Transformers

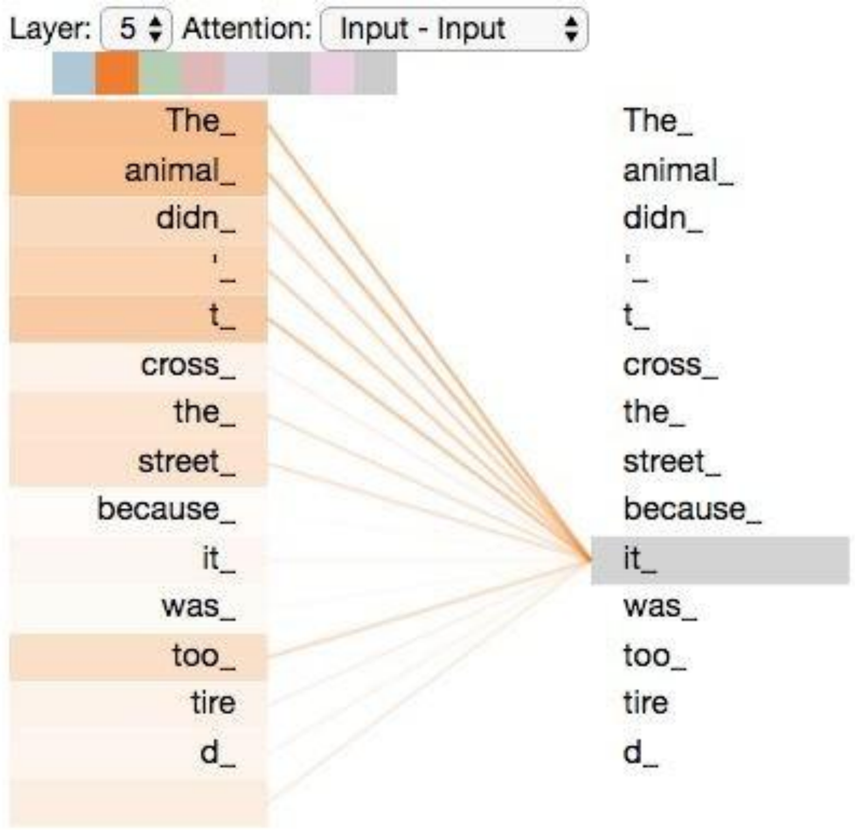
Alsu Sagirova
Neural Networks and Deep Learning Lab
MIPT

alsu.sagirova@phystech.edu

Agenda

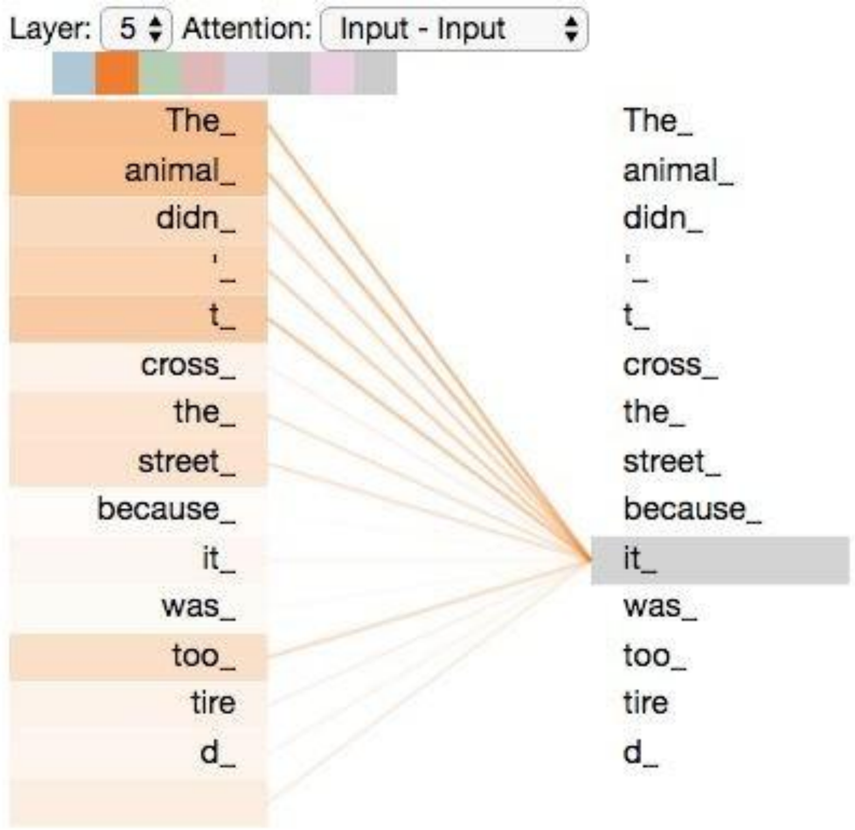
1. Transformer
2. PLMs

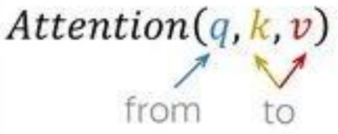
Attention recap



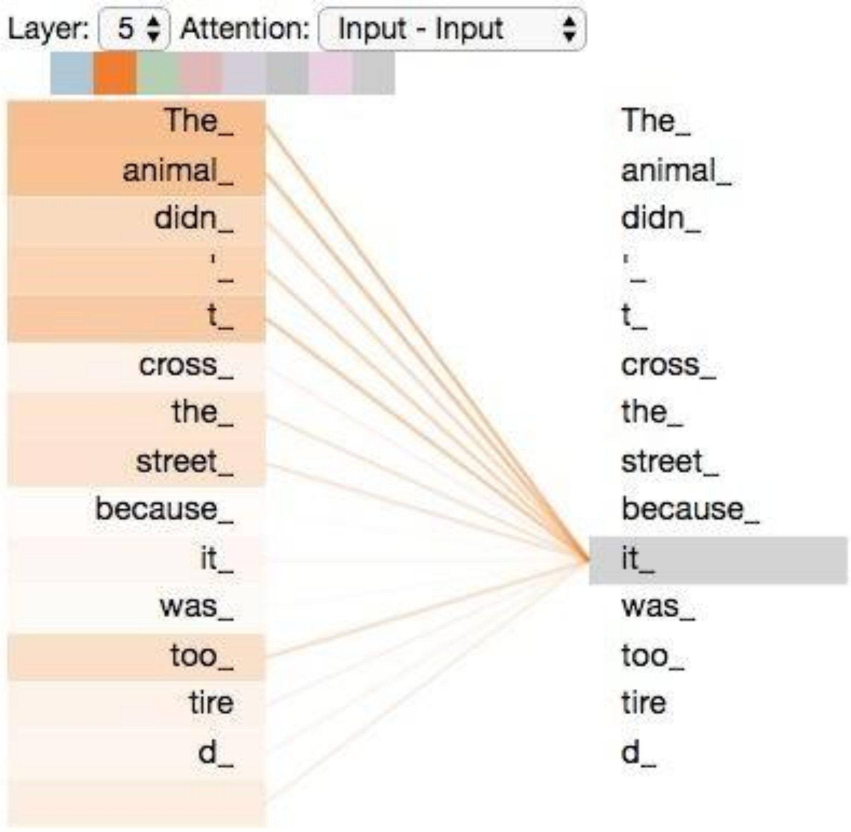
$Attention($

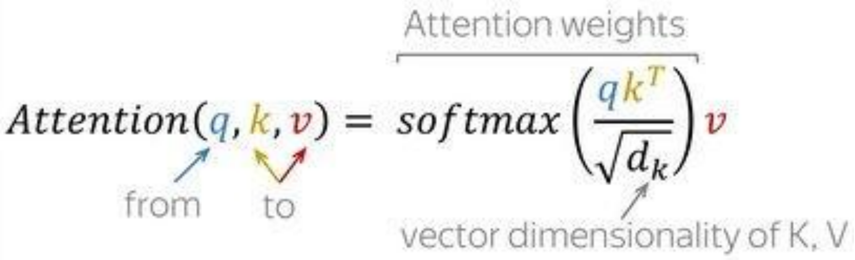
Attention recap





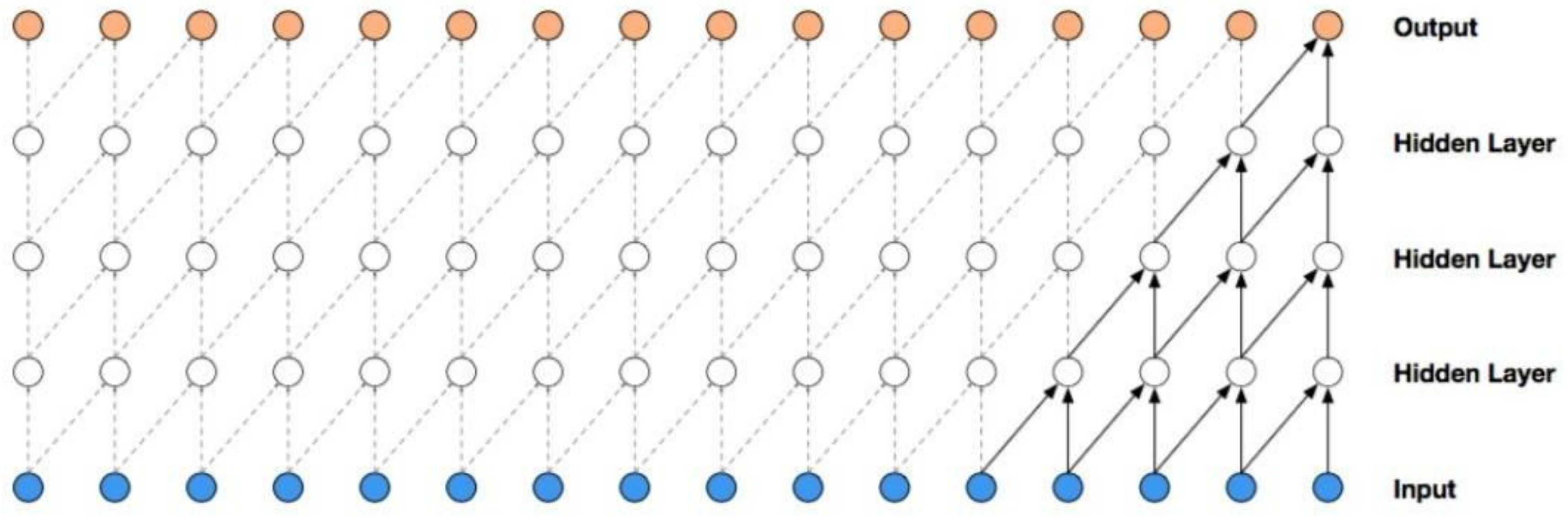
Attention recap



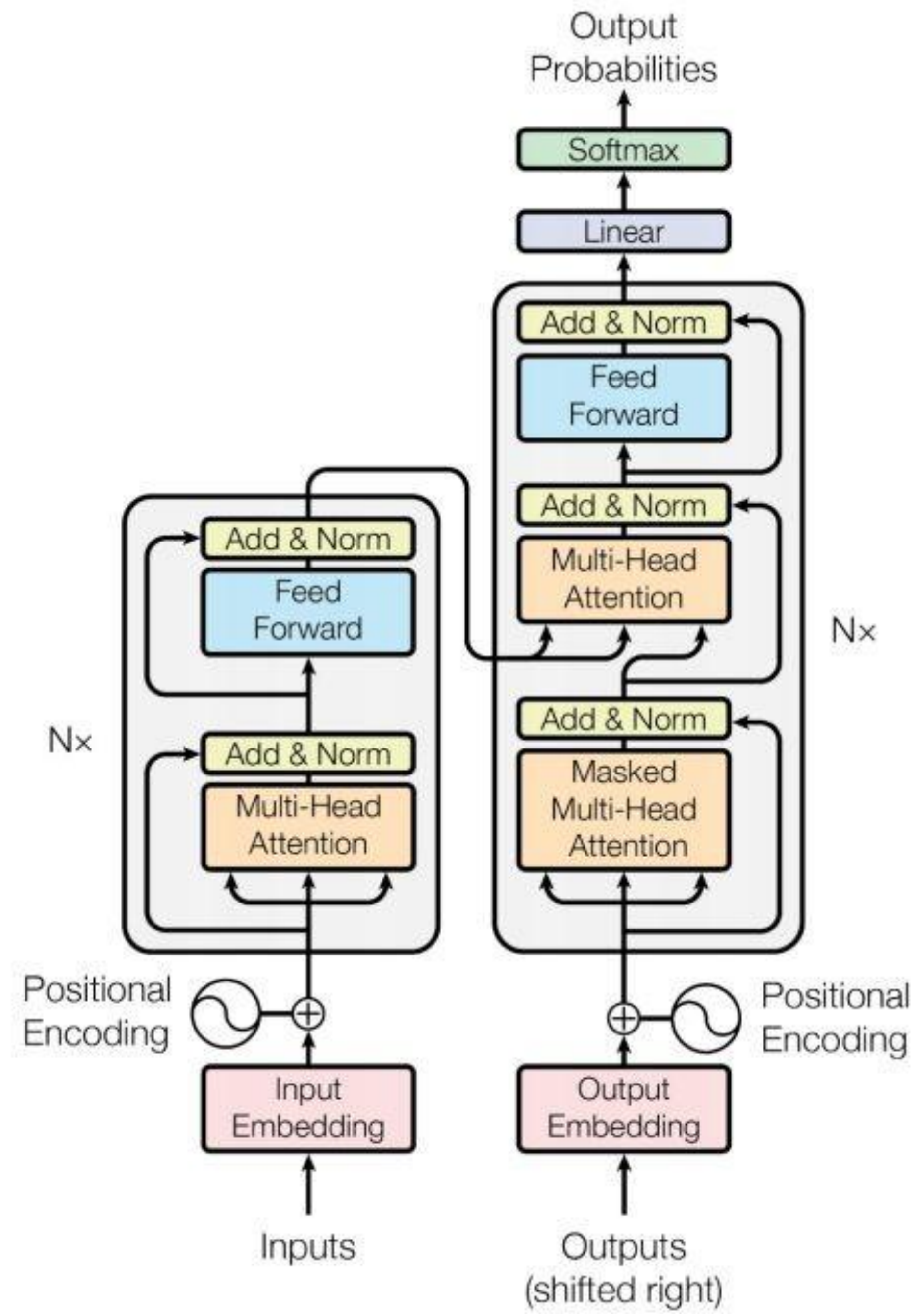


Attention is All You Need

- The model has access to the entire history at each hidden layer
- But the context will have different input length at each time step
- Attention is used to aggregate the context information by attending to one or a few important tokens from the past



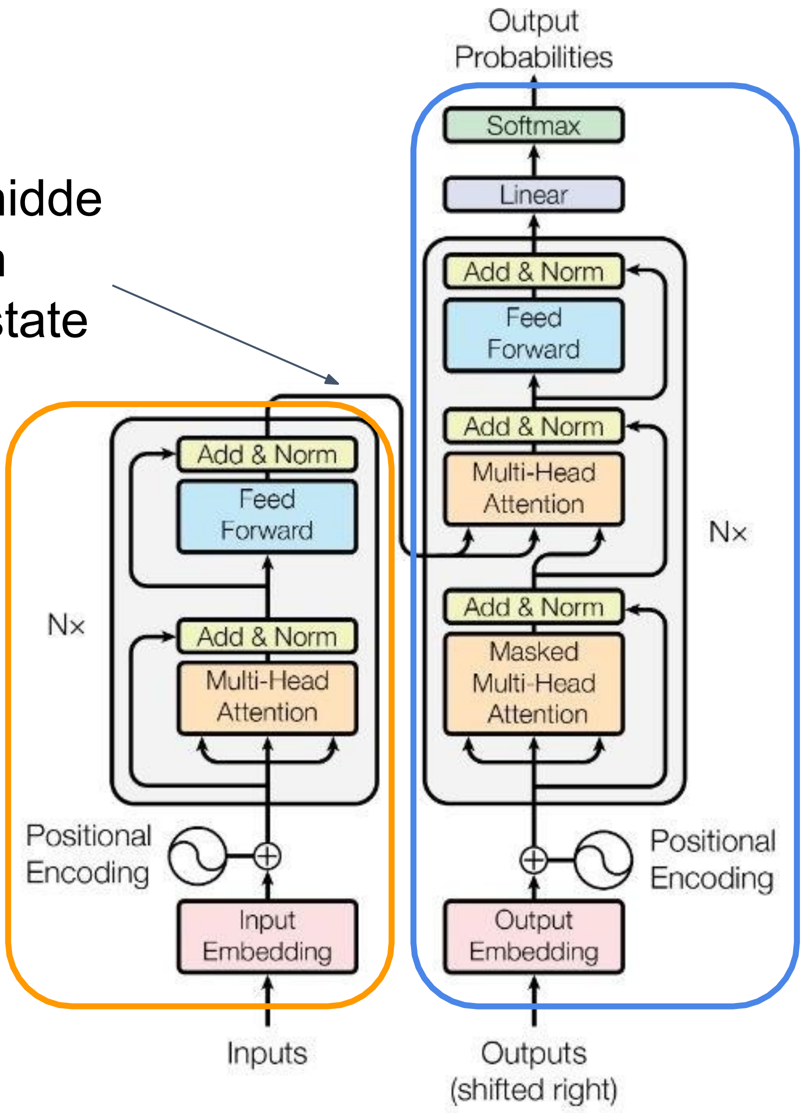
Transformer



Seq2Seq

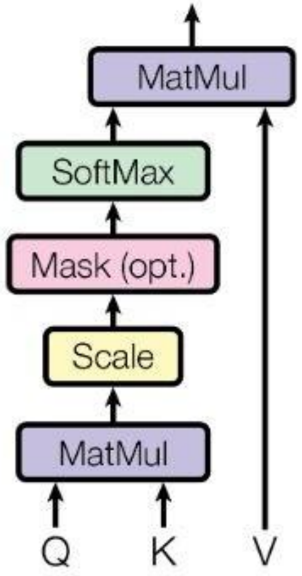
hidde
n
state

Encoder

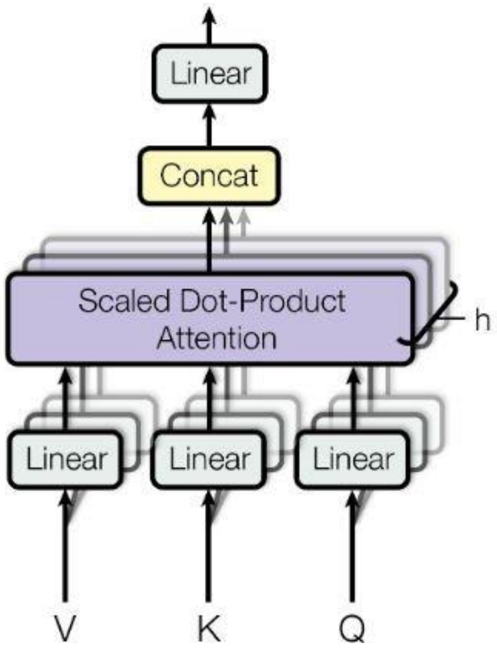


Decoder

Attention



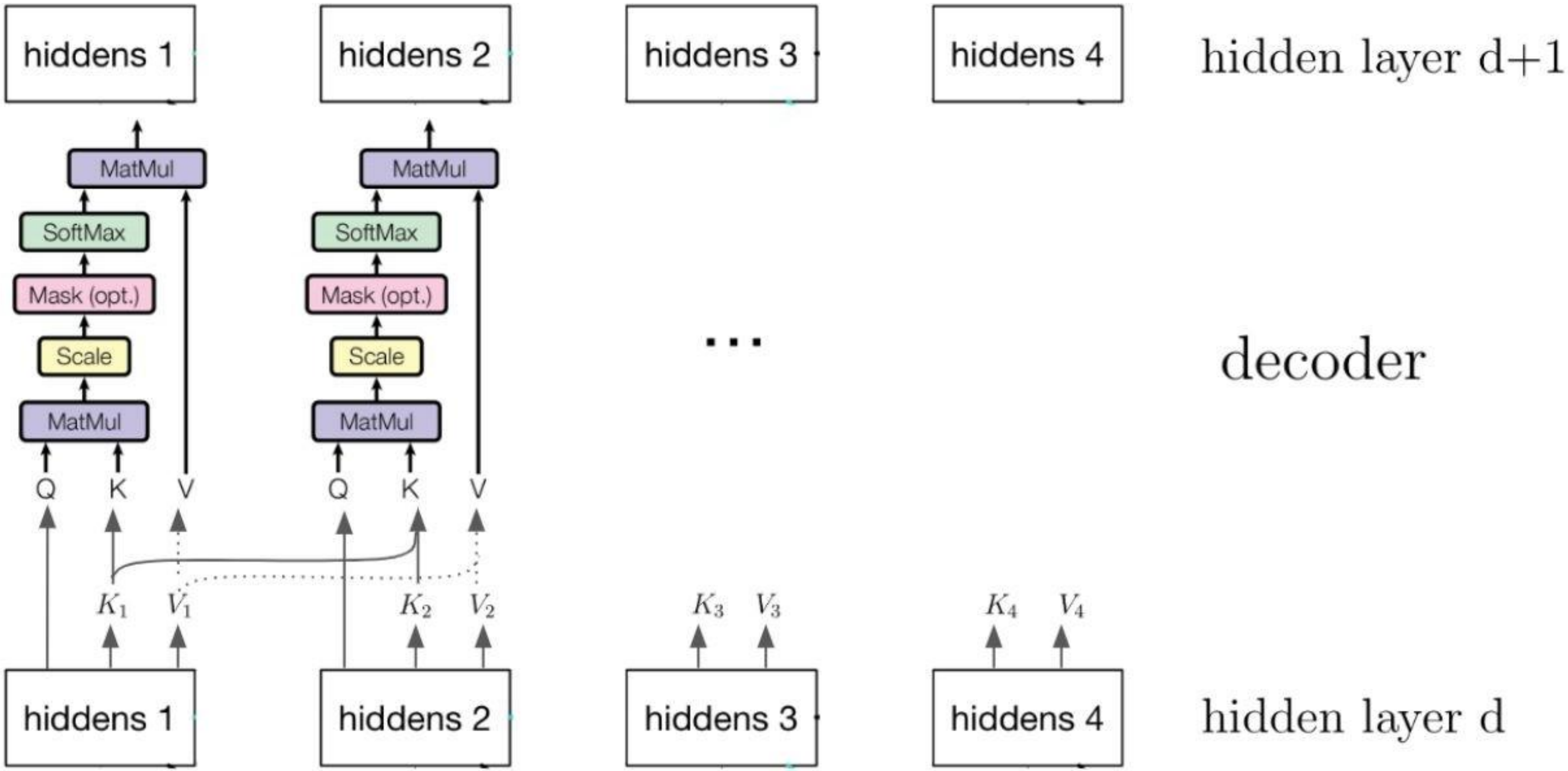
Scaled dot-product attention



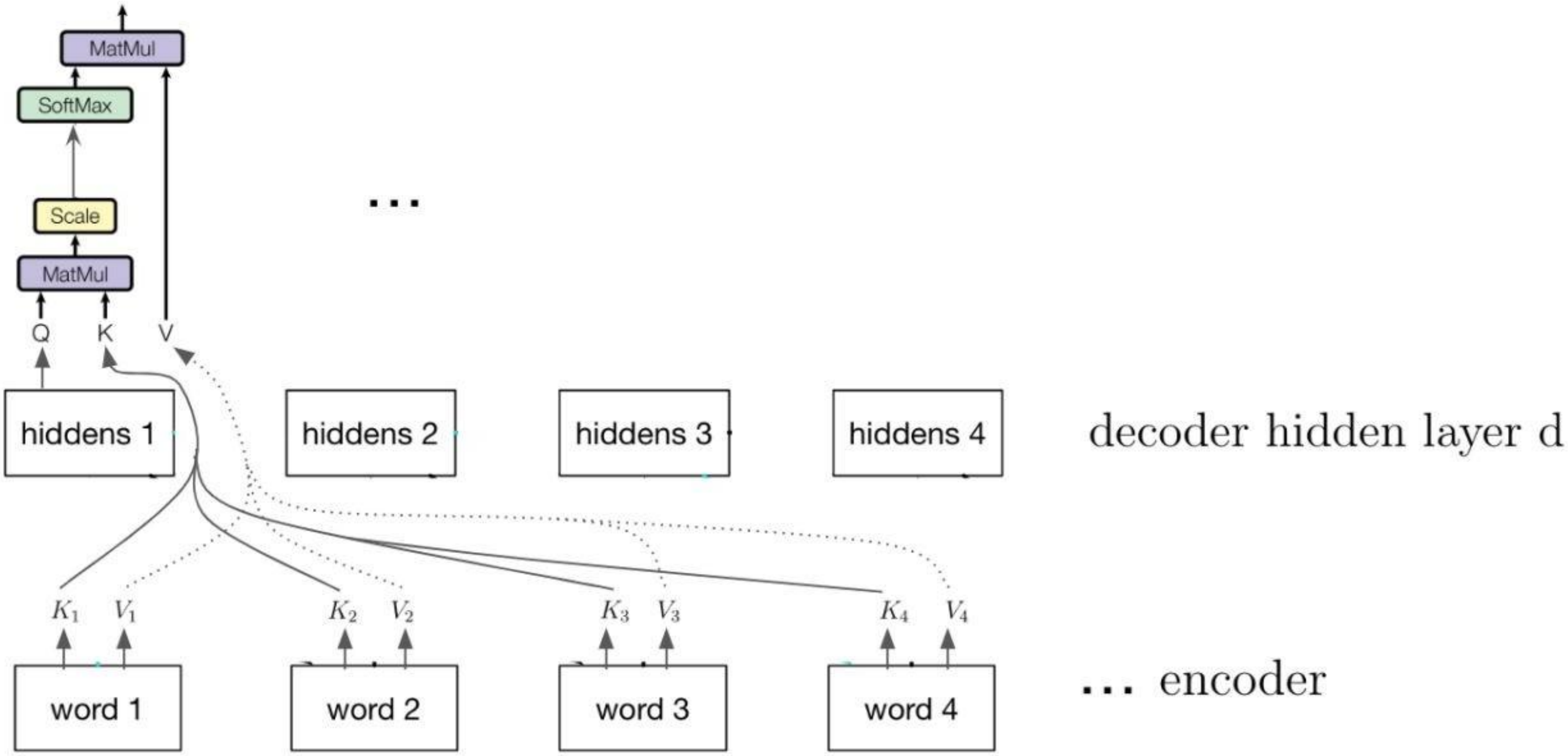
Multi-head attention

Vaswani et al. (2017)

Self-attention



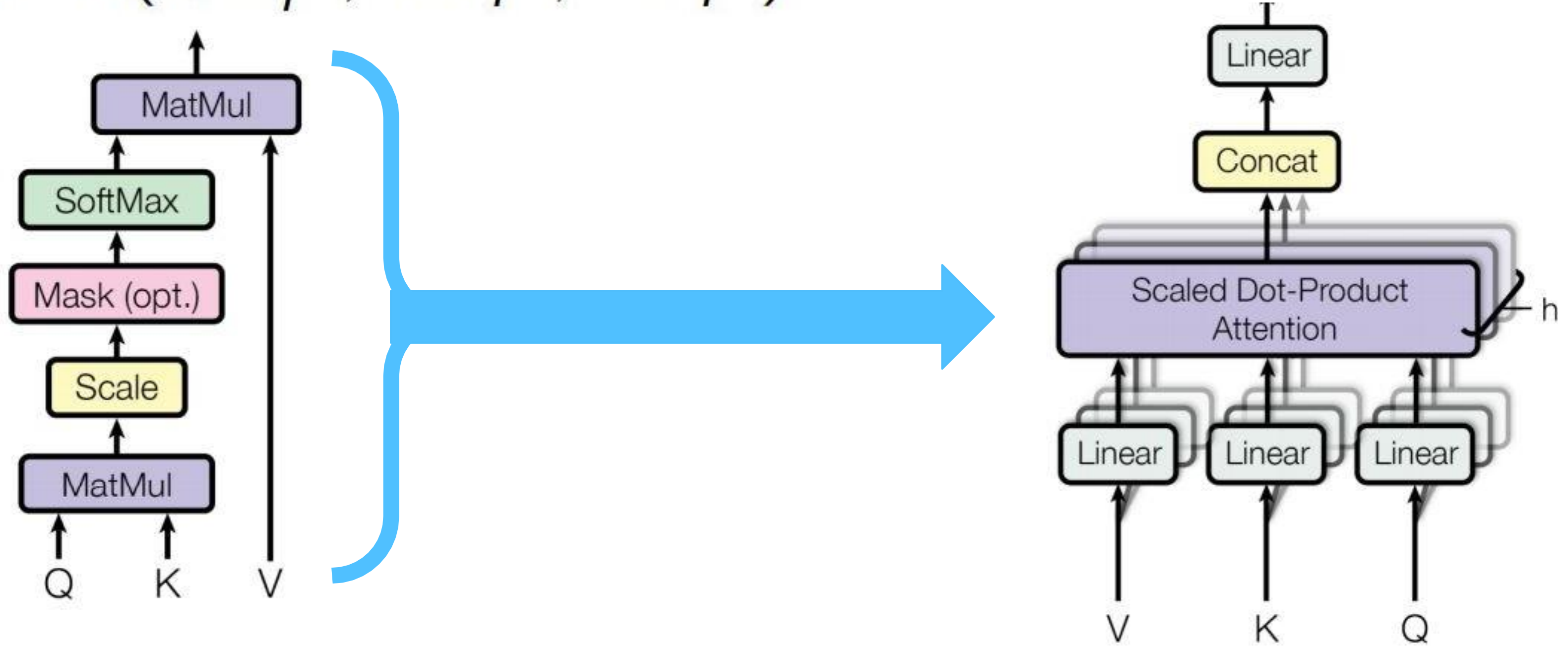
Cross-attention



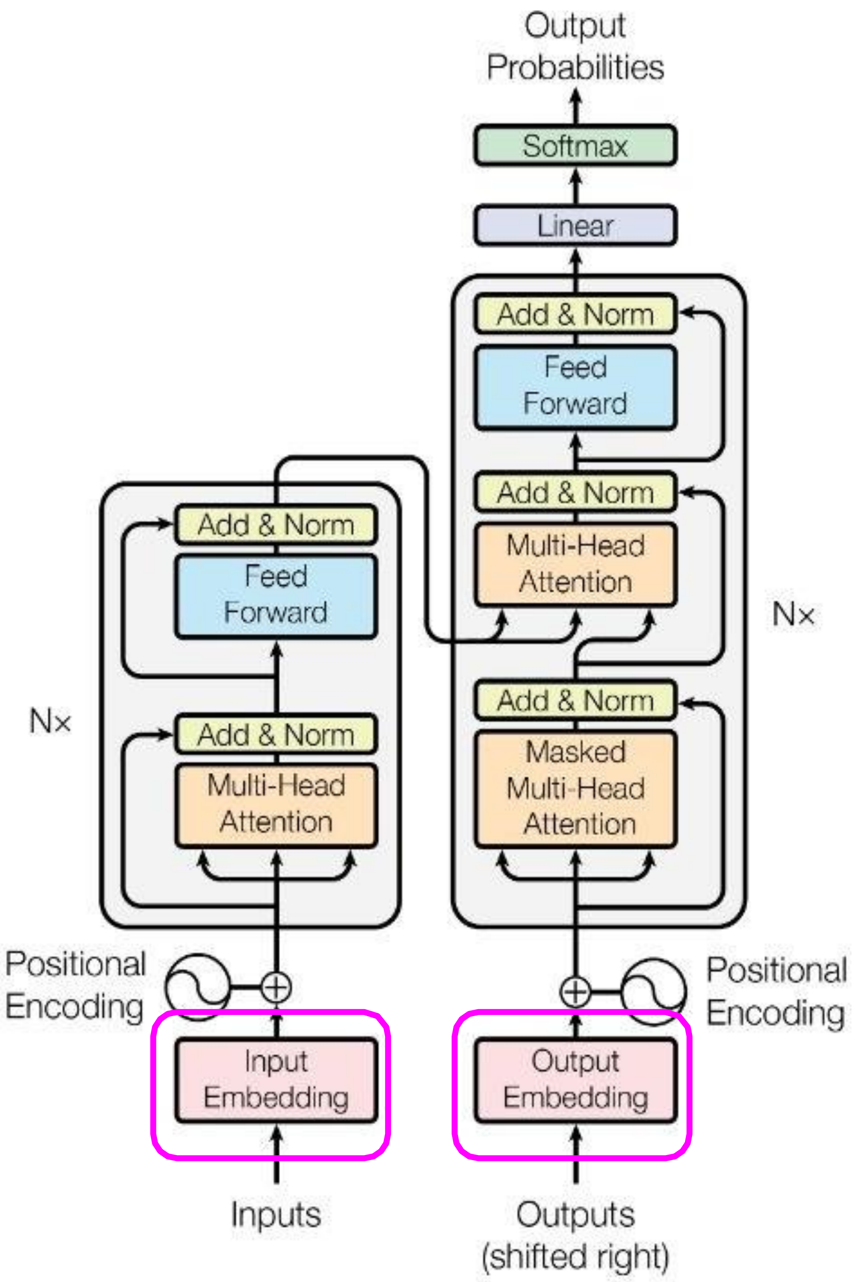
Multi-Head Attention

- The Scaled Dot-Product Attention attends to one or few entries in the input key-value pairs
    - Humans can attend to many things simultaneously
- The idea: apply Scaled Dot-Product Attention multiple times on the linearly transformed $\text{MultiHead}(Q, K, V) = \text{concat}(\mathbf{c}_1, \cdots, \mathbf{c}_h) W^O$,

$$\mathbf{c}_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V).$$



Tokenization



Byte-Pair Encoding (BPE)

1. Start with a vocabulary containing only characters and an “end-of-word” symbol.
2. Using a corpus of text, find the most common pair of adjacent characters “r, e”; add subword “re” to the vocab.
3. Replace instances of the character pair with the new subword; repeat until desired vocab size.

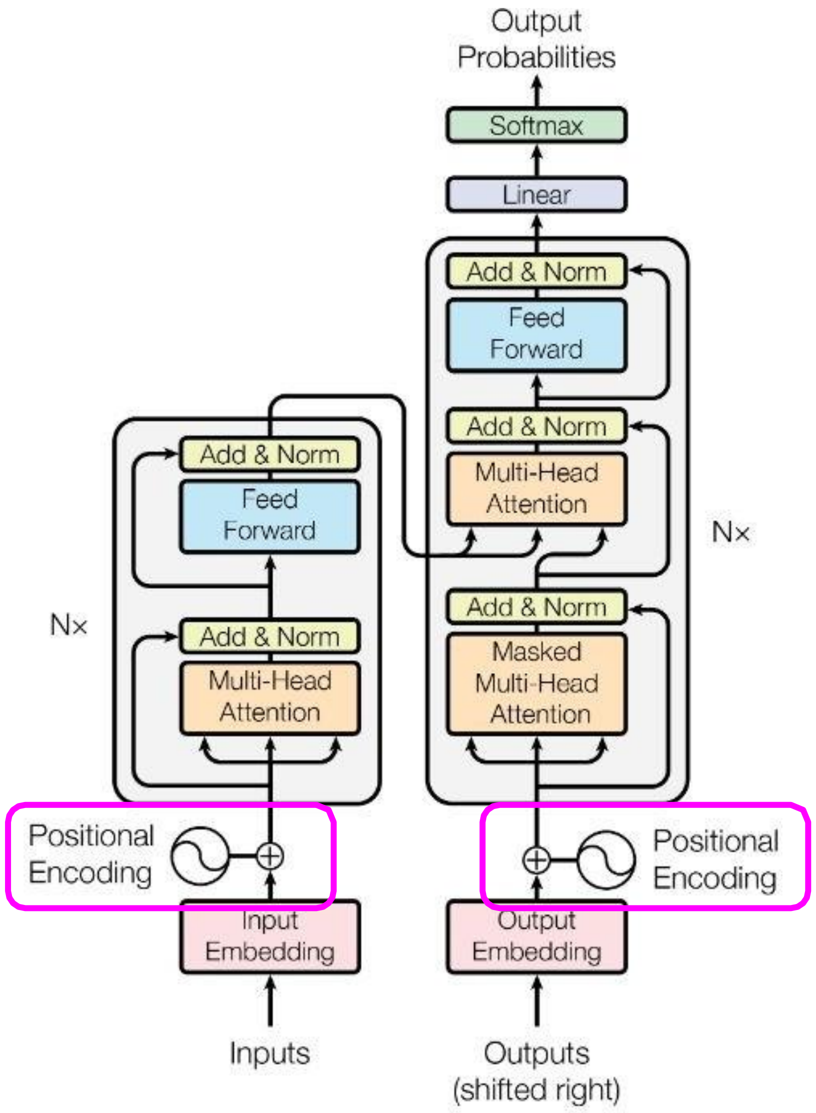
```
u-n-r-e-l-a-t-e-d
u-n re-l-a-t-e-d
u-n re-l-at-e-d
u-n re-l-at-ed
un re-l-at-ed
un re-l-ated
un rel-ated
un-related
unrelated
```

Originally BPE was used in NLP for machine translation.
Now a similar method called WordPiece is used in pretrained models.

*[Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units, 2016]

Positional Encoding

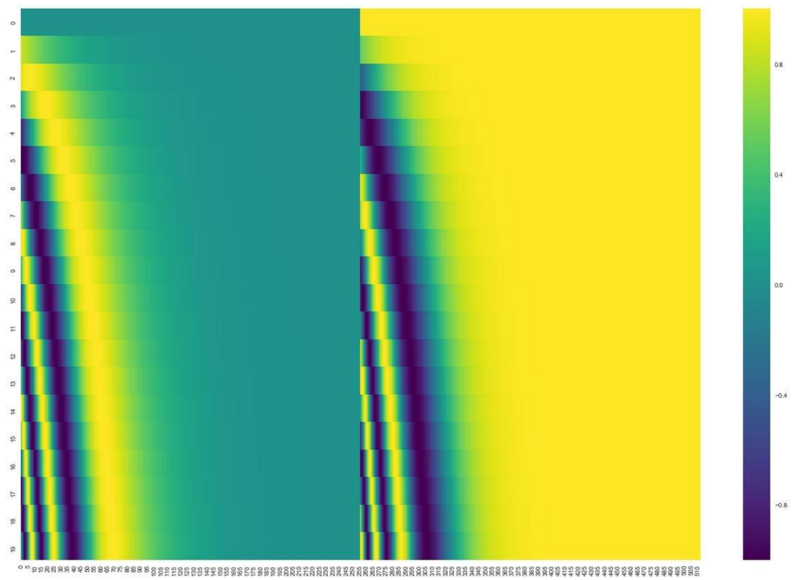
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Positional Encoding

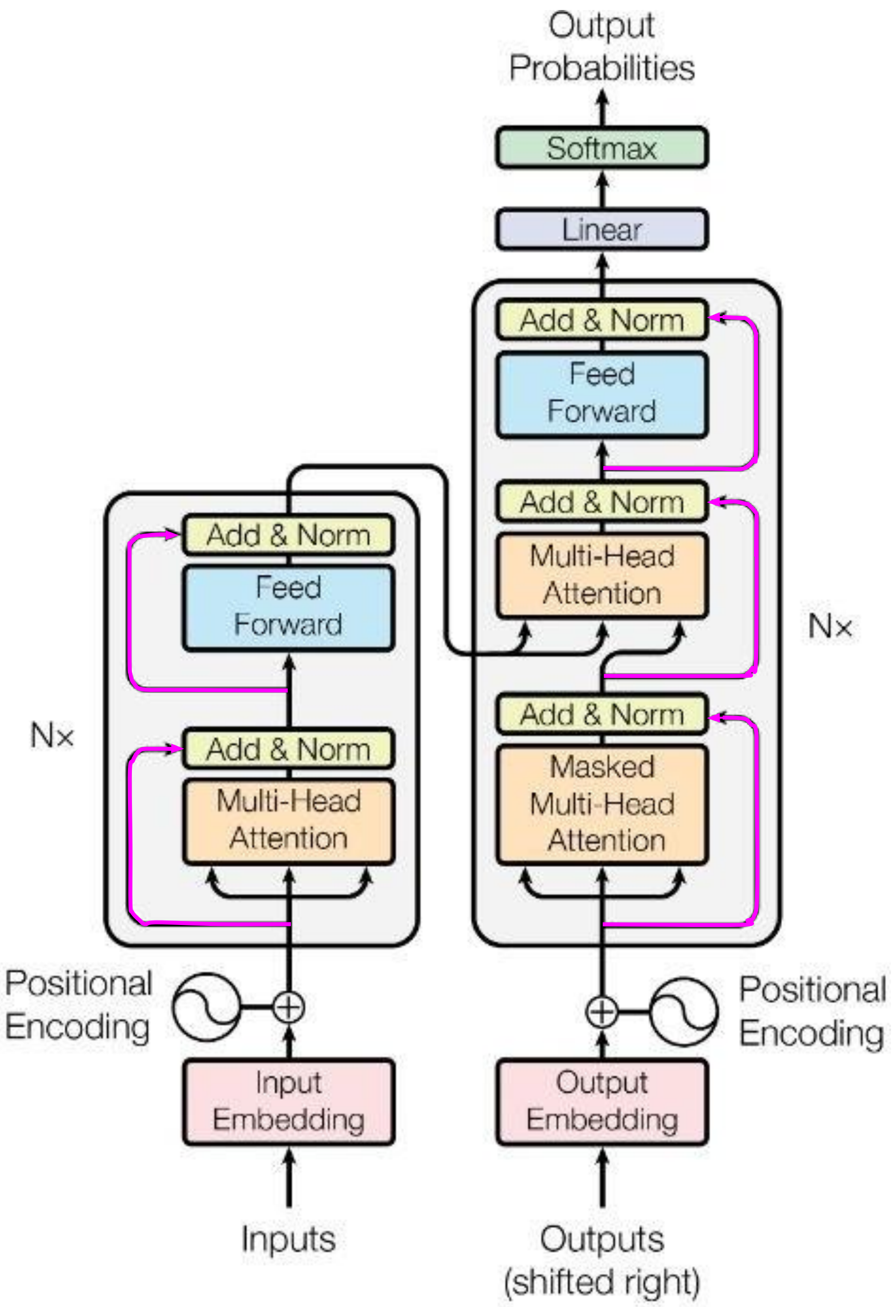
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

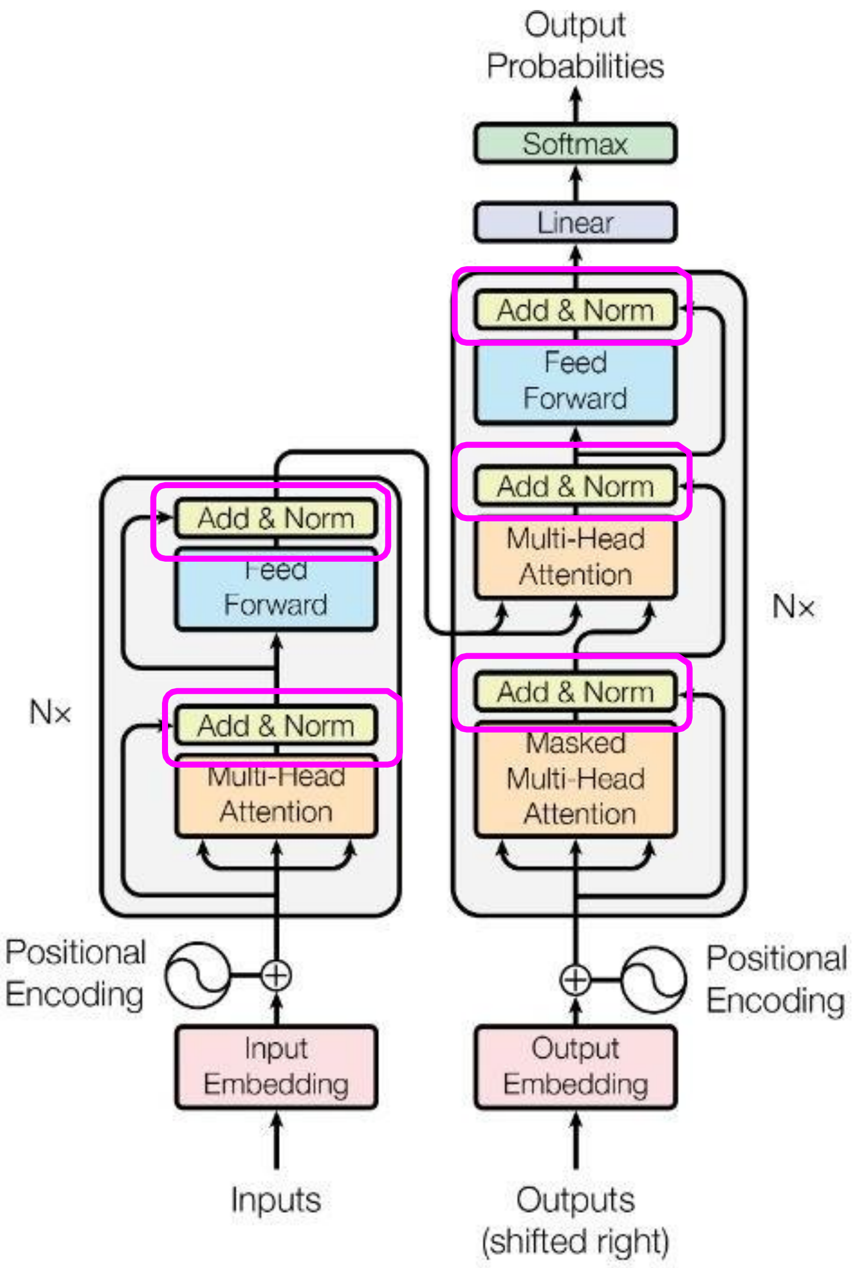


Residual Connections

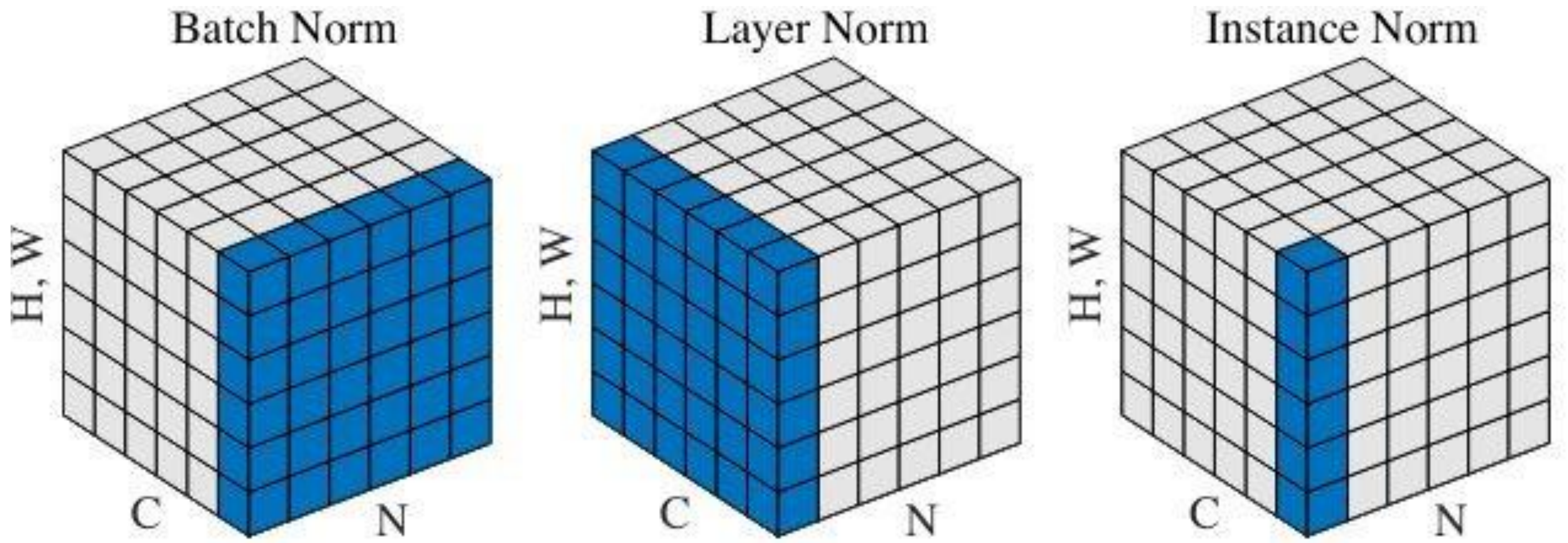
allowing gradients to flow
freely & speed up training



Layer Normalization

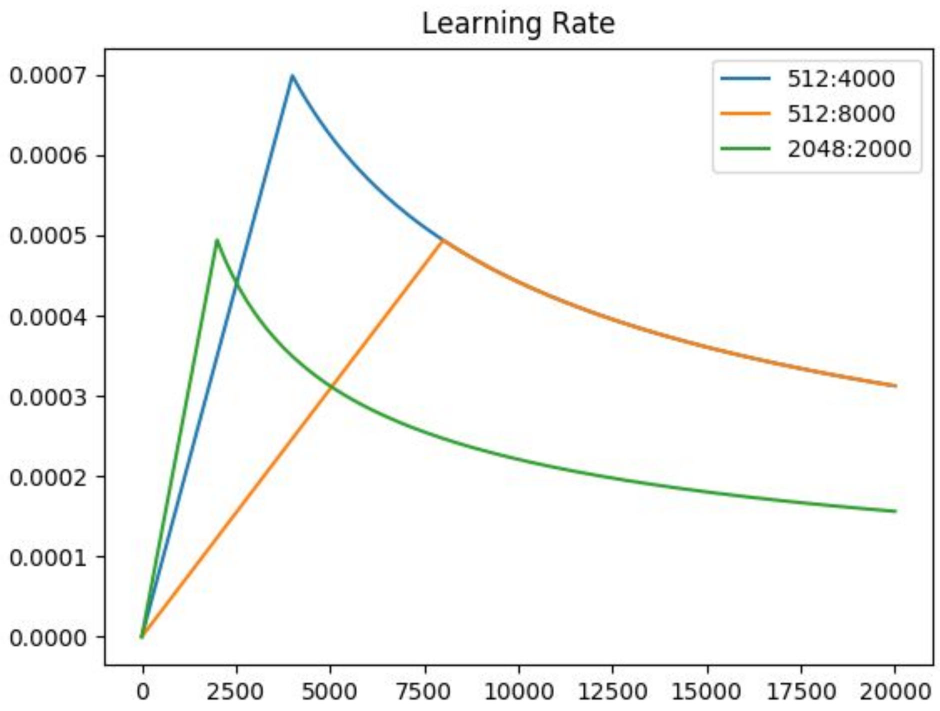


Layer Normalization



Noam Learning Rate Schedule

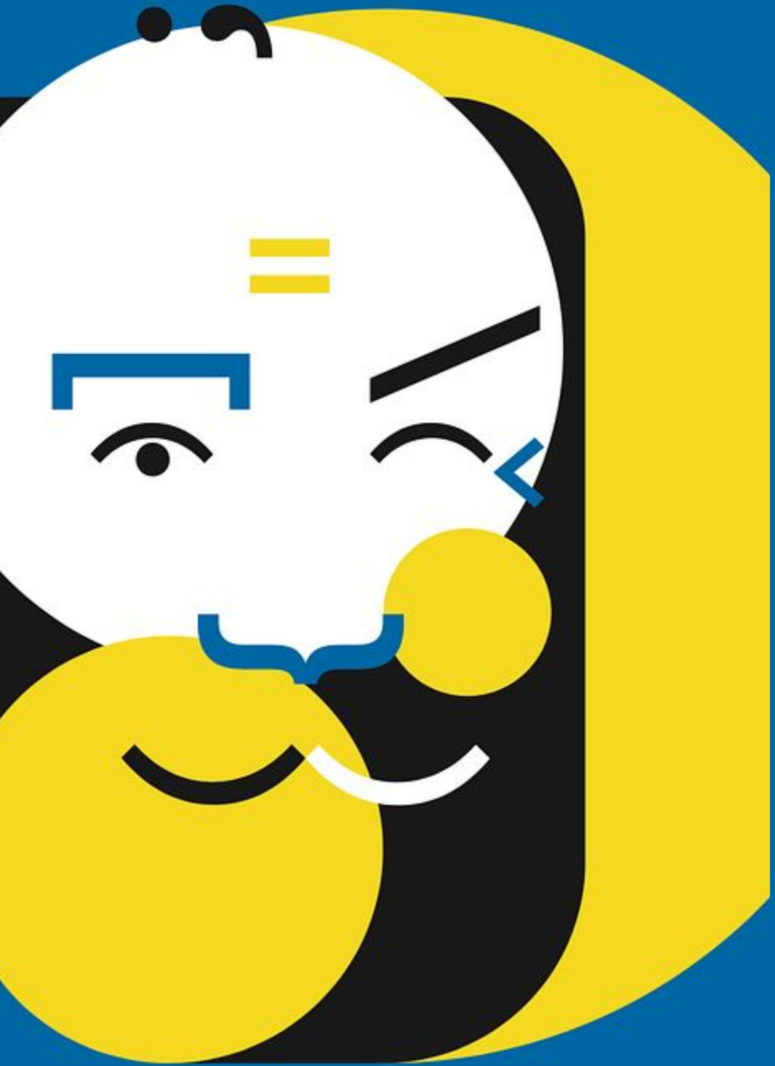
$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$



Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	



PLMs

Pretrained models

Through **the pretrained model**, we "transfer" the knowledge of its training data to our task-specific model.

Advantages:

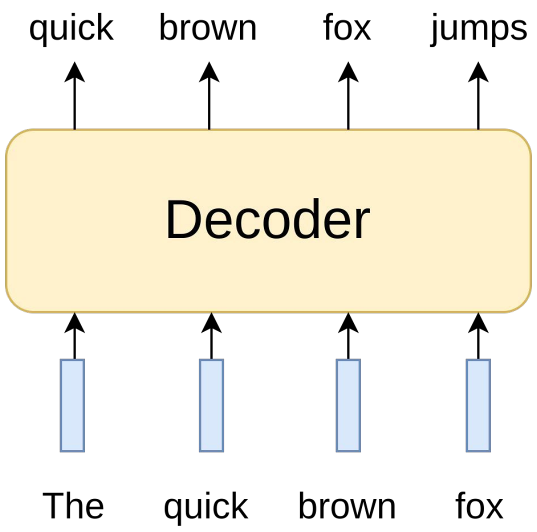
- strong representations of language
- strong parameter initializations for strong NLP models
- strong probability distributions over language that we can sample from

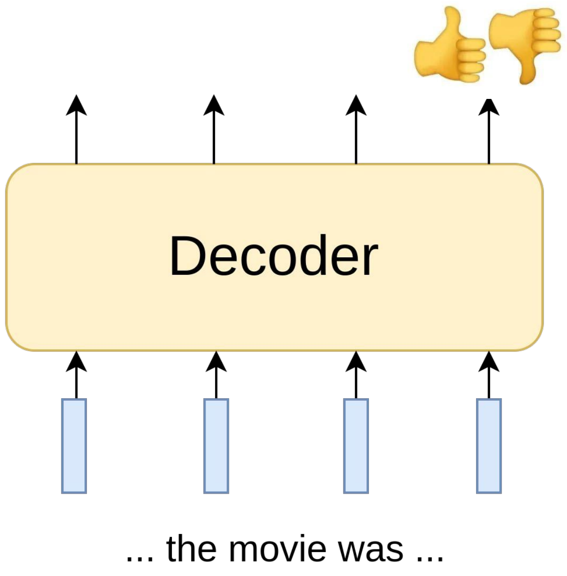
Language modeling for pretraining

In 2015, Andrew M. Dai and Quoc V. Le* from Google presented the idea of pretraining through language modeling followed by the task-specific fine-tuning.

- **Pretrain** a neural network to perform language modeling on a *large* amount of text
- Save the network parameters

- **Fine-tune** on your task
- *Not many* labels, adapting to the task!

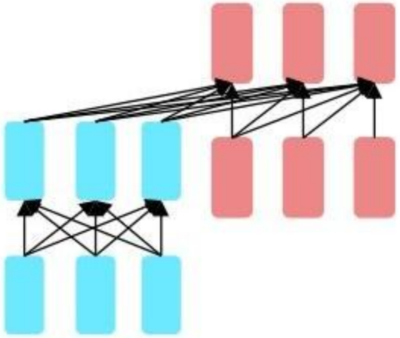




* Semi-supervised Sequence Learning, 2015

PLM architectures

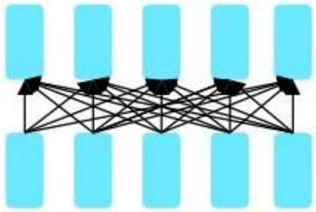
Transformer-like



Encoder-Decoders

(Transformer, Meena, T5)

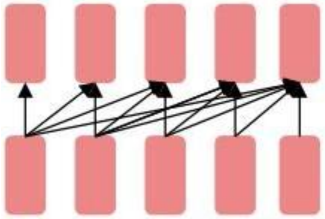
Bidirectional context architectures



Encoders

(BERT, RoBERTa etc.)

LMs used for sequence generation that do not condition on future

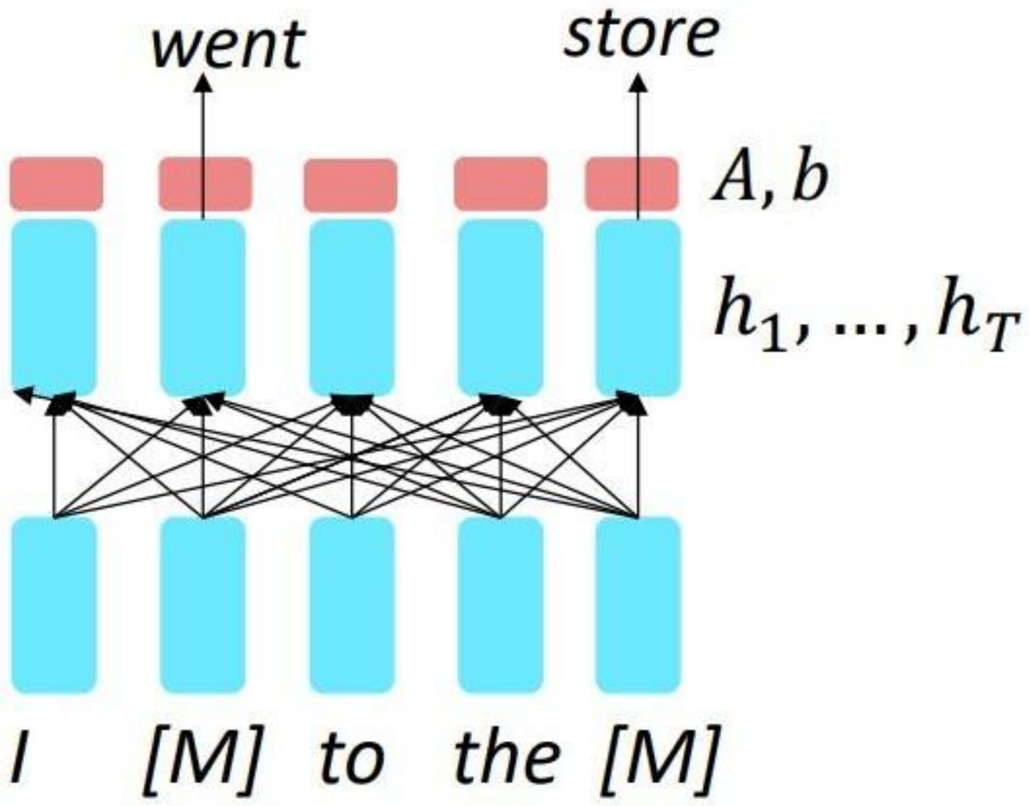


Decoders

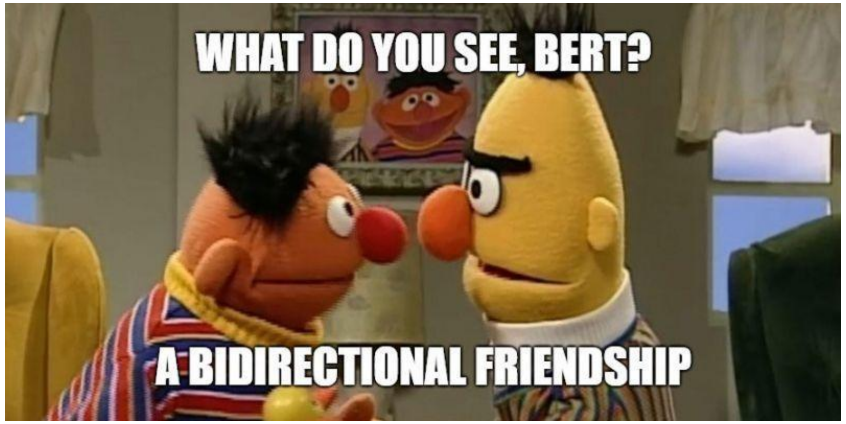
(GPT-2, GPT-3, LaMDA)

Encoder-only

- Bidirectional access
- LM pretraining impossible!
- Solution:
  Masked Language Modeling
- Loss is calculated for masked tokens only



EXAMPLE: BERT ([Bidirectional Encoder Representations from Transformers](#))



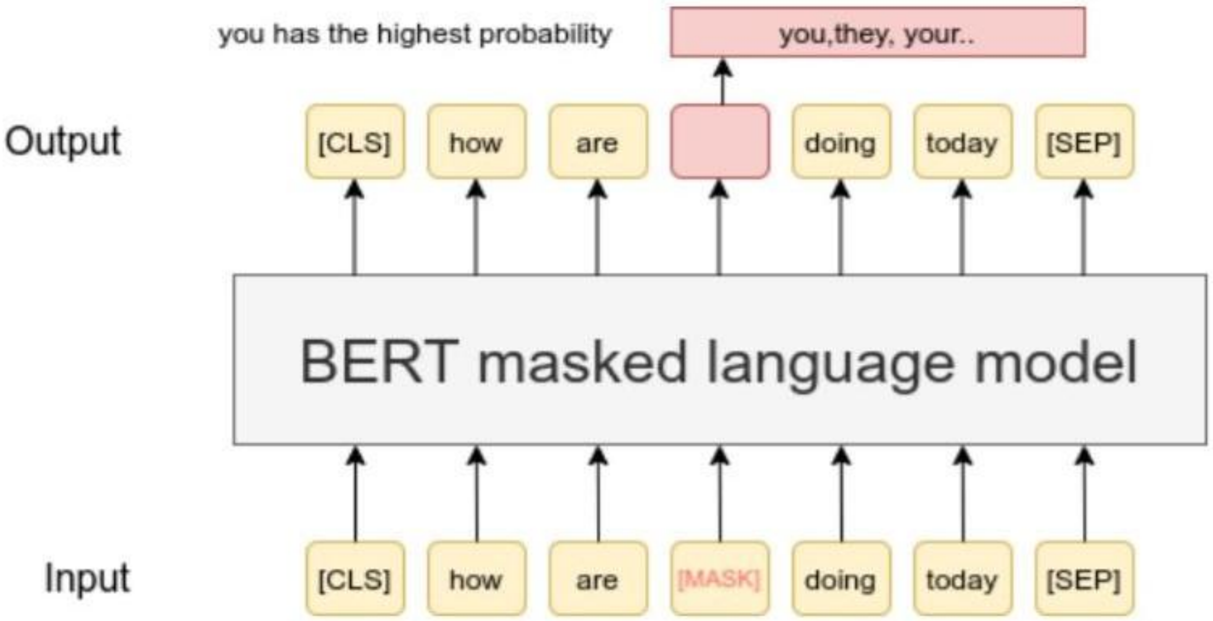
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

- Released models
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params
- Datasets: BooksCorpus (800M words) and English Wikipedia (2,500M words)
- Pretraining done with 64 TPU chips for a total of 4 days
- Fine-tuning on a single GPU

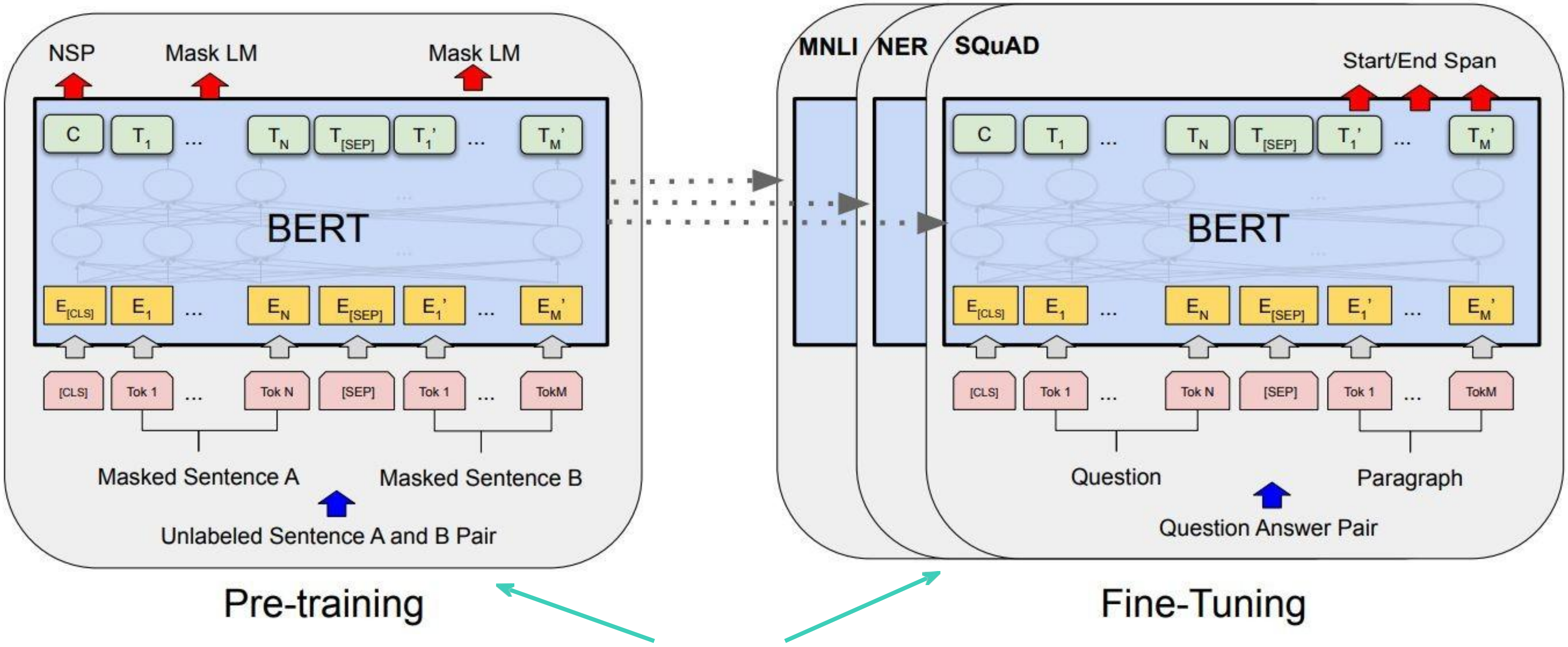
MLM pretraining for BERT

How BERT was pretrained:

- A random 15% of input tokens are masked
- [MASK]token does not appear in fine-tuning → input tokens are replaced with
  - 80%: replace with [MASK]
    `my dog is hairy -> my dog is [MASK]`
  - 10%: replace with random token
    `my dog is hairy -> my dog is apple`
  - 10%: leave token unchanged
    `my dog is hairy -> my dog is hairy`



BERT



- Unified architecture for pretraining & fine-tuning
- The same pretrained model weights for different downstream tasks
- During fine-tuning, all parameters are fine-tuned

BERT

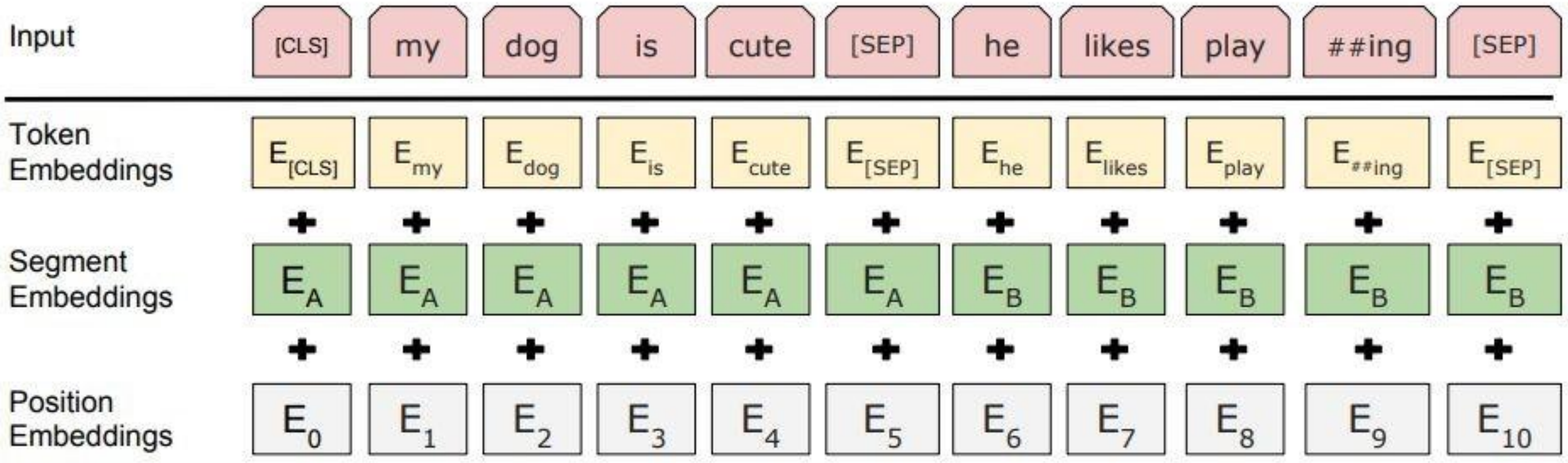


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

NB *In addition to MLM pretraining task, authors test Next Sentence Prediction pretraining (check if the second sentence in input is actually following the first sentence).*

BERT

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

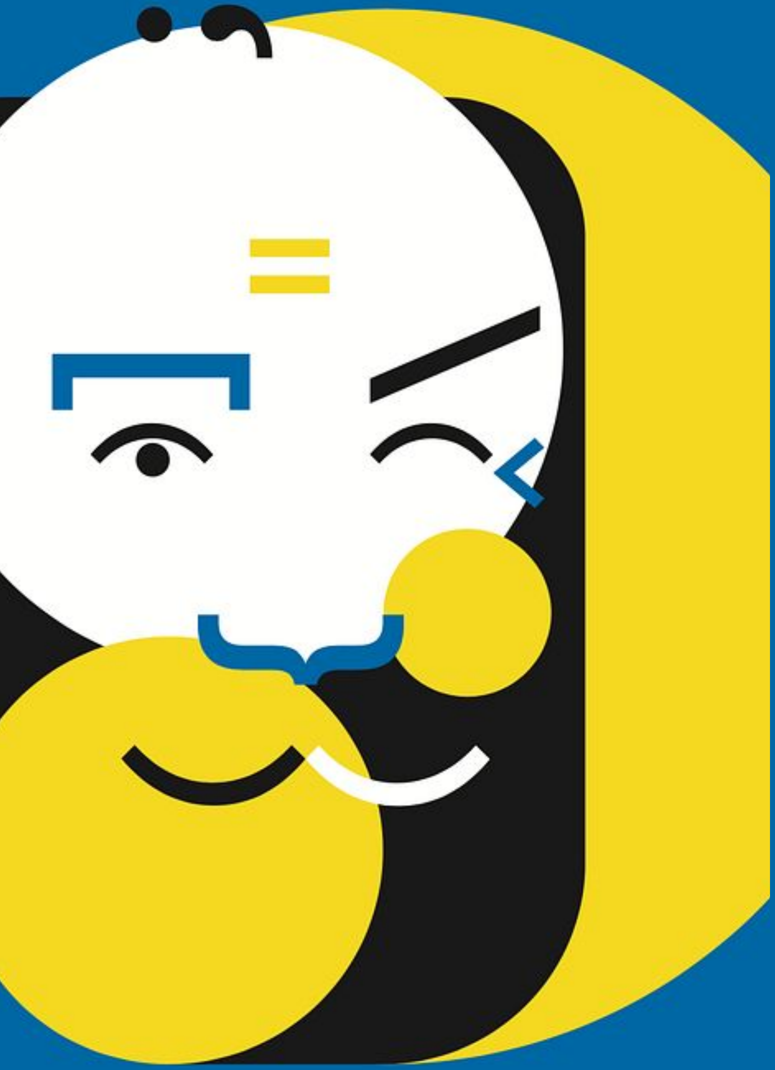
Table 1: GLUE Test results, scored by the evaluation server (`https://gluebenchmark.com/leaderboard`). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Major BERT successors

- RoBERTa:
    - more train data
    - bigger batches
    - longer inputs
    - remove next sentence prediction!

improved pretraining without architecture change

- SpanBERT
    - pre-training method for better span representation and prediction
    - masking contiguous spans of words makes a harder, more useful pretraining task

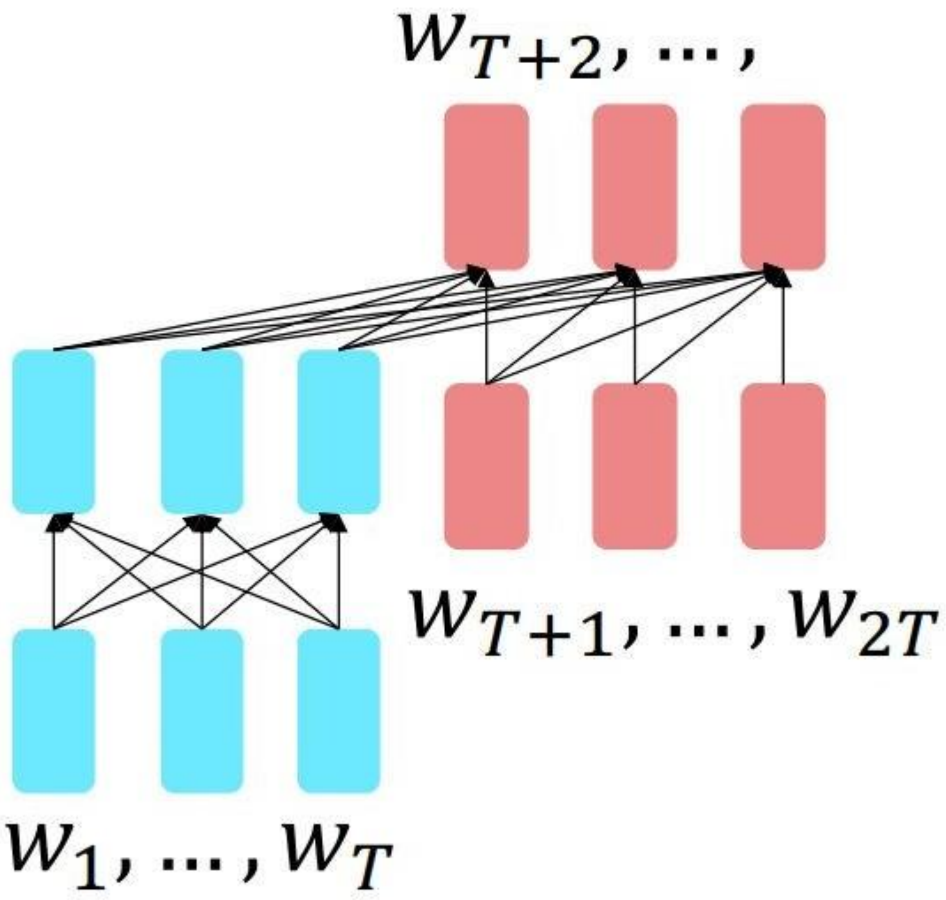


Encoder-decoders

Encoder-decoders pretraining

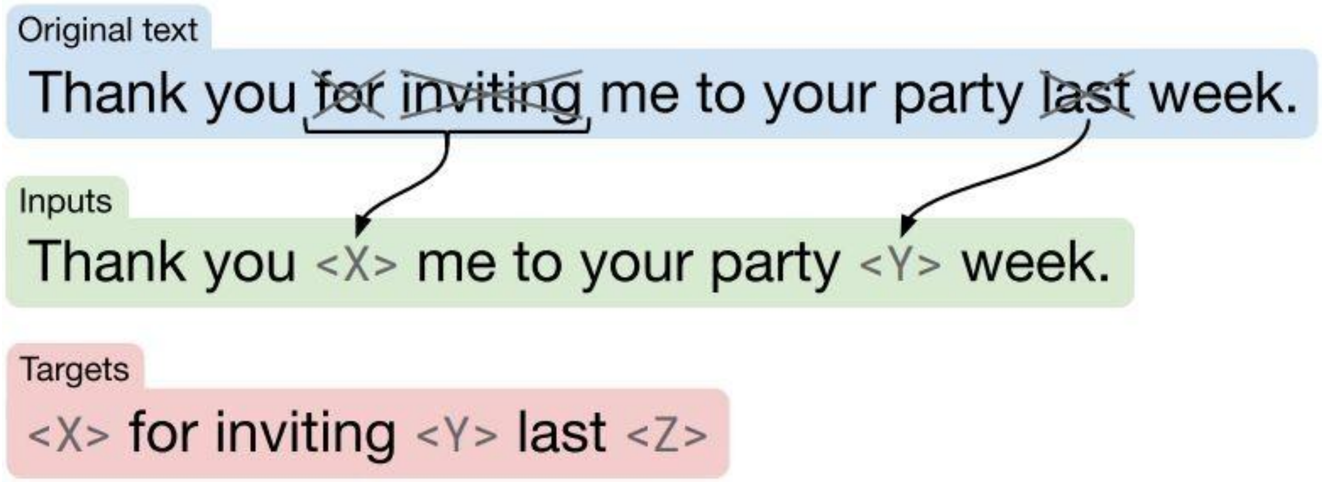
- Decoder: language modeling
- Encoder: language modeling where a prefix of every input is provided to the encoder and is not predicted.

EXAMPLE:
T5 ([Text-to-Text Transfer Transformer])



T5 pretraining objective

- Corrupted words are chosen randomly
- Each consecutive span of corrupted tokens is replaced by a sentinel token (<X> and <Y>)unique over the example
- The output sequence: dropped-out spans, delimited by the sentinel tokens used to replace them in the input plus a final sentinel token <Z>.

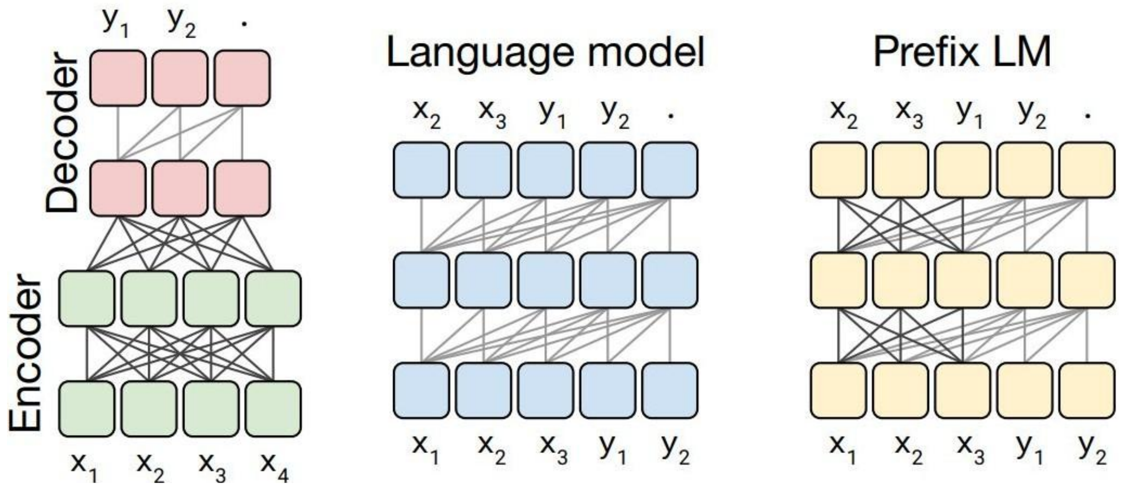


C4: The Colossal Clean Crawled Corpus

From the paper:

- We discarded any page with fewer than 5 sentences and only retained lines that contained at least 3 words.
- We removed any page that contained any word on the "List of Dirty, Naughty, Obscene or Otherwise Bad Words"
- Many of the scraped pages contained warnings stating that Javascript should be enabled so we removed any line with the word Javascript.
- Some pages had placeholder "lorem ipsum" text; we removed any page where the phrase "lorem ipsum" appeared.
- Some pages inadvertently contained code. Since the curly bracket "{" appears in many programming languages (such as Javascript, widely used on the web) but not in natural text, we removed any pages that contained a curly bracket.
- To deduplicate the data set, we discarded all but one of any three-sentence span occurring more than once in the data set

T5 pretraining objectives exploration



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
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

T5 pretraining objectives exploration

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

Table 3: Examples of inputs and targets produced by some of the unsupervised objectives we consider applied to the input text "Thank you for inviting me to your party last week ." Note that all of our objectives process *tokenized* text.

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**
BERT-style (Devlin et al., 2018)	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Table 4: Performance of the three disparate pre-training objectives

Attention patterns

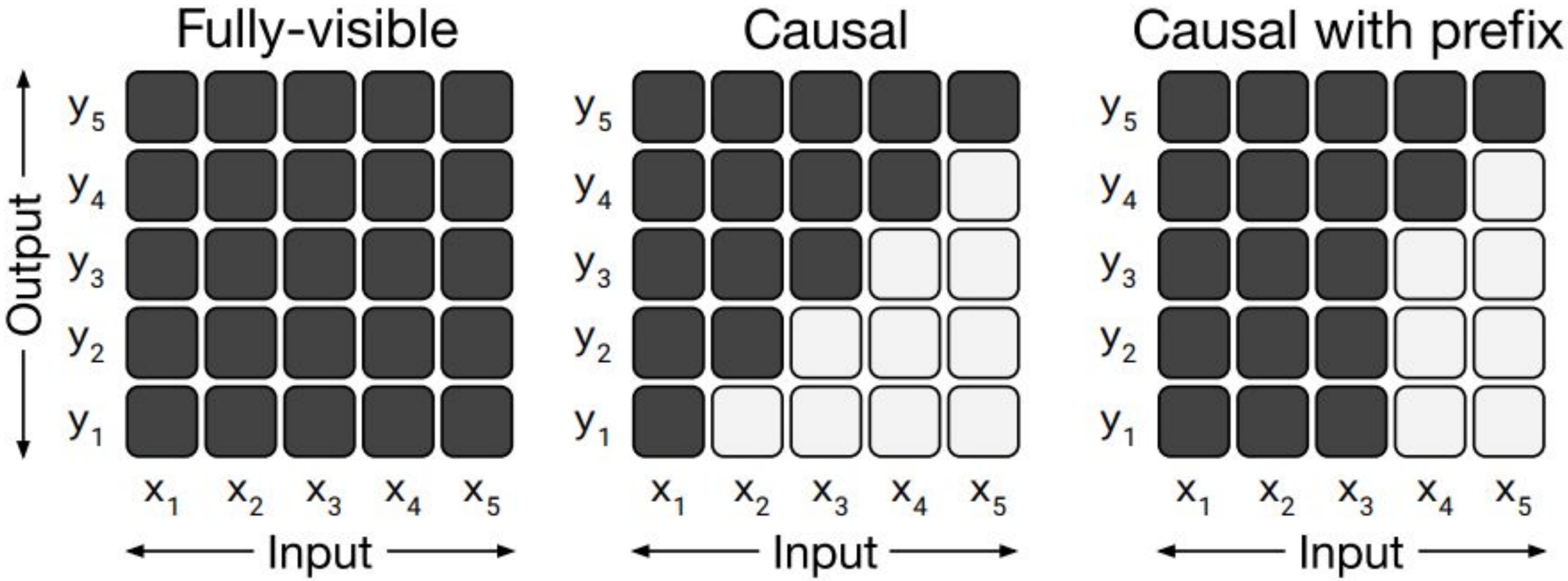


Figure 3: Matrices representing different attention mask patterns. The input and output of the self-attention mechanism are denoted x and y respectively.

Results

	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline average	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Baseline standard deviation	0.235	0.065	0.343	0.416	0.112	0.090	0.108
No pre-training	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

Table 1: Average and standard deviation of scores achieved by our baseline model and

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
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

Table 8: Performance resulting from pre-training on different data sets. The first four variants are based on our new C4 data set.

Results

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

Table 10: Comparison of different alternative fine-tuning methods that only update a subset of the model's parameters. For adapter layers, d refers to the inner dimensionality of the adapters.

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

Table 12: Comparison of unsupervised pre-training, multi-task learning, and various forms of multi-task pre-training.

Results

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).