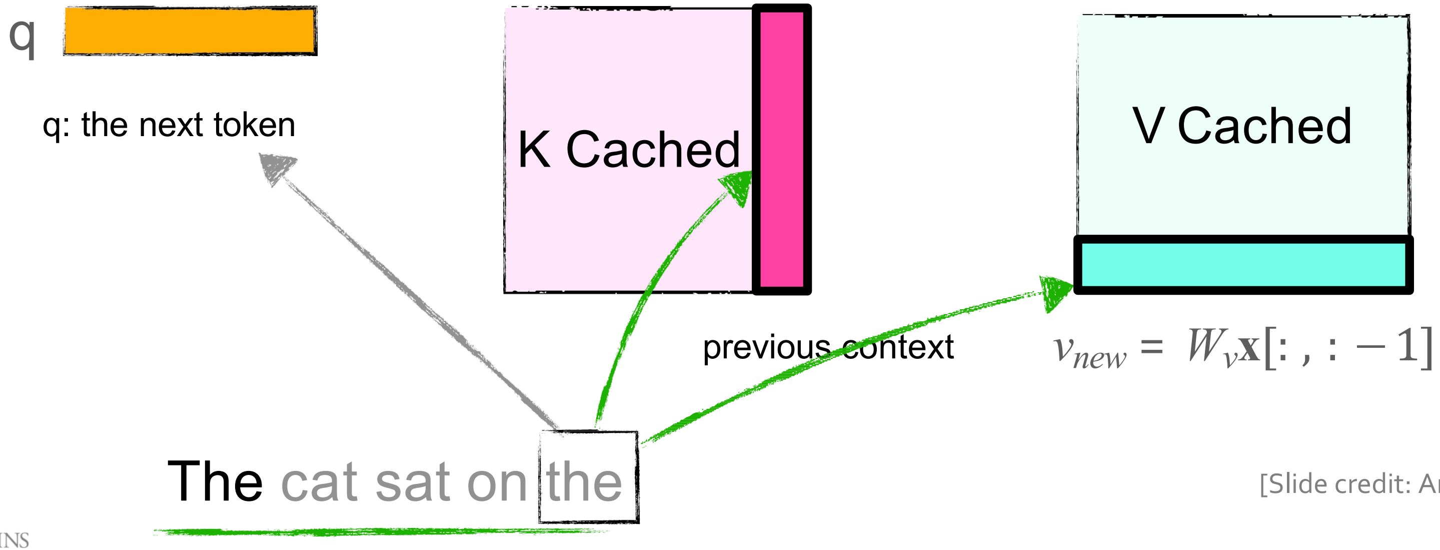
Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

- We are computing the Keys and Values many times!
    - Let's reduce redundancy! 😤

$$k_{new} = W_k \mathbf{x}[:,:-1]$$



$$v_{new} = W_v \mathbf{x}[:,:-1]$$

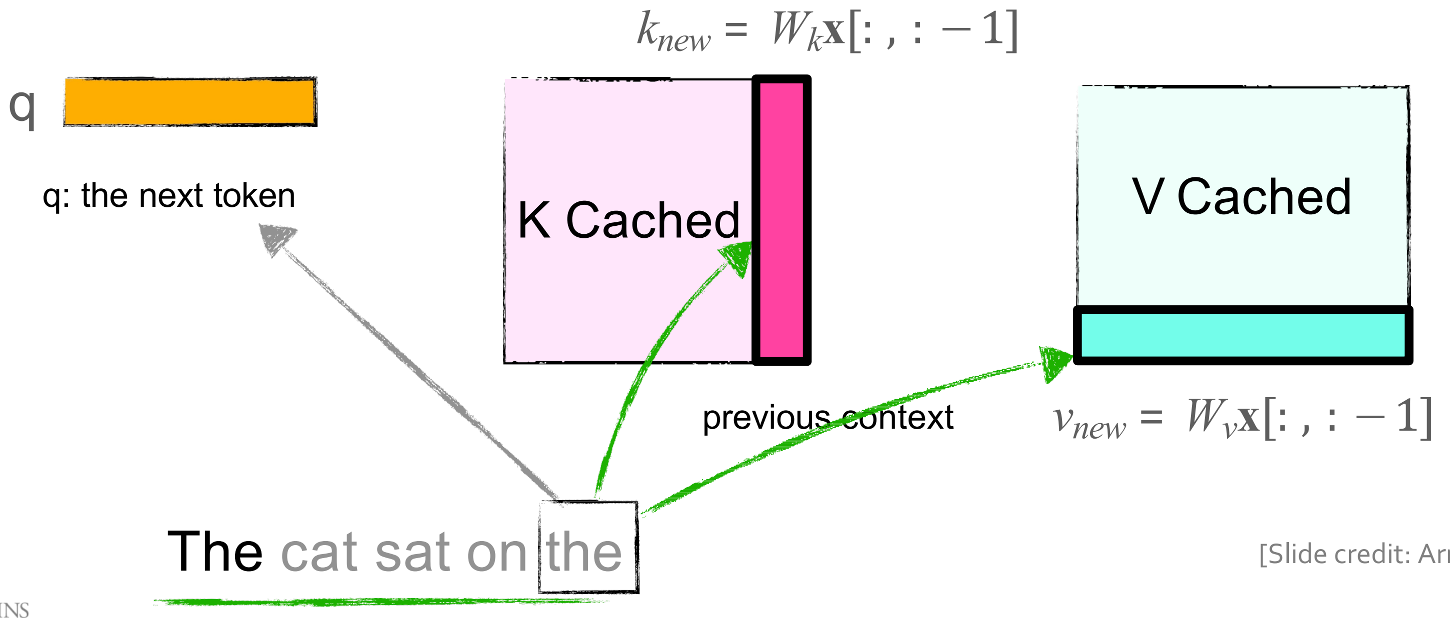
[Slide credit: Arman Cohan]

Making decoding more efficient

$Q = \mathbf{W}^q \mathbf{x}$
$K = \mathbf{W}^k \mathbf{x}$
$V = \mathbf{W}^v \mathbf{x}$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d}}\right)V$$

- **Question:** How much memory does this K, V cache require?



[Slide credit: Arman Cohan]

Summary

- This is a very generic Transformer!
- We will implement this in HW5 to build a simple Transformer Language Model!!

- **Next:**
  - Architectural variants
  - Efficiency issues.
  - ...

Transformer Architectural Variants

Encoder-decoder

- It is possible to have two stacks of transformer layers
- The encoder is as we’ve seen
- We can also add a decoder layer that is identical to the encoder but we give it the ability to also attend to the input

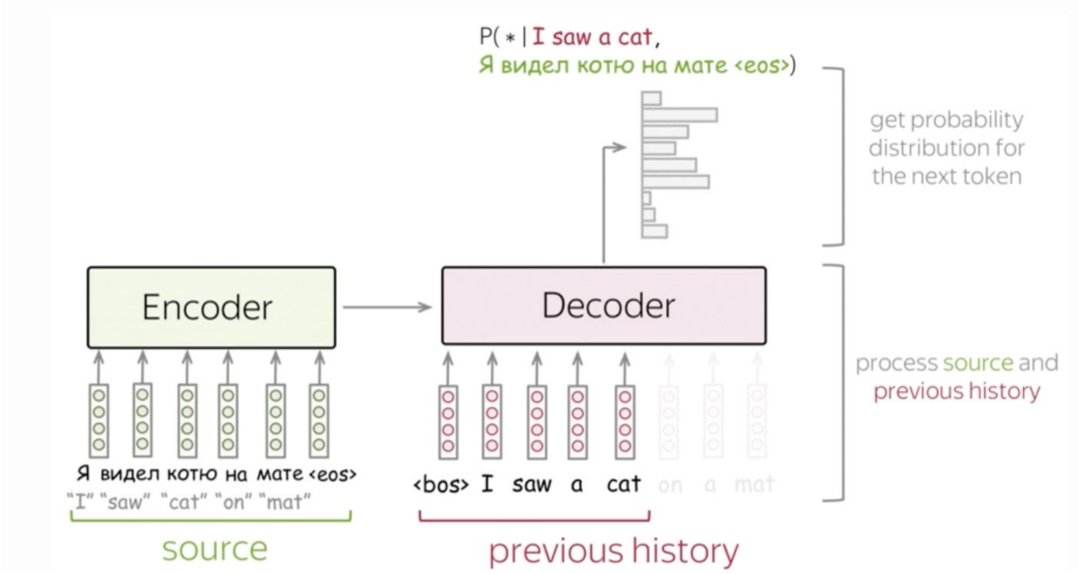
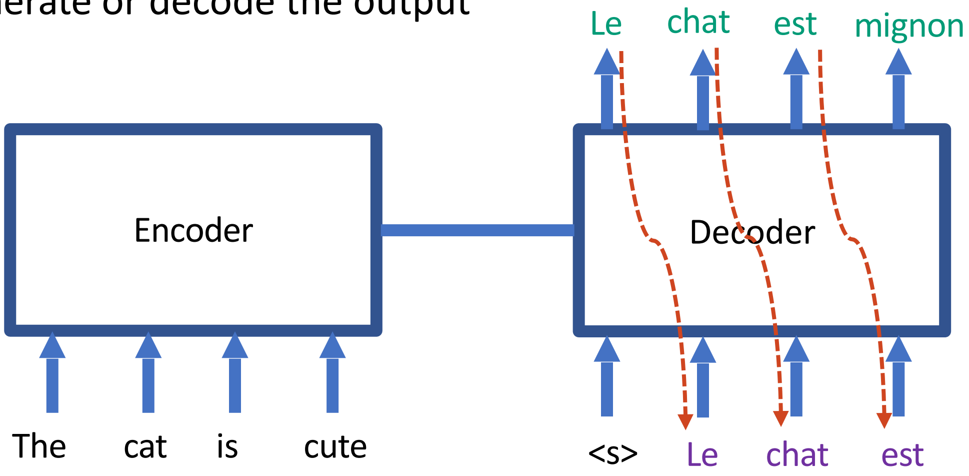


Fig from: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

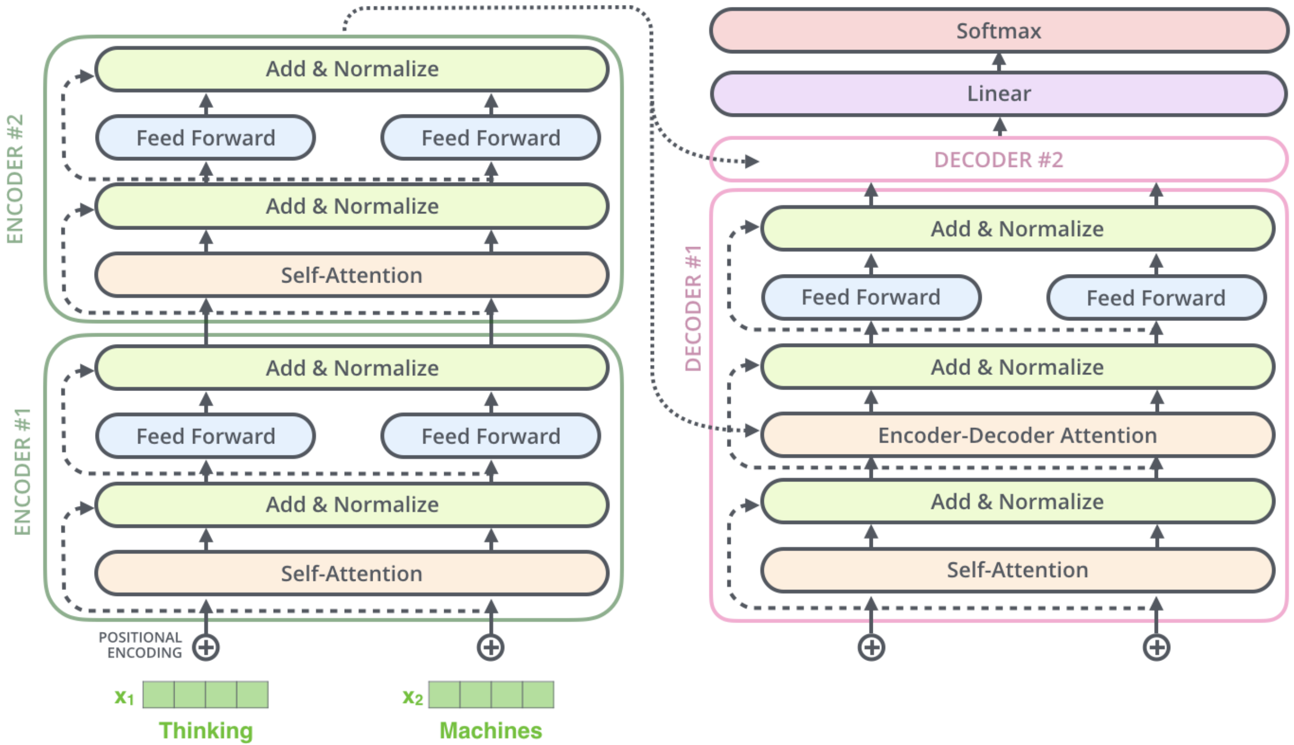
Encoder-decoder models

- Encoder = read or encode the input,
- Decoder = generate or decode the output



Transformer [Vaswani et al. 2017]

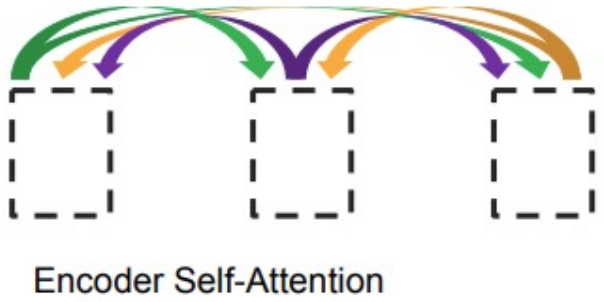
- An encoder-decoder architecture built with attention modules.

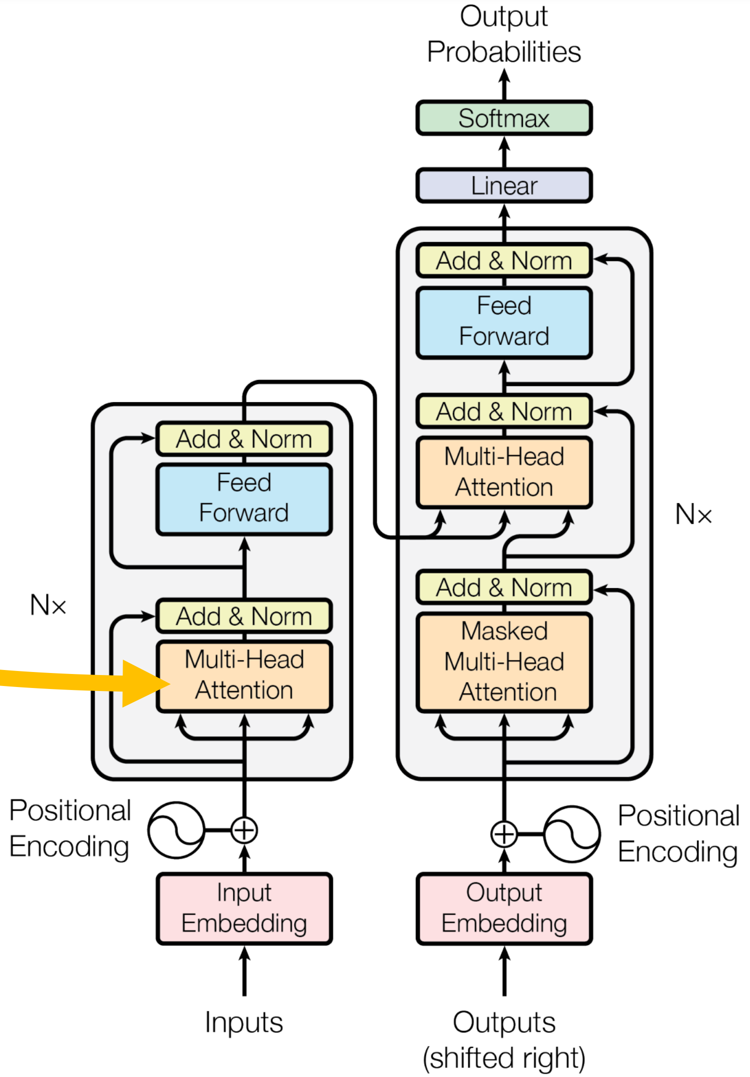


74

Transformer [Vaswani et al. 2017]

- Computation of **encoder** attends to both sides.

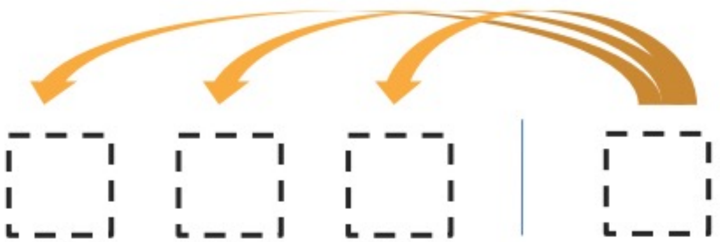




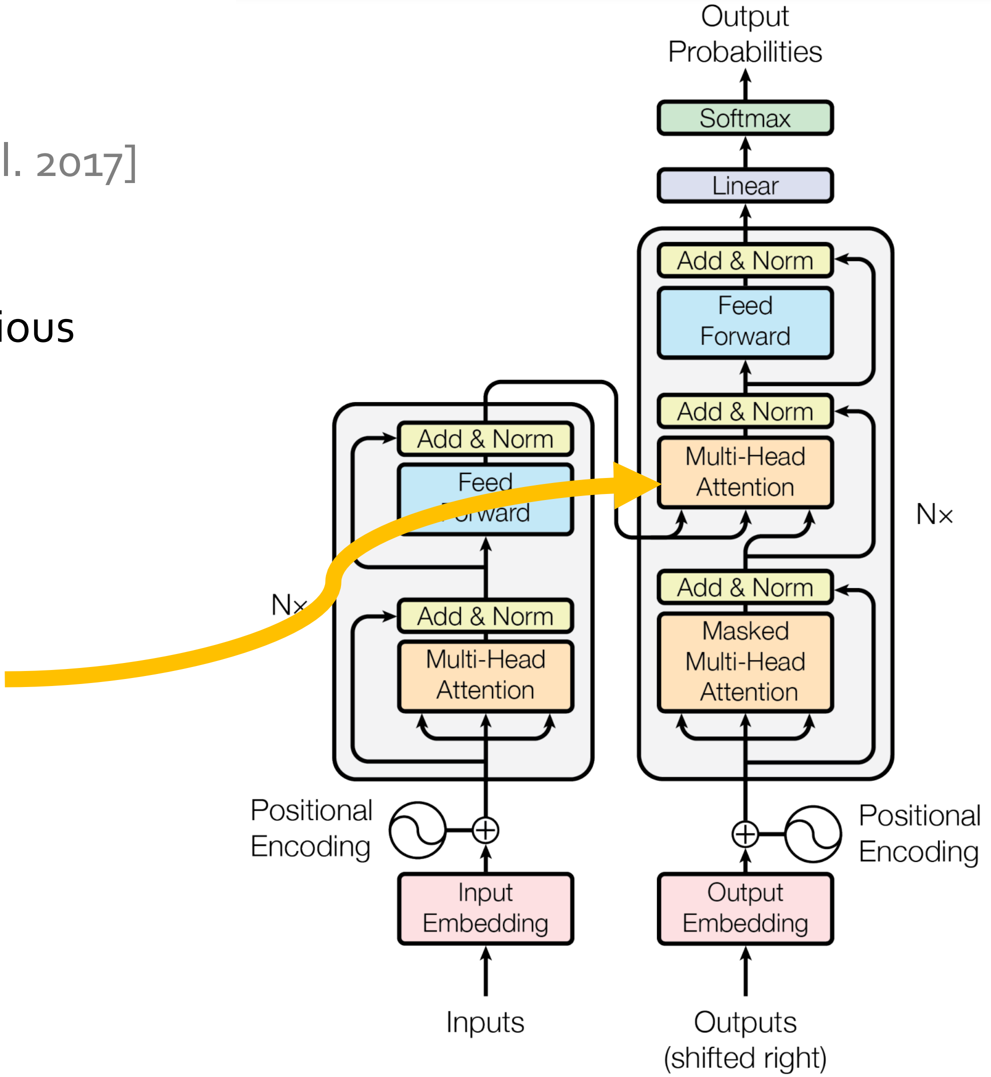
[Attention Is All You Need, Vaswani et al. 2017]

Transformer [Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder**



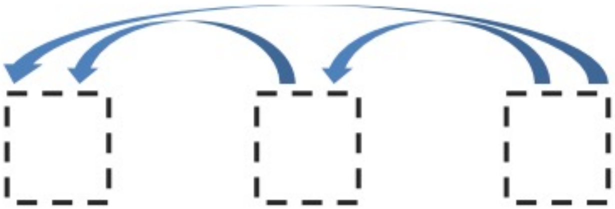
Encoder-Decoder Attention



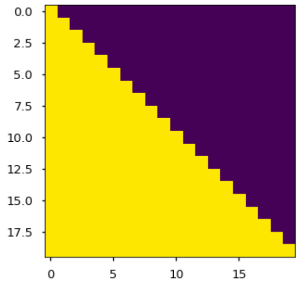
[Attention Is All You Need, Vaswani et al. 2017]

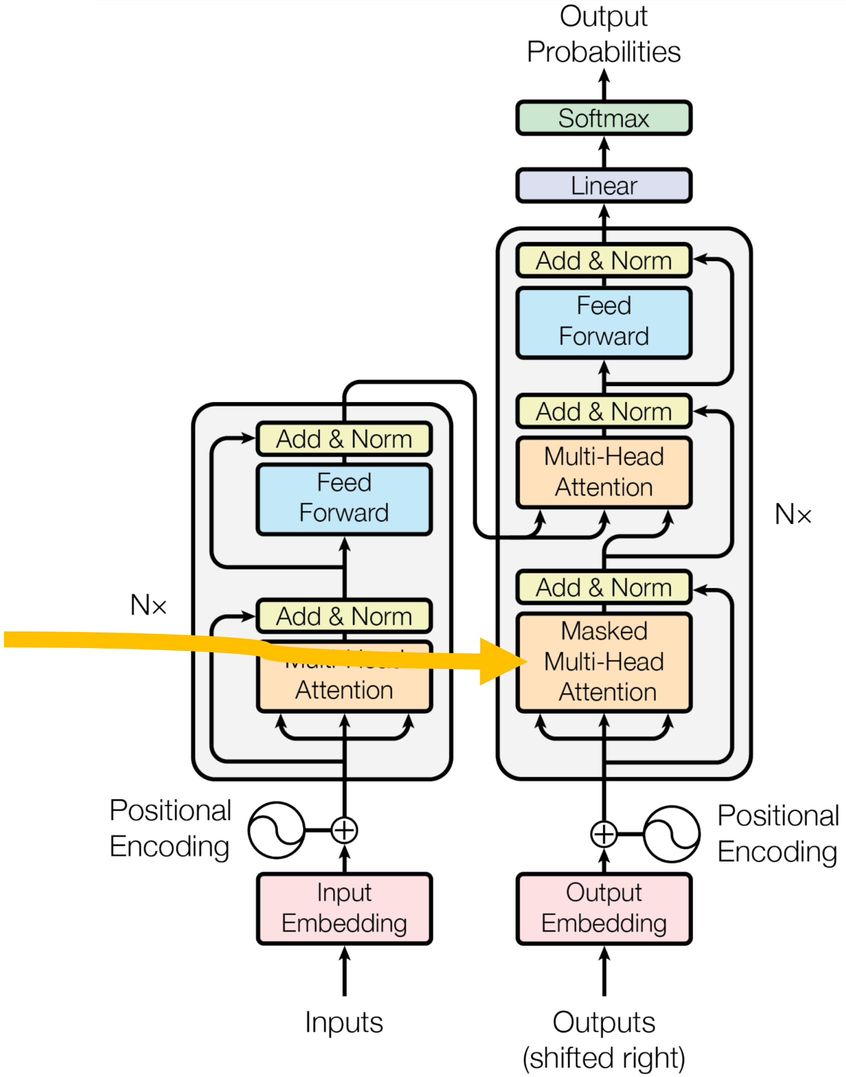
Transformer [Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder** as well as **decoder's** own generations



MaskedDecoder Self-Attention



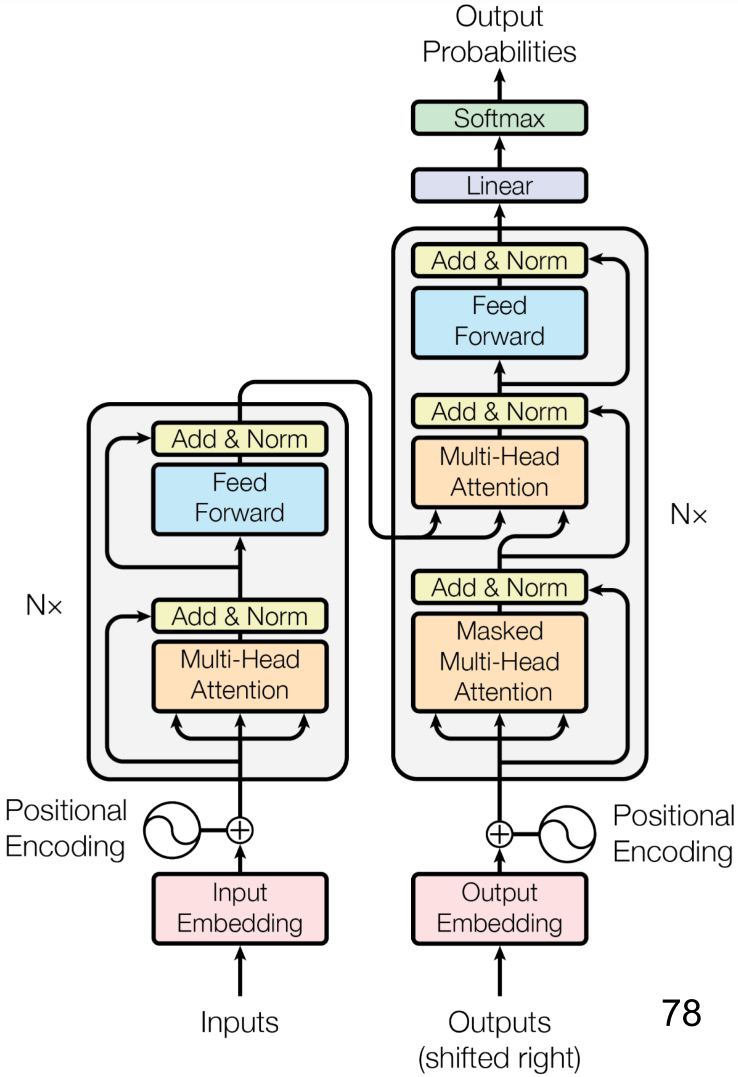


[Attention Is All You Need, Vaswani et al. 2017]

Transformer [Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder** as well as **decoder's** own generations

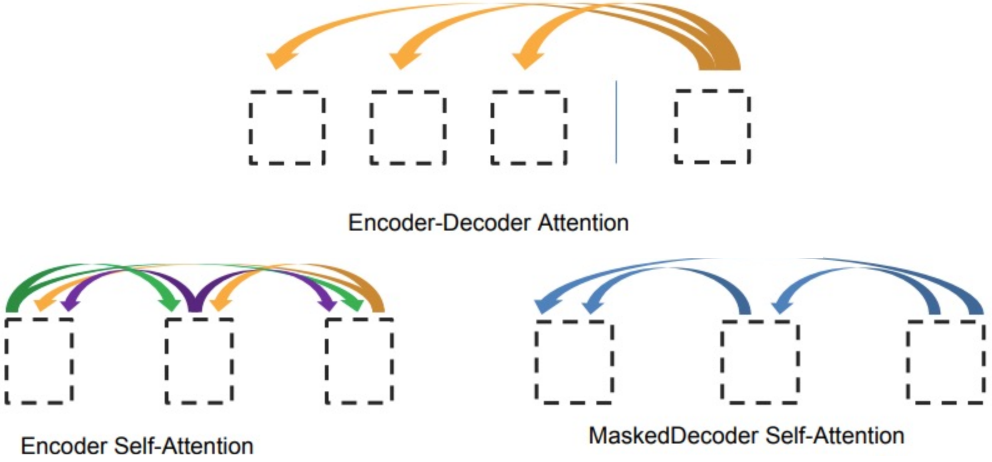
- At any step of decoder, **re-use** previous computation of encoder.
- Computation of decoder is **linear**, instead of quadratic.



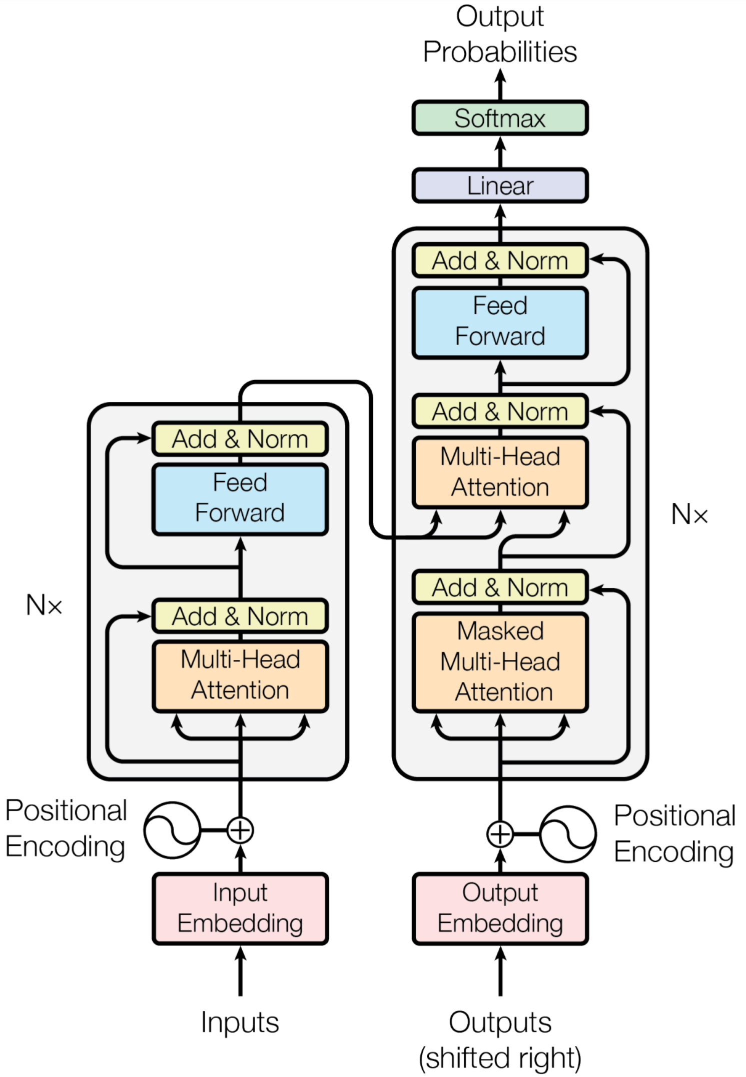
[Attention Is All You Need, Vaswani et al. 2017]

Recap: Transformer

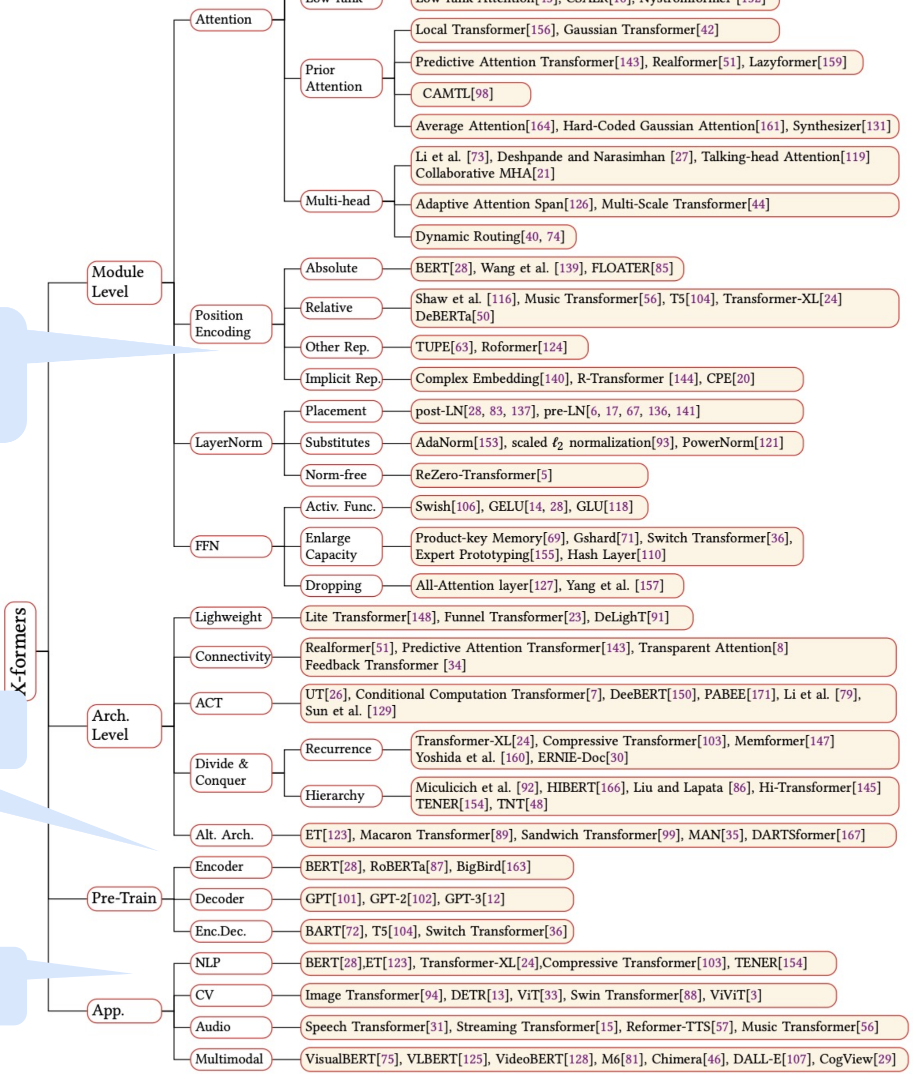
- Yaaay we know Transformers now! 🥳
- An encoder-decoder architecture
- 3 forms of attention



[Attention Is All You Need, Vaswani et al. 2017]



After Transformer …



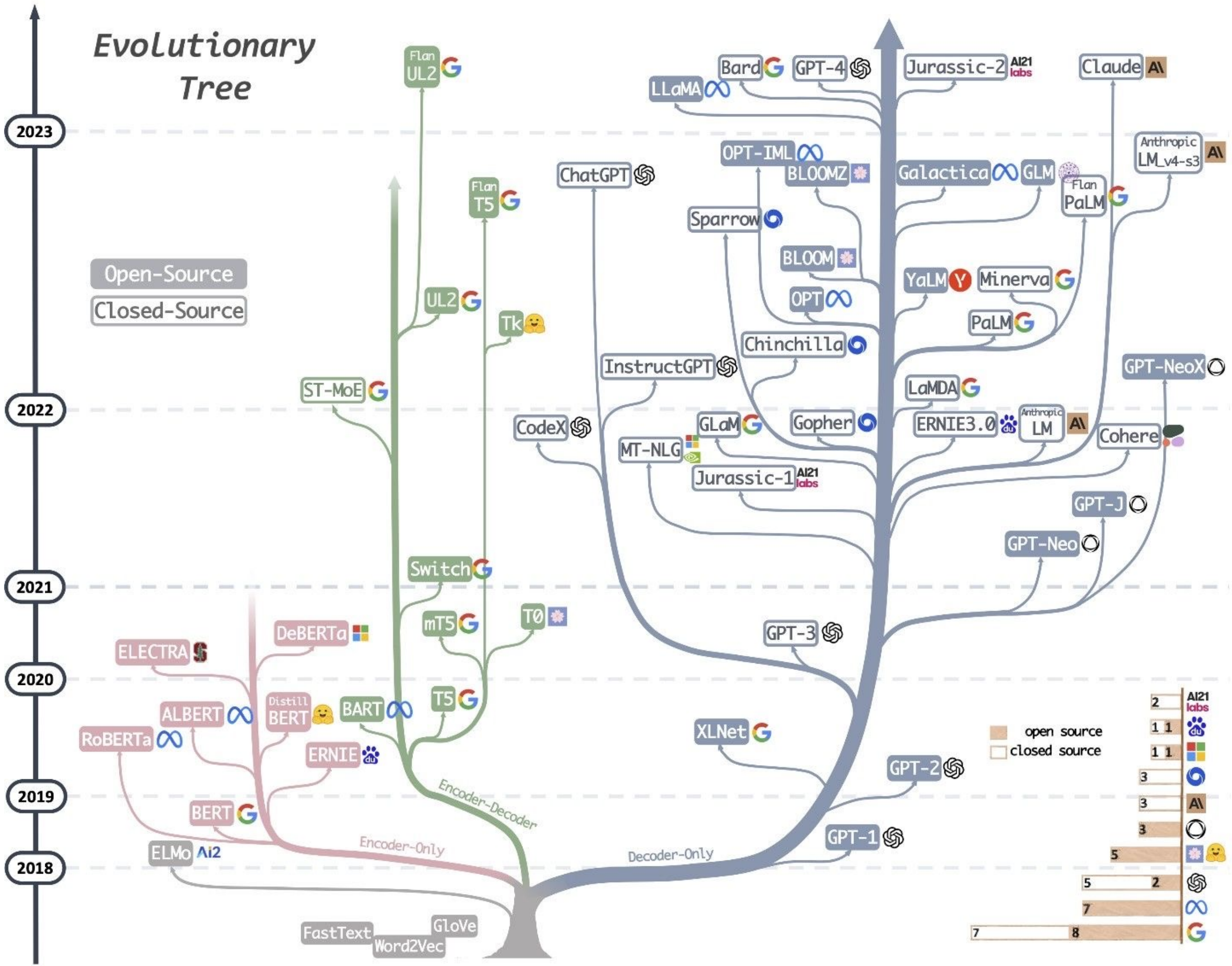
Variants of positional embeddings

Architectural choices

Multi-modal models

We will visit a few of these branches …

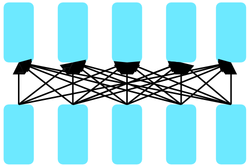
But there is a lot that we do **not** cover …



Yang et al. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, 2023

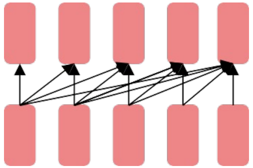
Impact of Transformers

- A building block for a variety of LMs



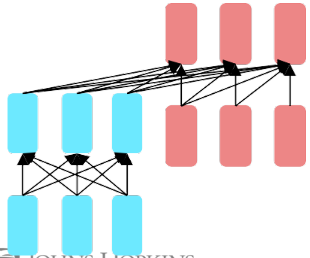
Encoders

- Examples: BERT, RoBERTa, SciBERT.
- Captures bidirectional context. Wait, how do we pretrain them?



Decoders

- Examples: GPT-2, GPT-3, LaMDA
- Other name: causal or auto-regressive language model
- Nice to generate from; can't condition on future words



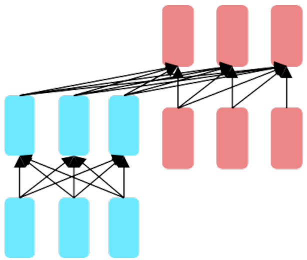
Encoder-Decoders

- Examples: Transformer, T5, Meena
- What's the best way to pretrain them?

85

Transformer
Language Model Families

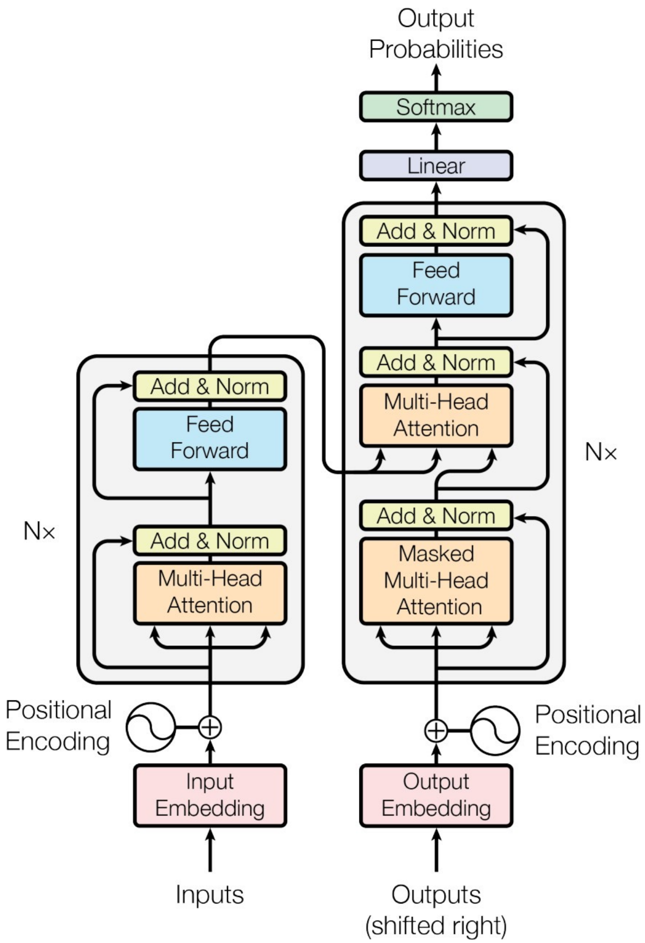
Encoder-Decoder Family of Transformers



Encoder-
Decoders

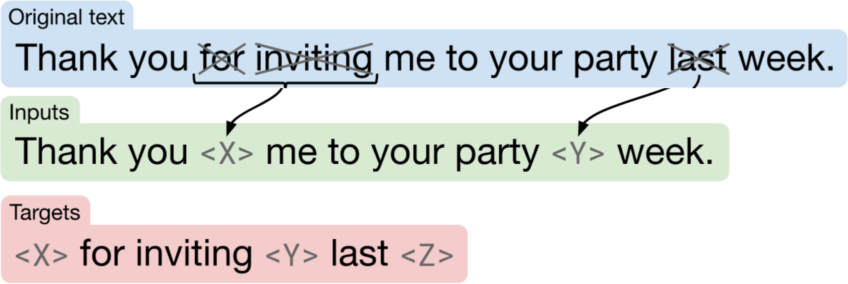
Encoder-decoder Models

- The original transformer architecture was encoder decoder
- Encoder-decoder models are flexible in both generation and classification tasks
- How can we pretrain an encoder-decoder model like BERT to be a good general language pretrained LM?



T5: Text-To-Text Transfer Transformer

- An encoder-decoder architecture
- Pre-training objective:
  corrupt and reconstruct objective



Model	Parameters	No. of layers	d_{model}	d_{ff}	d_{kv}	No. of heads
Small	60M	6	512	2048	64	8
Base	220M	12	768	3072	64	12
Large	770M	24	1024	4096	64	16
3B	3B	24	1024	16384	128	32
11B	11B	24	1024	65536	128	128

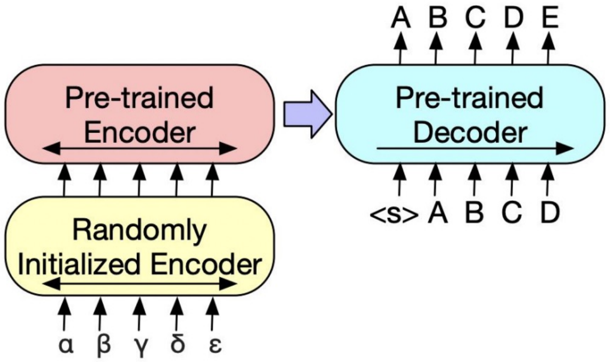
- The original paper is an excellent set of in-depth analysis of various parameters of model design. We discuss some of these results in other places.

`https://huggingface.co/t5-base`

Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

BART (Lewis et al. 2020)

- Similar Architecture as T5.
  - Corrupt the input -> ask the model to reconstruct the original input
  - Outperformed existing methods on generative tasks (question answering, and summarization).



BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

Mike Lewis*, Yinhan Liu*, Naman Goyal*, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer
Facebook AI
{mikelewis,yinhanliu,naman}@fb.com

BART

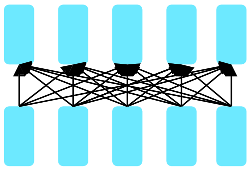
The code might be outdated, but the logic is the same …

```
from transformers import BartTokenizer, BartForConditionalGeneration

tokenizer = BartTokenizer.from_pretrained("facebook/bart-large")
model = BartForConditionalGeneration.from_pretrained("facebook/bart-large")

TXT = "The sun is <mask> ."
input_ids = tokenizer([TXT], return_tensors="pt")["input_ids"]
logits = model(input_ids).logits

masked_index = (input_ids[0] == tokenizer.mask_token_id).nonzero().item()
probs = logits[0, masked_index].softmax(dim=0)
values, predictions = probs.topk(5)

tokenizer.decode(predictions).split()
```

<u>Result:</u>

```
['located', 'at', 'approximately', 'also', 'about']
```

Encoder-only Family of Transformers



BERT

Bidirectional Encoder Representations from Transformers

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

Like Bidirectional LSTMs (ELMo), let's look in **both** directions

BERT

Bidirectional Encoder Representations from Transformers

Let's only use Transformer Encoders, no Decoders

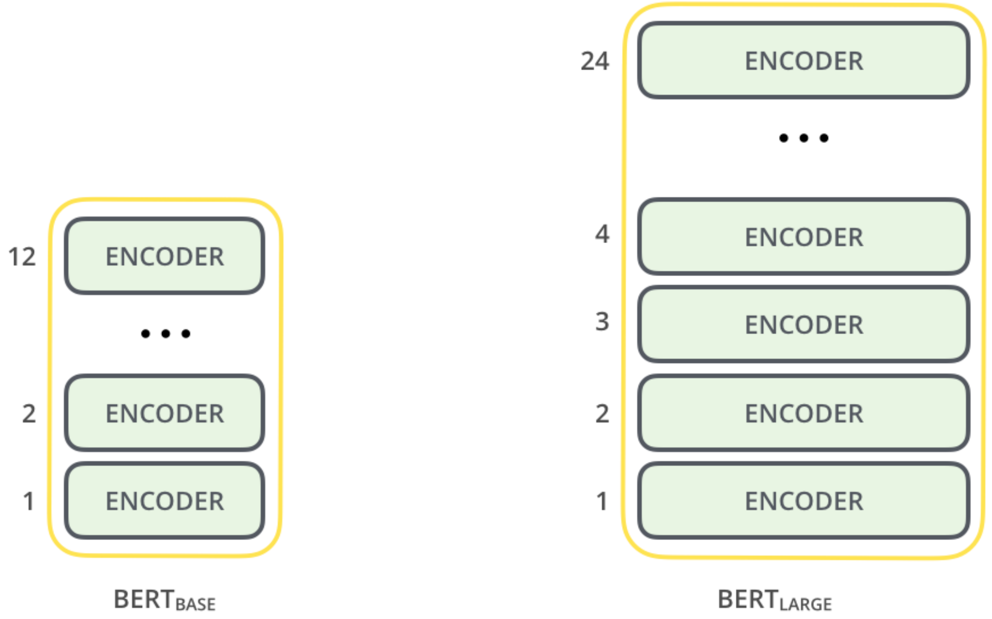
BERT

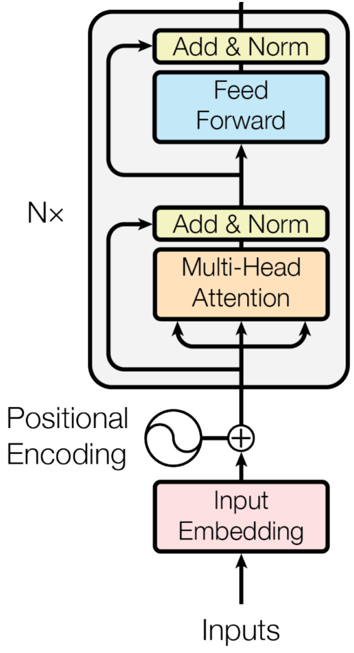
Bidirectional Encoder Representations from Transformers

It's a language model that builds rich representations via self-supervised learning (pre-training)

BERT: Architecture

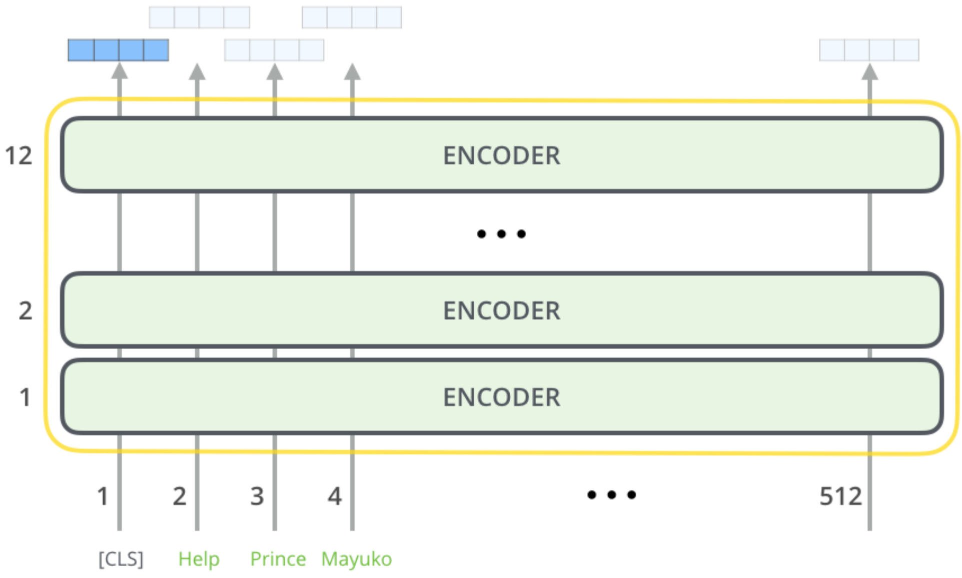
- Stacks of Transformer encoders

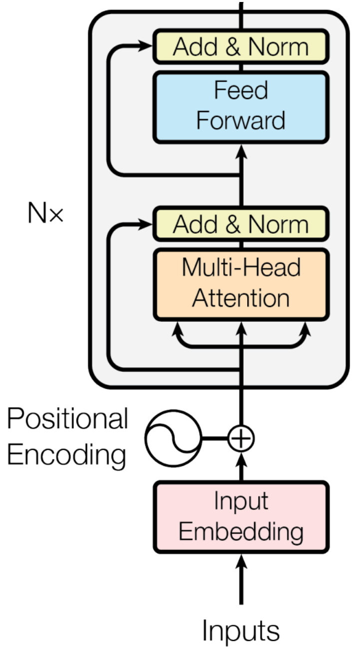




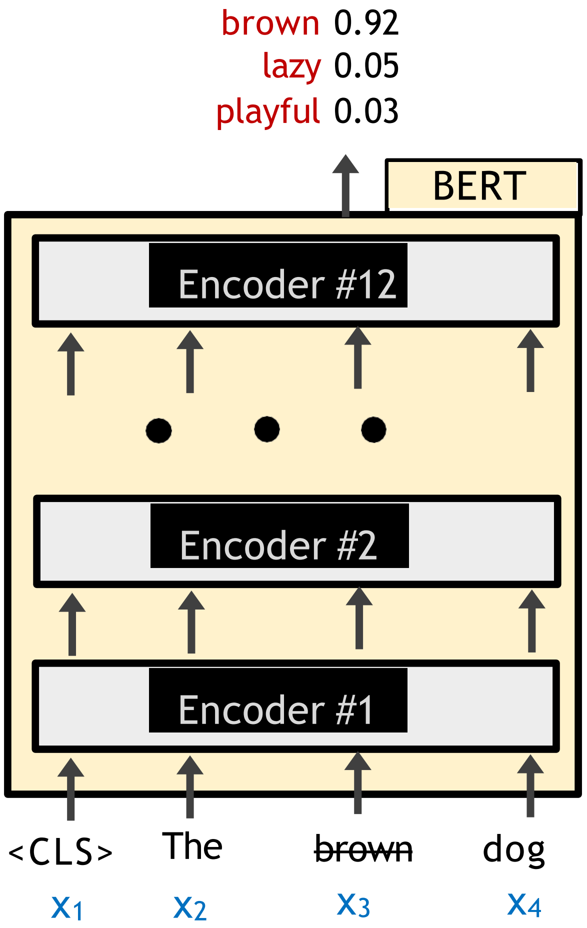
BERT: Architecture

- Model output dimension: 512





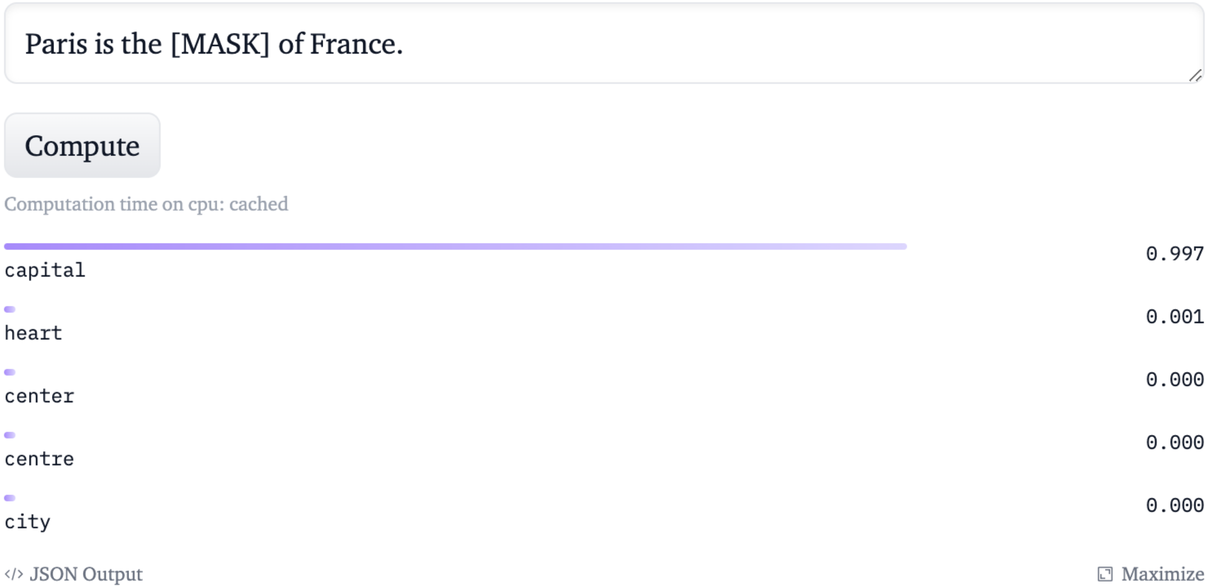
[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]



BERT is trained to uncover masked tokens.

Probing BERT Masked LM

- Masking words forces BERT to use context in both directions to predict the masked word.

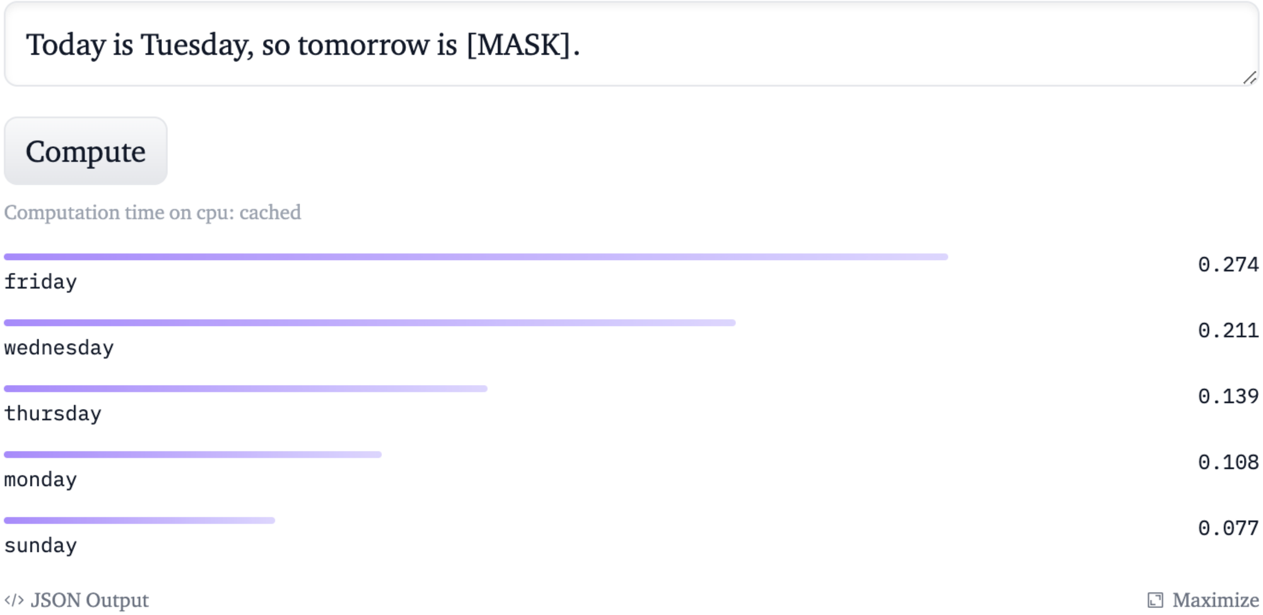


https://huggingface.co/bert-base-uncased

100

Probing BERT Masked LM

- Masking words forces BERT to use context in both directions to predict the masked word.

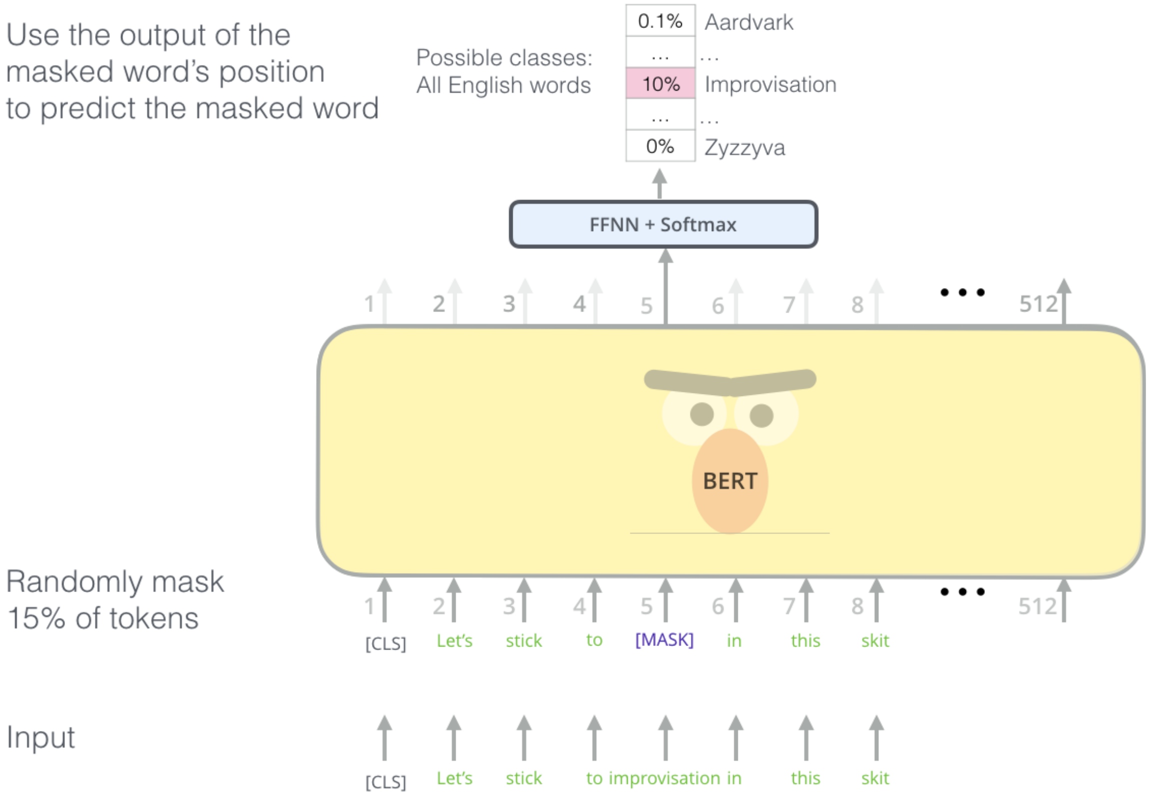


https://huggingface.co/bert-base-uncased

101

BERT: Pre-training Objective (1): Masked Tokens

- Randomly mask 15% of the tokens and train the model to predict them.



[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT: Pre-training Objective (1): Masked Tokens

store

Galon

```
the man went to the [MASK] to buy a [MASK] of milk
```

- Too little masking: Too expensive to train
- Too much masking: Underdefined
  - (not enough info for the model to recover the masked tokens)

Later work shows that more principled masking (instead of uniformly random) could benefit downstream task performance and result in faster training.

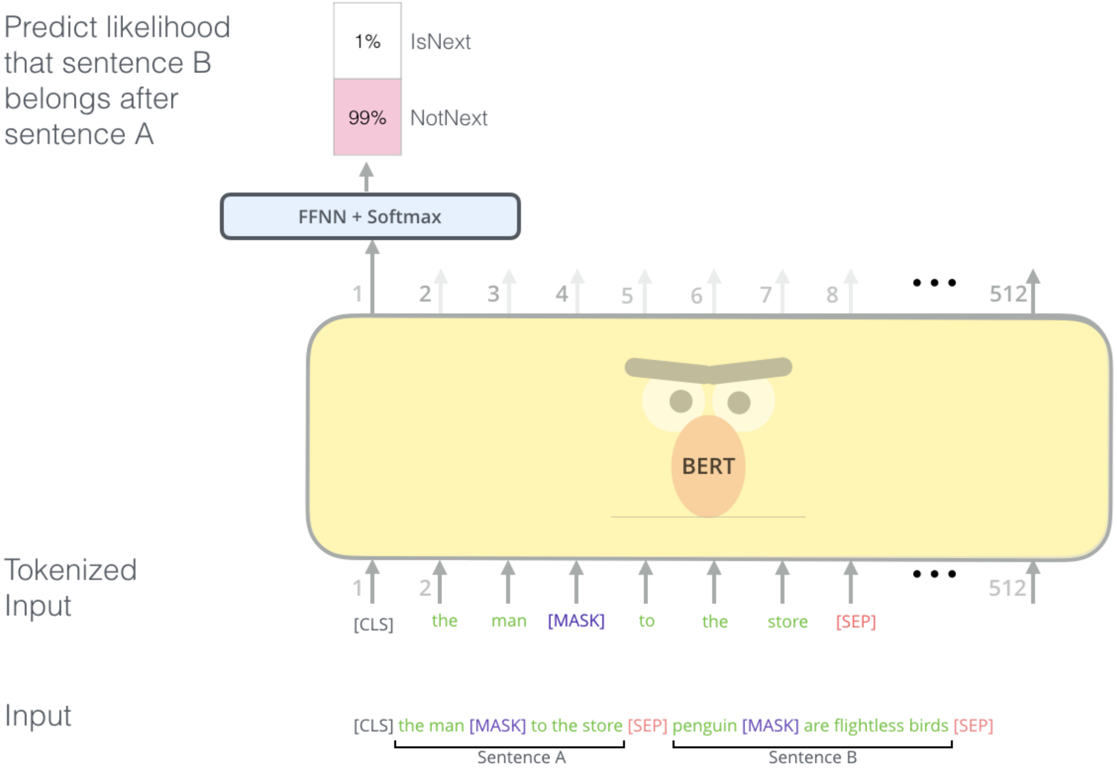
PMI Masking (Levine et al., 2021) https://arxiv.org/pdf/2010.01825.pdf
SpanBERT (Joshi et al., 2020) https://arxiv.org/pdf/1907.10529.pdf

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT: Pre-training Objective (2): Sentence Ordering

- Predict sentence ordering

- 50% correct ordering, and
  50% random incorrect ones



[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

BERT Pre-training Objective (2): Sentence Ordering

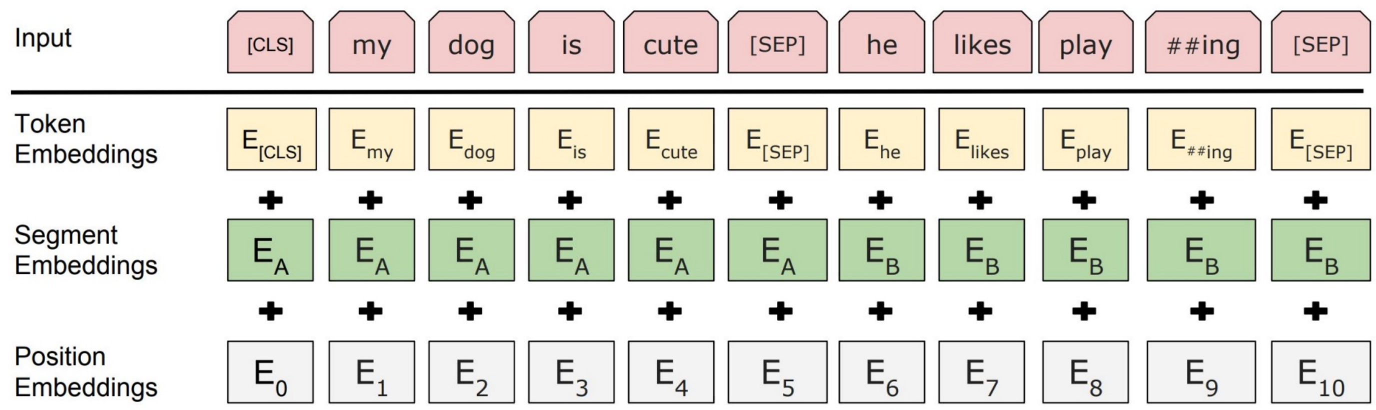
- Learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

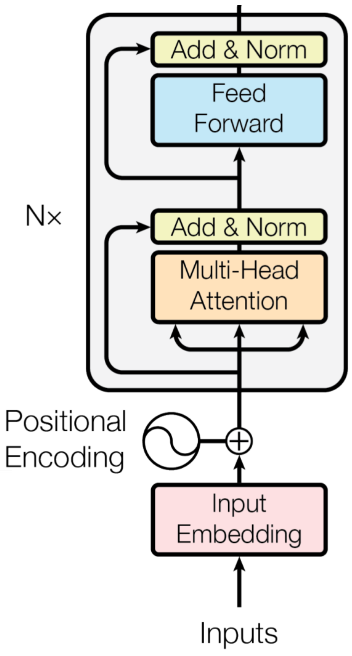
Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

BERT: Input Representation

- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
  - Addition to transformer encoder: sentence embedding





[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

Training

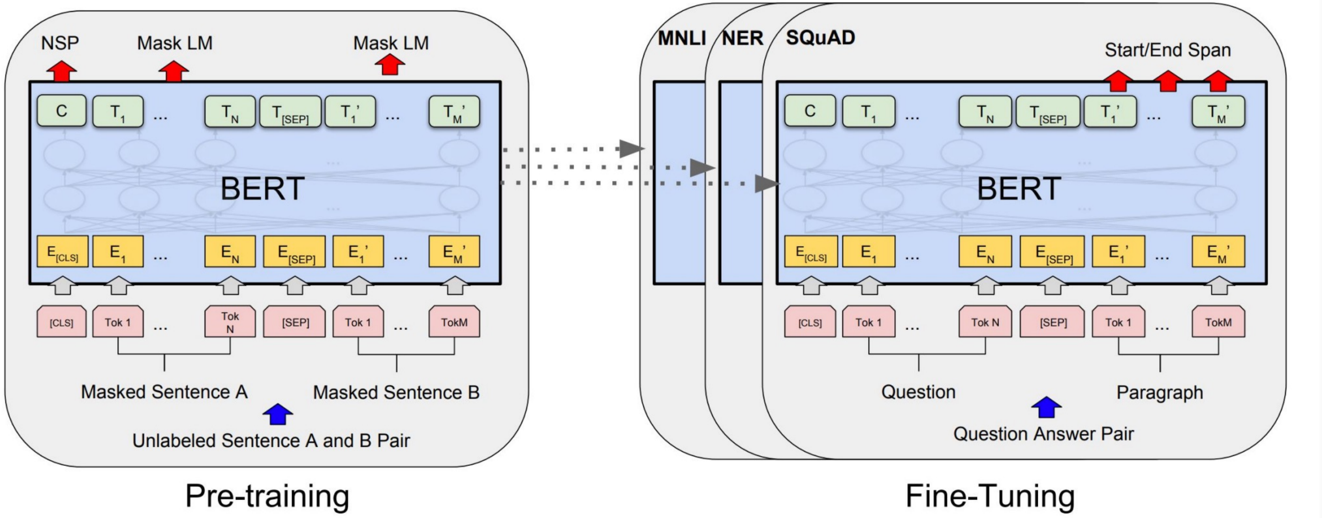
- Trains model on unlabeled data over different pre-training tasks (self-supervised learning)
- **Data:** Wikipedia (2.5B words) + BookCorpus (0.8B words)
- **Training Time:** 1M steps (~40 epochs)
- **Optimizer:** AdamW, 1e-4 learning rate, linear decay
- **BERT-Base:** 12-layer, 768-hidden, 12-head, sequence length of 512
- **BERT-Large:** 24-layer, 1024-hidden, 16-head, sequence length of 512
- Trained on 4x4 and 8x8 TPUs for 4 days (cost today using cloud TPU: $1.3K and $5K)

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

Fine-tuning BERT

"Pretrain once, finetune many times."

- **Idea:** Make pre-trained model **usable** in **downstream tasks**
- Initialized with pre-trained model parameters
- Fine-tune model parameters using labeled data from downstream tasks



[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

An Example Result: SWAG

```
A girl is going across a set of monkey bars.  She
(i)  jumps up across the monkey bars.
(ii)  struggles onto the bars to grab her head.
(iii)  gets to the end and stands on a wooden plank.
(iv)  jumps up and does a back flip.
```

Leaderboard

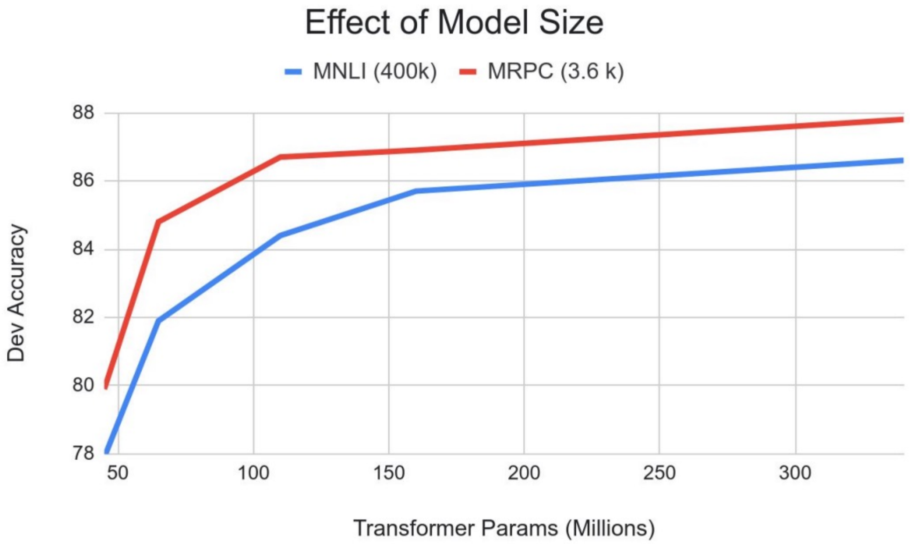
Human Performance (88.00%)
Running Best
Submissions

Rank	Model	Test Score
1	**BERT (Bidirectional Encoder Representations from Transfo...** *Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova* 10/11/2018	**86.28%**
2	**OpenAI Transformer Language Model** *Original work by Alec Radford, Karthik Narasimhan, Tim Salimans, ...* 10/11/2018	**77.97%**
3	**ESIM with ELMo** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/30/2018	**59.06%**
4	**ESIM with Glove** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/29/2018	**52.45%**

- Run each Premise + Ending through BERT.
- Produce logit for each pair on token 0 ([CLS])

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]

Effect of Model Size



- Big models help a lot
- Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples
- Improvements have **not** plateaued!

Impact of BERT

- In order to have state-of-the-art performance on different tasks, there is no need for coming up with a novel model architecture
  - End of task-specific model architecture engineering

- An early sign that larger scales and self-supervised learning (language modeling) are the key for future performance improvements

Why did no one think of this before?

- Why wasn't contextual pre-training popular before 2018 with ELMo?

- Good results on pre-training is >1,000x to 100,000 more expensive than supervised training.

What Happened After BERT?

- RoBERTa (Liu et al., 2019)
  - Exact same architecture as BERT
  - Drops the next sentence prediction loss!
  - Trained on 10x data (the original BERT was actually under-trained)
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)

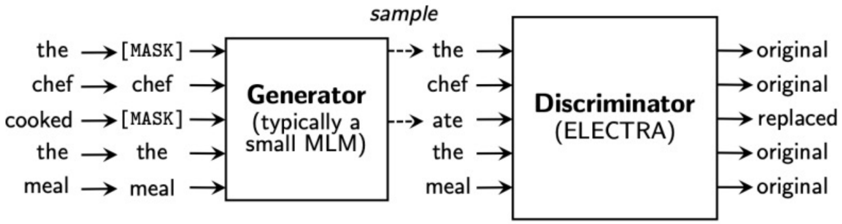
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$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

What Happened After BERT?

- RoBERTa (Liu et al., 2019)
  - Exact same architecture as BERT
  - Drops the next sentence prediction loss!
  - Trained on 10x data (the original BERT was actually under-trained)
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)

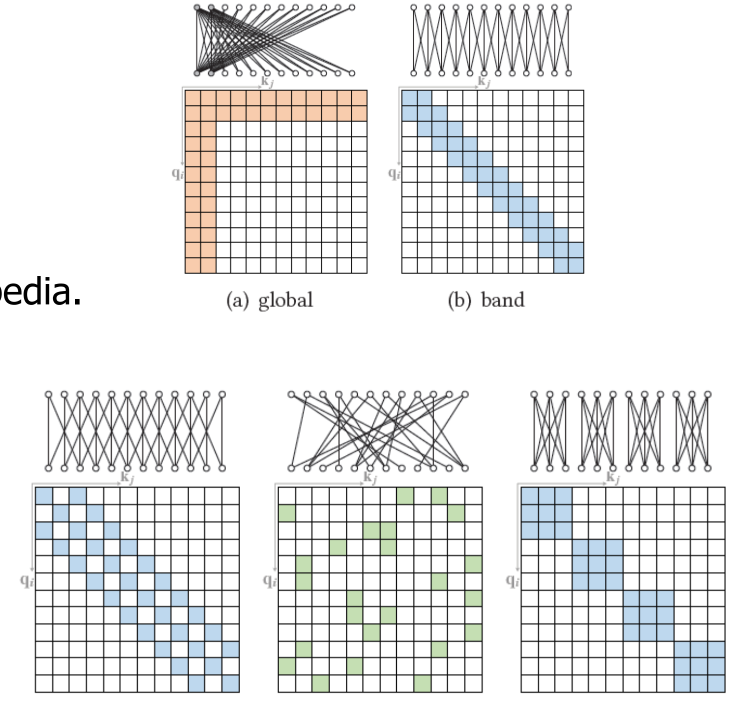
- ALBERT (Lan et al., 2020)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower..

- ELECTRA (Clark et al., 2020)
  - Pre-training objective: replaced-token detection
  - Two models generator and discriminator (GAN-like)
  - It provides a more efficient training method



What Happened After BERT?

- Models that handle long contexts
  - Longformer, Big Bird, …
- Multilingual BERT
  - Trained single model on 104 languages from Wikipedia.
- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT, …
- Making BERT smaller to use
  - DistillBERT, TinyBERT, …



(a) global (b) band (c) dilated (d) random (e) block local

Text generation using BERT

- Does not support generation or sequence-to-sequence tasks
  - Summarization, Translation, Text simplification, etc

**BERT has a Mouth, and It Must Speak:
BERT as a Markov Random Field Language Model**

Alex Wang
New York University
alexwang@nyu.edu

Kyunghyun Cho
New York University
Facebook AI Research
CIFAR Azrieli Global Scholar
kyunghyun.cho@nyu.edu

**Mask-Predict: Parallel Decoding of
Conditional Masked Language Models**

Marjan Ghazvininejad* Omer Levy* Yinhan Liu* Luke Zettlemoyer
Facebook AI Research
Seattle, WA

Exposing the Implicit Energy Networks behind Masked Language Models via Metropolis--Hastings

Kartik Goyal, Chris Dyer, Taylor Berg-Kirkpatrick

src	Der Abzug der franzsischen Kampftruppen wurde am 20. November abgeschlossen .
$t = 0$	The departure of the French combat completed completed on 20 November .
$t = 1$	The departure of French combat troops was completed on 20 November .
$t = 2$	The withdrawal of French combat troops was completed on November 20th .

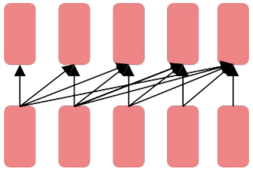
Leveraging Pre-trained Checkpoints for Sequence Generation Tasks

Sascha Rothe, Shashi Narayan, Aliaksei Severyn

Summary Thus Far

- BERT and the family
- An encoder; Transformer-based networks trained on massive piles of data.
- Incredible for learning contextualized embeddings of words
- It's very useful to pre-train a large unsupervised/self-supervised LM then fine-tune on your particular task (replace the top layer, so that it can work)
- However, they were not designed to generate text.

Decoder-only Family of Transformers



Decoders

GPT

Generative Pre-trained Transformer

GPT-2: A Big Language Model (2019)

Language Models are Unsupervised Multitask Learners

Alec Radford * [1] Jeffrey Wu * [1] Rewon Child [1] David Luan [1] Dario Amodei ** [1] Ilya Sutskever ** [1]

GPT: An Auto-Regressive LM (2018)

Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

GPT-2

- GPT-2 uses only Transformer Decoders (no Encoders) to generate new sequences from scratch or from a starting sequence
- As it processes each subword, it masks the "future" words and conditions on and attends to the previous words

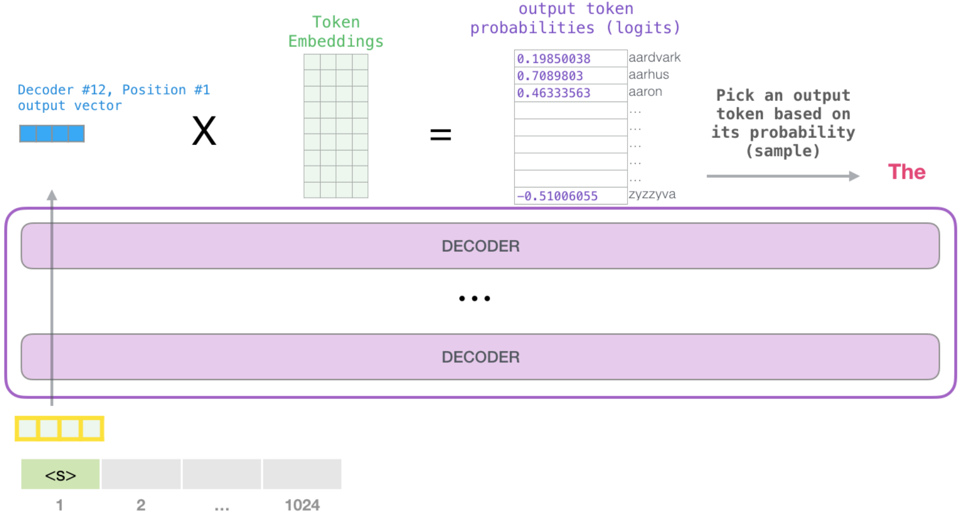
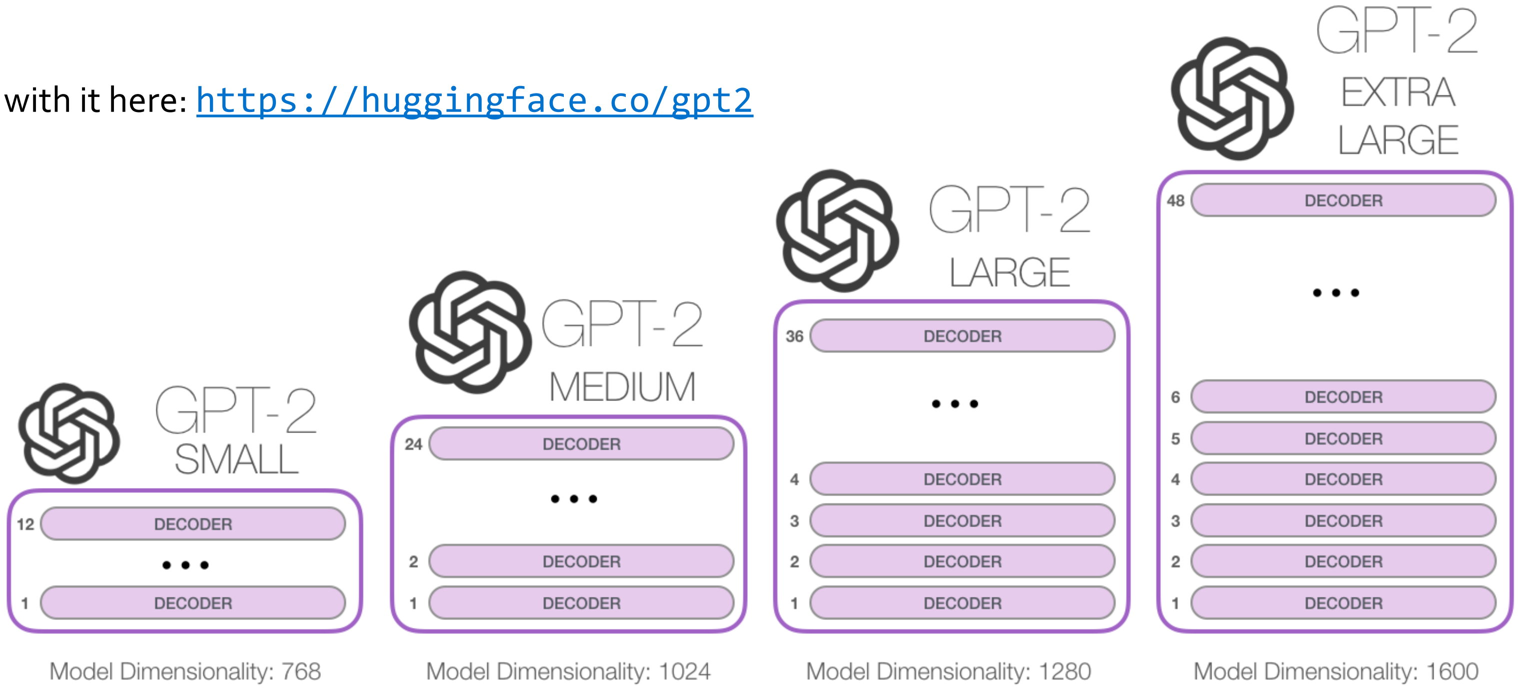


Image by http://jalammar.github.io/illustrated-gpt2/

GPT2: Model Sizes

Play with it here: https://huggingface.co/gpt2



117M parameters 345M 762M 1542M

[Image by http://jalammar.github.io/illustrated-gpt2/]

GPT-2 is identical to GPT-1, but:

- Has Layer normalization in between each sub-block (as we've already seen)
- Vocab extended to 50,257 tokens and context size increased from 512 to 1024
- Data: 8 million docs from the web (Common Crawl), minus Wikipedia

Language Models are Unsupervised Multitask Learners

Alec Radford *[1] **Jeffrey Wu** *[1] **Rewon Child**[1] **David Luan**[1] **Dario Amodei** **[1] **Ilya Sutskever** **[1]

GPT2: Some Results

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	56.25	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). Other language model results are from (Dai et al., 2019).

Article: Prehistoric man sketched an incredible array of prehistoric beasts on the rough limestone walls of a cave in modern day France 36,000 years ago. Now, with the help of cutting-edge technology, those works of art in the Chauvet-Pont-d'Arc Cave have been reproduced to create the biggest replica cave in the world. The manmade cavern named the Caverne du Pont-d'Arc has been built a few miles from the original site in Vallon-Pont-D'arc in Southern France and contains 1,000 painstakingly-reproduced drawings as well as around 450 bones and other features... Cavemen and women sketched an incredible array of prehistoric beasts on the rough limestone walls of a cave 36,000 years ago and now a replica has been created (pictured) ...	
GPT-2: The original site in Vallon-Pont-D'arc in Southern France is a Unesco World Heritage site and is the oldest known and the best preserved cave decorated by man. The replica cave was built a few miles from the original site in Vallon-Pont-D'Arc in Southern France. The cave contains images of 14 different species of animals including woolly rhinoceros, mammoths, and big cats.	**Reference:** Cave mimics famous Caverne du Pont-d'Arc in France, the oldest cave decorated by man and the best preserved. The replica contains all 1,000 paintings which include 425 such as a woolly rhinoceros and mammoths. Minute details were copied using 3D modelling and anamorphic techniques, often used to shoot widescreen images. The modern cave also includes replica paw prints of bears, bones and details preserved in the original cave.

Impact of GPT2

- Zero-shot learning (no use of task-specific supervision) increasingly become a reality.

NMT: “Translate to french,” <English text>, <French text>.

QA: “Answer the question,” <Document>, <Question>, <Answer>.

SUMM: <Document> “TL; DR:” <Summarization>

GPT-3: A Very Large Language Model (2020)

- More layers & parameters
- Bigger dataset
- Longer training
- Larger embedding/hidden dimension
- Larger context window

[Slide credit: Sbhya Chhabria & Michael Tang]

Size Comparisons

- BERT-Base model has 12 transformer blocks, 12 attention heads,
  - 110M parameters!
- BERT-Large model has 24 transformer blocks, 16 attention heads,
  - 340M parameters!
- GPT-2 is trained on 40GB of text data (8M webpages)!
  - 1.5B parameters!
- GPT-3 is an even bigger version of GPT-2, but isn't open-source
  - 175B parameters!

Impact of GPT3

- Moving away from the fine-tuning paradigm
  - Zero/Few-shot learning and in-context learning
- Massive LM scale makes high zero/few-shot performance possible
- Start of closed source models
  - Not too many details about their model
  - No released code / model checkpoint
- Also revitalized ppen source efforts:
  - OPT, LLaMA by Meta, BLOOM by Huggingface, etc.

GPT4

Model	Usage
davinci-002	$0.0020 / 1K tokens

Model	Input	Output
gpt-4	$0.03 / 1K tokens	$0.06 / 1K tokens

- Transformer-based
  - The rest is …. mystery! ☺
  - If we're going based on costs, GPT4 is ~15-30 times costlier than GPT3. That should give you an idea how its likely size!

- Note, these language models involve more than just pre-training.
  - Pre-training provides the foundation based on which we build the model.
  - We will discuss the later stages (post hoc alignment) in a 2-3 weeks.

Accessing API Models

```python
import openai
openai.api_key = ("sk-[redacted]")
my_prompt = '''The sun is [MASK].

    Replace [MASK] with the most probable 5 words to replace, and give me their probabilities.'''
# Here set parameters as you like
response = openai.Completion.create(
  engine="text-davinci-002",
  prompt=my_prompt,
  temperature=0,
  max_tokens=100,
)

print(response['choices'][0]['text'])
```

Other Available [Decoder] LMs

EleutherAI: GPT-Neo (6.7B), GPT-J (6B), GPT-NeoX (20B)

https://huggingface.co/EleutherAI

https://6b.eleuther.ai/

LLaMA, 65B: https://github.com/facebookresearch/llama

Mistral and Mixtral:

https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

Training Transformer LMs: Empirical Considerations

Pre-training Transformer LMs

- You have learned about the basics of pre-training Transformer language models.
- There is so much empirical knowledge/experiences that goes into training these models.
- Various empirical issues about:
  - Preparation/pre-processing data
  - Efficient training of models
  - …

C4: The Data

- C4: Colossal Clean Crawled Corpus
  - Web-extracted text
  - English language only
  - 750GB

	Data set	Size
★	C4	745GB
	C4, unfiltered	6.1TB

Play with the data: https://c4-search.apps.allenai.org/

C4: The Data

Remove any:
- References to Javascript
- "Lorem ipsum" text — placeholder text commonly used to demonstrate the visual form of a document

Menu

Lemon

Introduction

The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.
The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Article

The origin of the lem … wn, though

Please enable JavaScript to use our site.

Home
Products
Shipping
Contact
FAQ

Dried Lemons, $3.59/pound

Organic dried lemons from our farm in California.
Lemons are harvested and sun-dried for maximum flavor.
Good in soups and on popcorn.

The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.
The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Curabitur in tempus quam. In mollis et ante at consectetur.
Aliquam erat volutpat.
Donec at lacinia est.
Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit.
Fusce quis blandit lectus.
Mauris at mauris a turpis tristique lacinia at nec ante.
Aenean in scelerisque tellus, a efficitur ipsum.
Integer justo enim, ornare vitae sem non, mollis fermentum lectus.
Mauris ultrices nisl at libero porta sodales in ac orci.

```
function Ball(r) {
  this.radius = r;
  this.area = pi * r ** 2;
  this.show = function(){
    drawCircle(r);
  }
}
```

Retain:
- Sentences with terminal punctuation marks
- Pages with at least 5 sentences, sentences with at least 3 words

Pre-training Data: Experiment

- Takeaway:
  - Clean and compact data is better than large, but noisy data.
  - Pre-training on in-domain data helps.

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21

Pre-training Data Duplicates

- There is a non-negligible number of duplicates in any pre-training data.

	% train examples with		% valid with
	dup in train	dup in valid	dup in train
C4	3.04%	1.59%	4.60%
RealNews	13.63%	1.25%	14.35%
LM1B	4.86%	0.07%	4.92%
Wiki40B	0.39%	0.26%	0.72%

Dataset	Example	Near-Duplicate Example
Wiki-40B	\n_START_ARTICLE_\nHum Award for Most Impactful Character \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]	\n_START_ARTICLE_\nHum Award for Best Actor in a Negative Role \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]
LM1B	I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .	I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .
C4	Affordable and convenient holiday flights take off from your departure country, "Canada". From May 2019 to October 2019, Condor flights to your dream destination will be roughly 6 a week! Book your Halifax (YHZ) - Basel (BSL) flight now, and look forward to your "Switzerland" destination!	Affordable and convenient holiday flights take off from your departure country, "USA". From April 2019 to October 2019, Condor flights to your dream destination will be roughly 7 a week! Book your Maui Kahului (OGG) - Dubrovnik (DBV) flight now, and look forward to your "Croatia" destination!

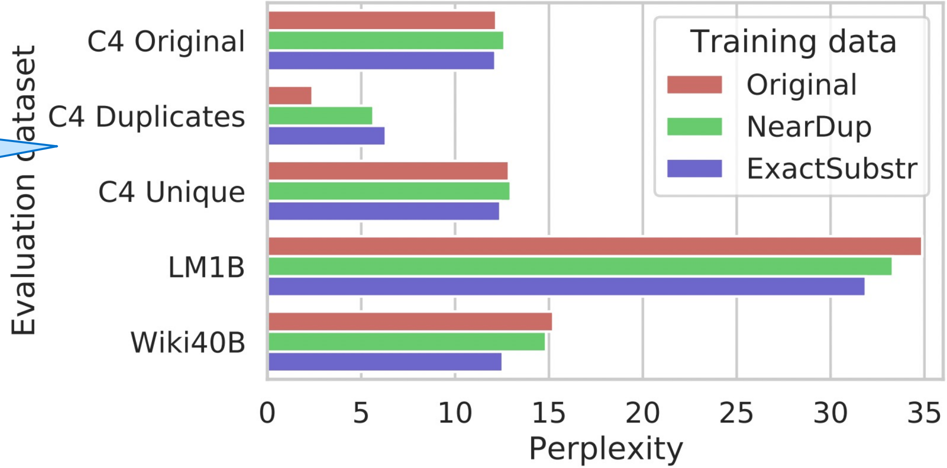
Deduplicating Training Data Makes Language Models Better, 2020

Deduplicating Data Improves LMs

- Models: GPT-2-like (1.5B param) models
- On there datasets:
  - C4 : the original training data
  - C4-NearDup: C4 excluding exact duplicates
  - C4-ExactSubs: C4 excluding near-duplicates

Except when evaluated on duplicate evaluation data!

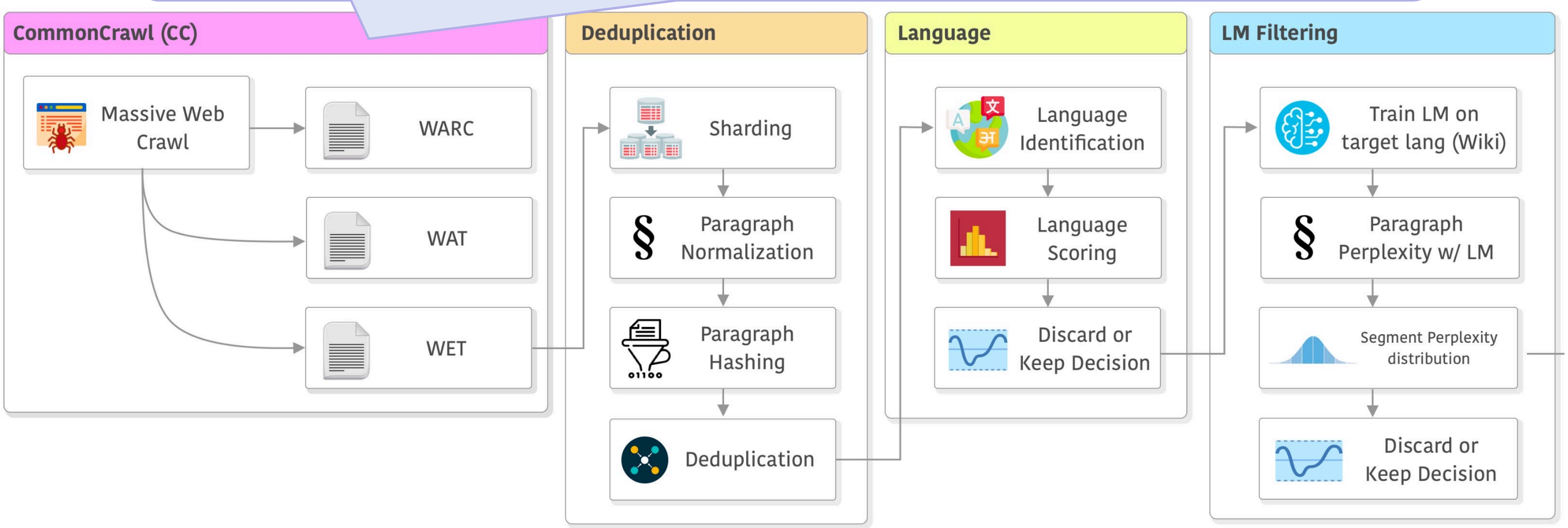
Training on deduplicated data always leads to lower PPL!



Deduplicating Training Data Makes Language Models Better, 2020

LLaMA’s Data Pipeline

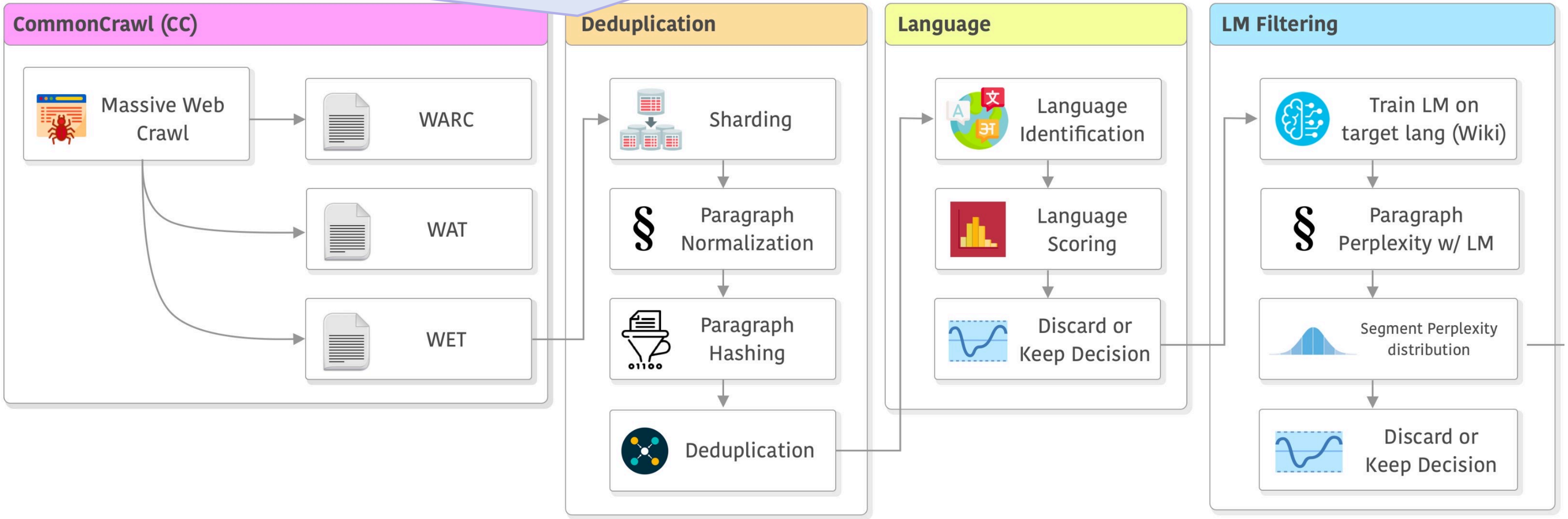
Starts with the massive crawled data by CommonCrawl.
The WET format that contains textual information.
WARC is raw, WAT is metadata, WET is text+some metadata.



https://twitter.com/tarantulae/status/1650170087708454913?t=ncWWY0FI0tYC_dYLs76r5g

LLaMA’s Data Pipeline

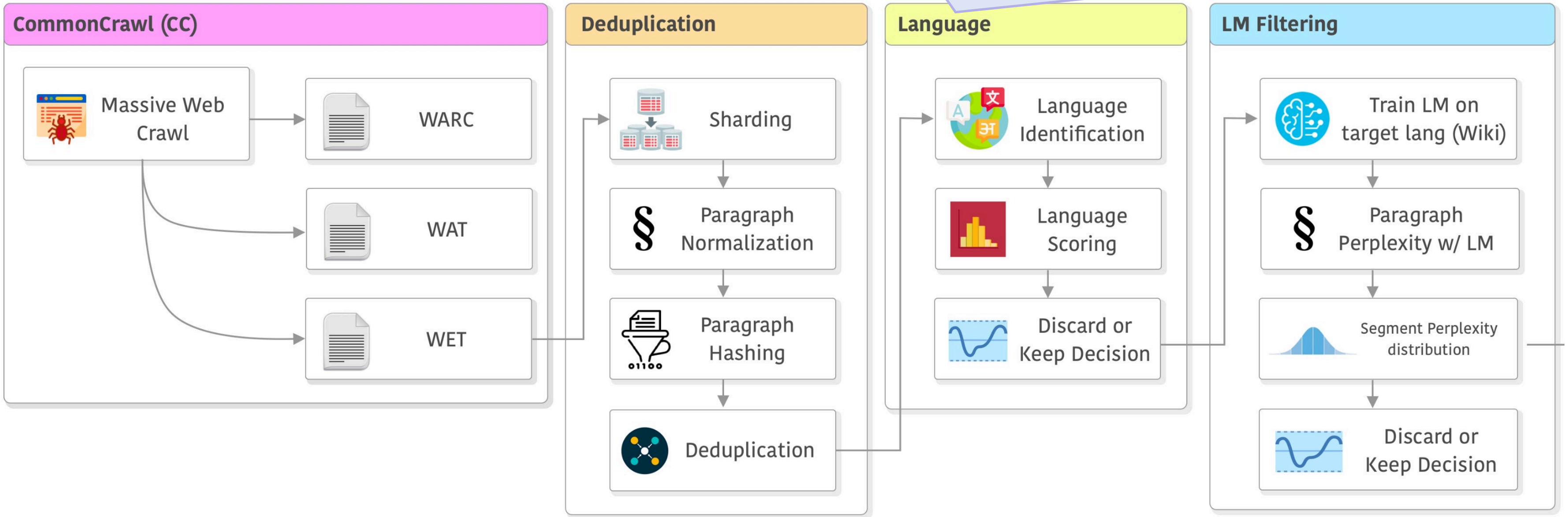
Shard WET content into shards of 5GB each (one CC snapshot can have 30TB). Then you normalize paragraphs (lowercasing, numbers as placeholders, etc), compute per-paragraph hashes and then duplicate them.



https://twitter.com/tarantulae/status/1650170087708454913?t=ncWWY0FI0tYC_dYLs76r5g

LLaMA's Data Pipeline

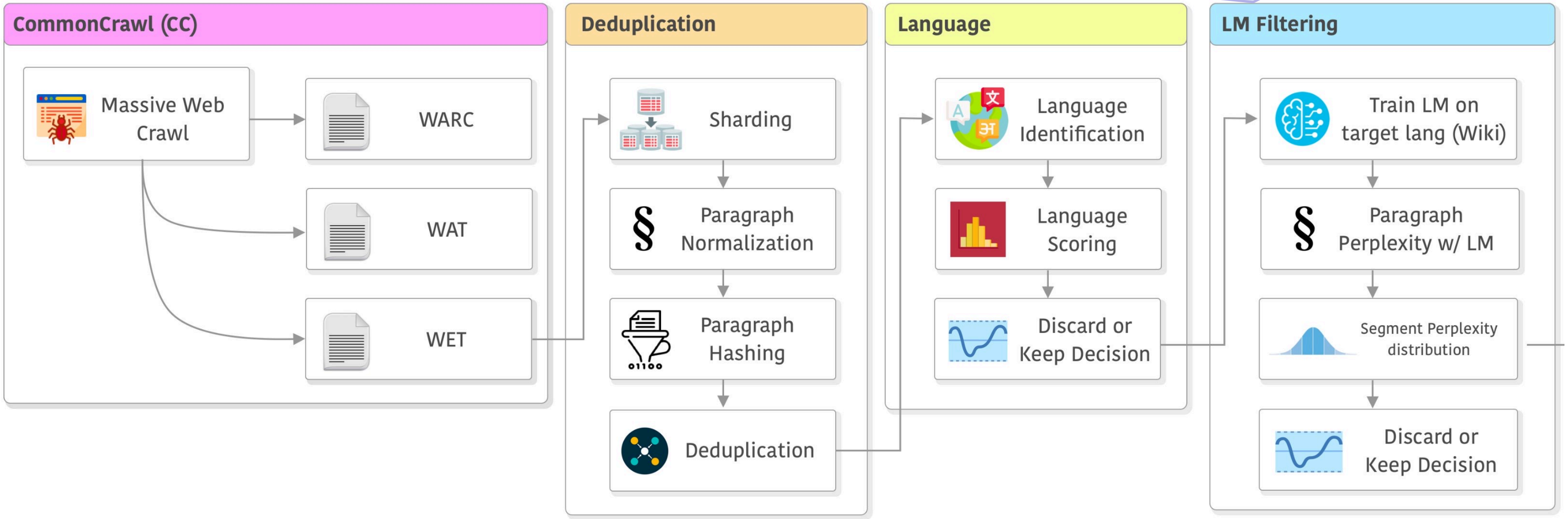
Perform language identification and decide whether to keep or discard languages.
The order of when you do this in the pipeline can impact the language discrimination quality.



LLaMA's Data Pipeline

Do further quality filtering: Train a simple LM (n-gram) on target languages using Wikipedia, then compute per-paragraph perplexity on the rest of the data:

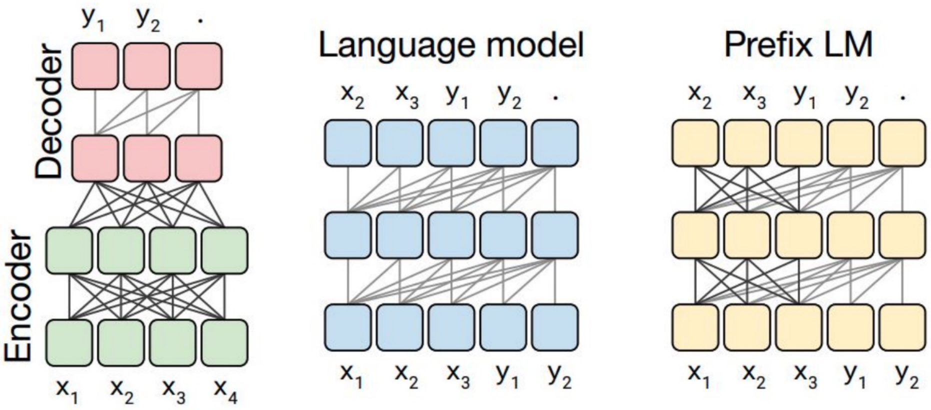
- Very high PPL: Very different than Wiki and likely low-quality → Drop
- Very low PPL: Very similar or near duplicates to Wiki → Drop



CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data, 2019

Architectural choices

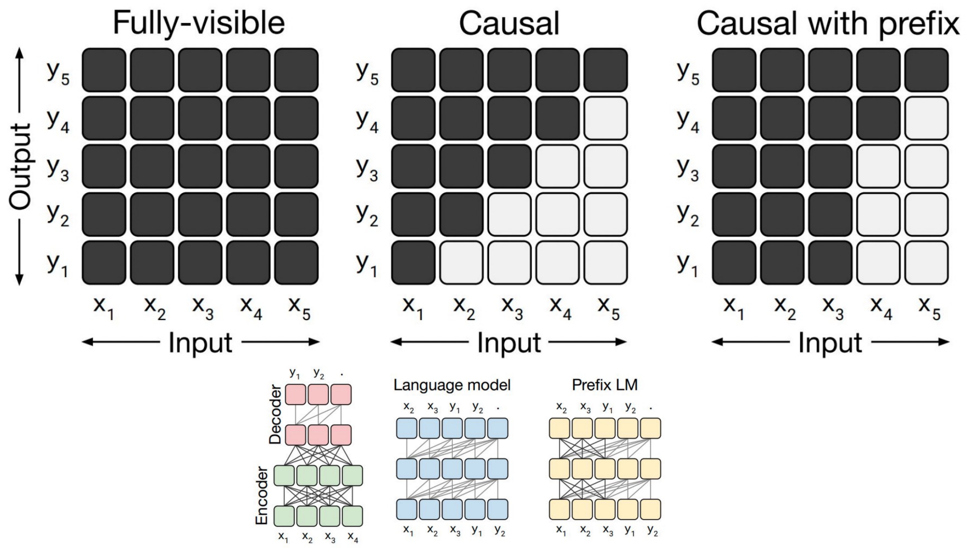
Architectures: Different Choices



Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectures: Different Attention Masks

- **Fully visible** mask allows the self attention mechanism to attend to the full input.
- A **causal mask** doesn't allow output elements to look into the future.
- **Causal mask** with prefix allows to fully-visible masking on a portion of input.

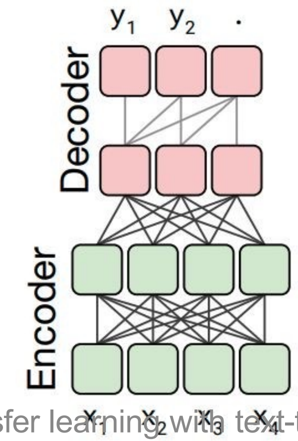


Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**



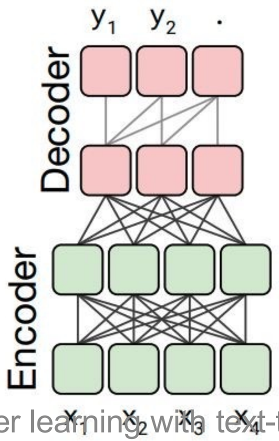
Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**

Input: Thank you for <X> me to your party <Y>. Target: <X> inviting <Y> last week.

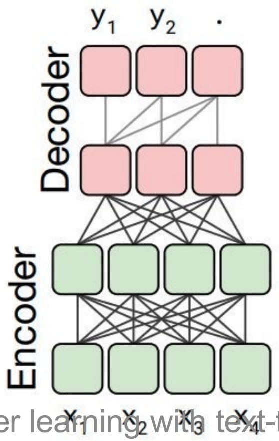


Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**

Number of parameters



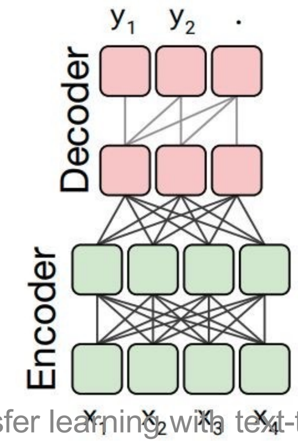
Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**

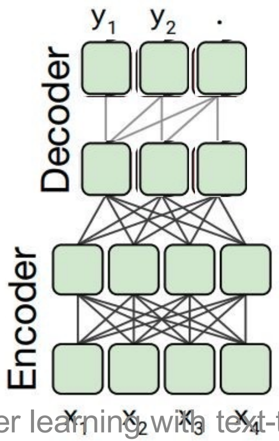
Number of FLOPS



Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**

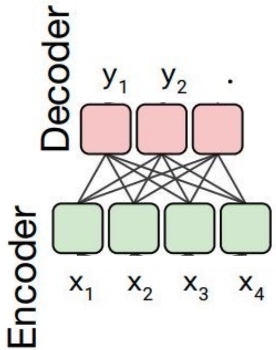


Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95



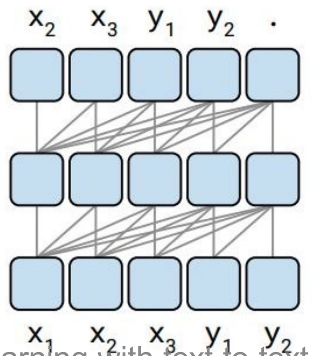
Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

	Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★	Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
	Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
	Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
	Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86

Language model



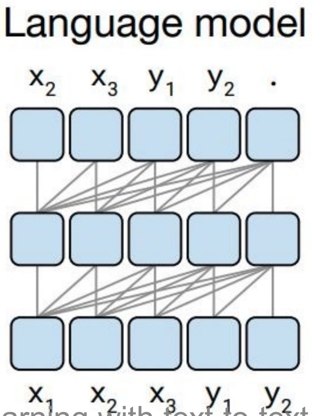
Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86

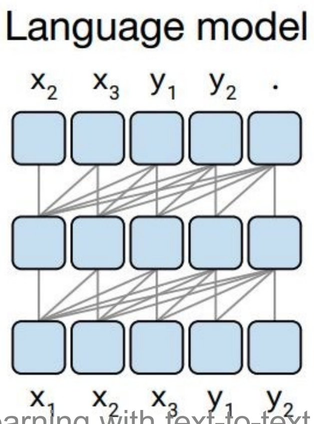
Language model is decoder-only



Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86



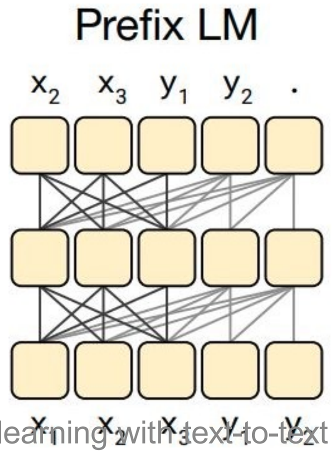
LM looks at both input and target, while encoder only looks at input sequence and decoder looks at output sequence.

Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39



Exploring the limits of transfer learning with text-to-text transfer transformers, 2020

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39

- Takeaways:
  1. Halving the number of layers in encoder and decoder hurts the performance.

Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39

- Takeaways:
  1. Halving the number of layers in encoder and decoder hurts the performance.
  2. Performance of Enc-Dec with shared params is almost on-par with prefix LM.

Pre-training objectives

On Pre-training Objectives

- So far, the dominant objective we have seen is “next-token” prediction.
- In reality any “marginal” observations about language can be a source of supervision.

Objectives

- Prefix language modeling
  - **Input:** `Thank you for inviting`
  - **Output:** `me to your party last week`
- BERT-style denoising
  - **Input:** `Thank you <M> <M> me to your party apple week`
  - **Output:** `Thank you for inviting me to your party last week`
- Deshuffling
  - **Input:** `party me for your to. last fun you inviting week Thanks.`
  - **Output:** `Thank you for inviting me to your party last week`
- IID noise, replace spans
  - **Input:** `Thank you <X> me to your party <X> week`
  - **Output:** `<X> for inviting <Y> last <Z>`
- IID noise, drop tokens
  - **Input:** `Thank you me to your party week .`
  - **Output:** `for inviting last`

Objectives: Experiments

- All the variants perform similarly
- "Replace corrupted spans" and "Drop corrupted tokens" are more appealing because target sequences are shorter, speeding up training.

Assuming Enc-Dec architecture.
Evaluated for classification tasks.

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Grouped Query-Attention

- Used for training LLaMA 2.
- One key-value vector for each group of queries — an interpolation between "multi-head" attention and "multi-query" attention.

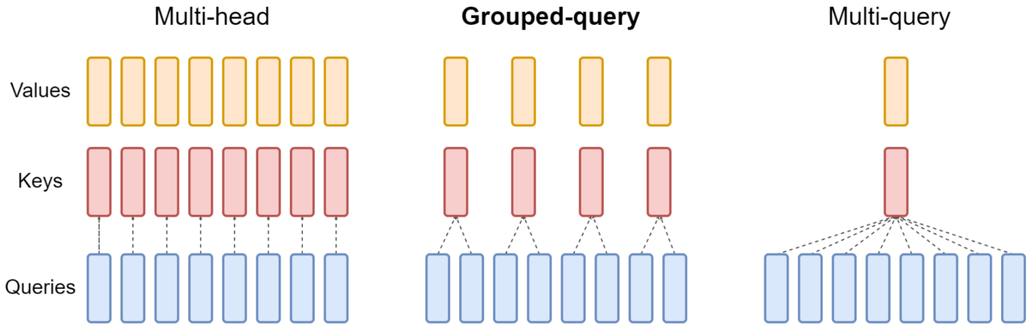


Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

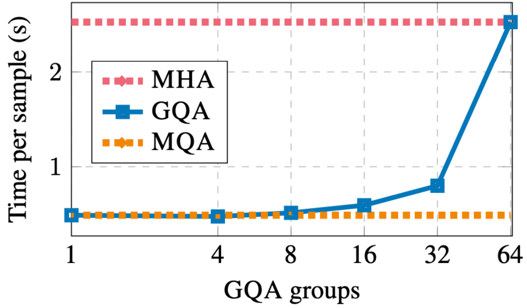


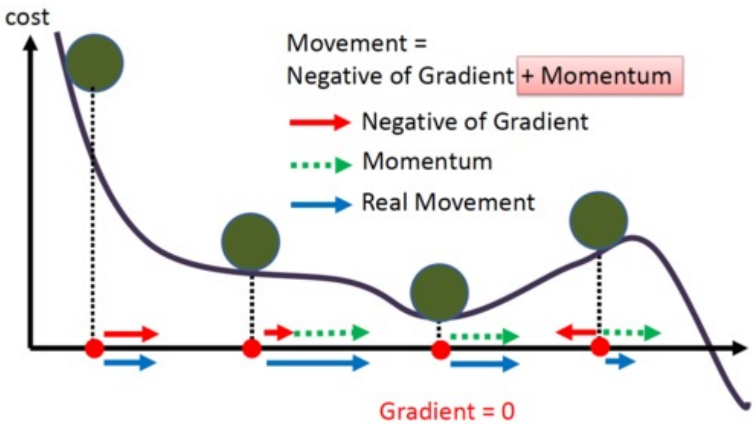
Figure 6: Time per sample for GQA-XXL as a function of the number of GQA groups with input length 2048 and output length 512. Going from 1 (MQA) to 8 groups adds modest inference overhead, with increasing cost to adding more groups.

- Improves inference scalability for our larger models

GQA: Training generalized multi-query transformer models from multi-head checkpoints, 2023

Optimizers

- Most modern models use "AdamW" optimizer (not vanilla Gradient Descent).
  - Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order "momentums".
  - "W" because it decouples "weight decay" from "learning rate". (Details out of scope for us. See the cited paper.)

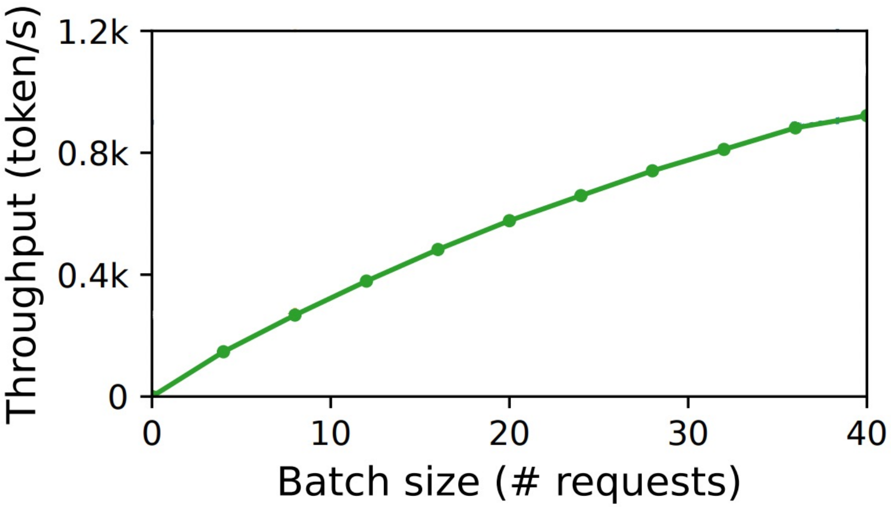


https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html
https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[Decoupled Weight Decay Regularization, 2017]

Batching Data

- Previously we talked about the importance of batching data
- GPUs are faster at Tensor operations and hence, we want to do batch processing
- The lager batch of data, the faster they get processed.
- Alas, the speedup is often sub-linear (e.g., 2x larger batch leads to less than 2x speedup).

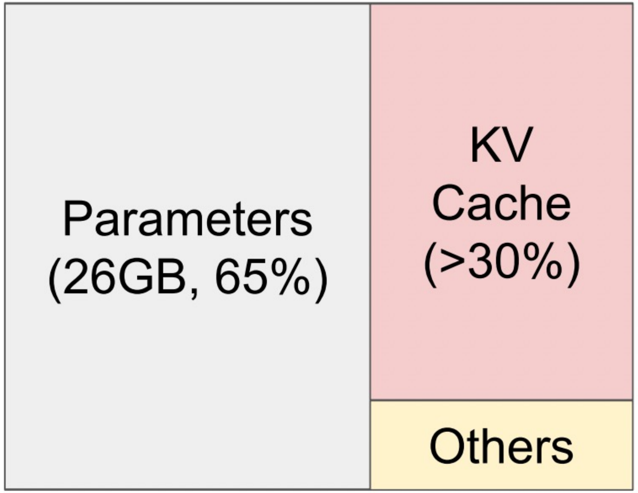
Model: 13B LLaMA on A100 GPU



The Memory Usage

- Here is the memory usage of an NVIDIA A100 when serving (i.e., no training)
  - Model: 13B LLaMA
  - Batch size of 10

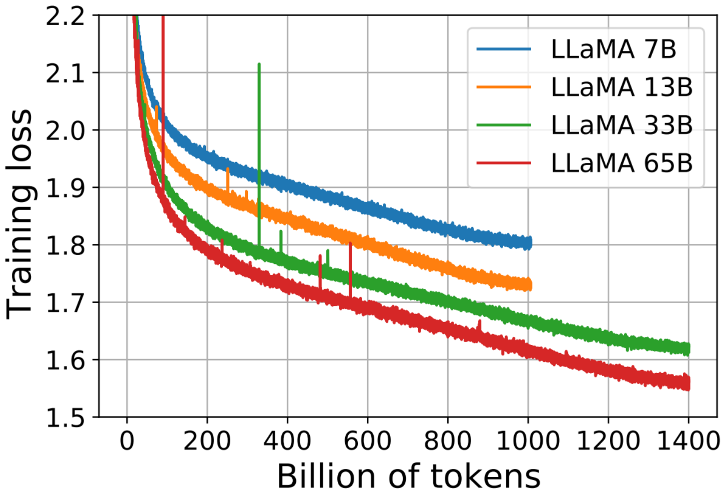
- Notice:
  - ~65% of your GPU memory is the model parameters that never change
  - ~32% of your memory are KV tensors that change for each input.
    - This KV cache will increase for larger batch sizes.

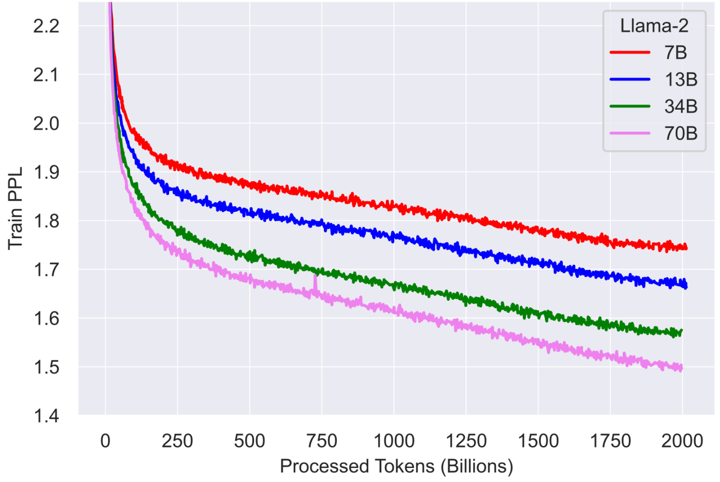


NVIDIA A100 40GB

Convergence

- In practice, your model's loss should continue to go down with more training on more data.
- So, the real bottlenecks are:
  - (1) compute;
  - (2) data.
- Sometimes training diverges (spikes in the loss), at which point practitioners usually restart training from an earlier checkpoint.





JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Summary

- There is many empirical knowledge that goes into engineering LMs.
- Here we covered a basic topics about data and architecture engineering.
- Various topics are forthcoming: scaling laws, efficient training, etc.