SFU NatLangLab

CMPT 413/713: Natural Language Processing

Pretraining Language Models

Spring 2025

2025-02-24

Some slides adapted from Stanford CS224n and Anoop Sarkar

Pretraining and task-specific fine-tuning

Pretraining

- Big pile of unlabeled text data!
- Lots of resources to train!

Helps to build

- Useful representations of language
- Provide good initial parameters for downstream tasks
- Probability distributions that can be sampled from

Task-specific fine-tuning

- Annotated data specific to a task (usually small)
- Initialize with pre-trained model

LLM performance depends on

- Model architecture
- Training strategy
- Training objective
- Training data

Pretraining language models

- Model (Neural Architecture)
  - Does it use FFN, RNN (LSTM, GRU), or Transformer?
    - Is it an **encoder**-based, **decoder**-based, or **encoder-decoder** model?
  - Specifics of the neural architecture (number of layers, embedding size, etc)

- Dataset
  - What is the data that is used to pretrain the model?

- Training objective
  - What is the training objective?

- Other details
  - Tokenization: what tokenization is applied?
  - Implementation and training details?

Summary of pretrained models we looked at

Paper	Model	Dataset	Training Objective
W2V CBOW [Miklov et al, 2013]	FFN	Google News (100B words)	Masked LM (within window)
ELMo [Peters et al, 2018]	Bi-LSTM	1B Word benchmark (800M words)	Bidirectional LM
BERT [Devlin et al, 2018]	Transformer (encoder block)	BookCorpus + English Wikipedia (3.3B words)	Masked LM Next sentence prediction

Training recipe for LLMs

Pre-training
LM training on large, large amount of data

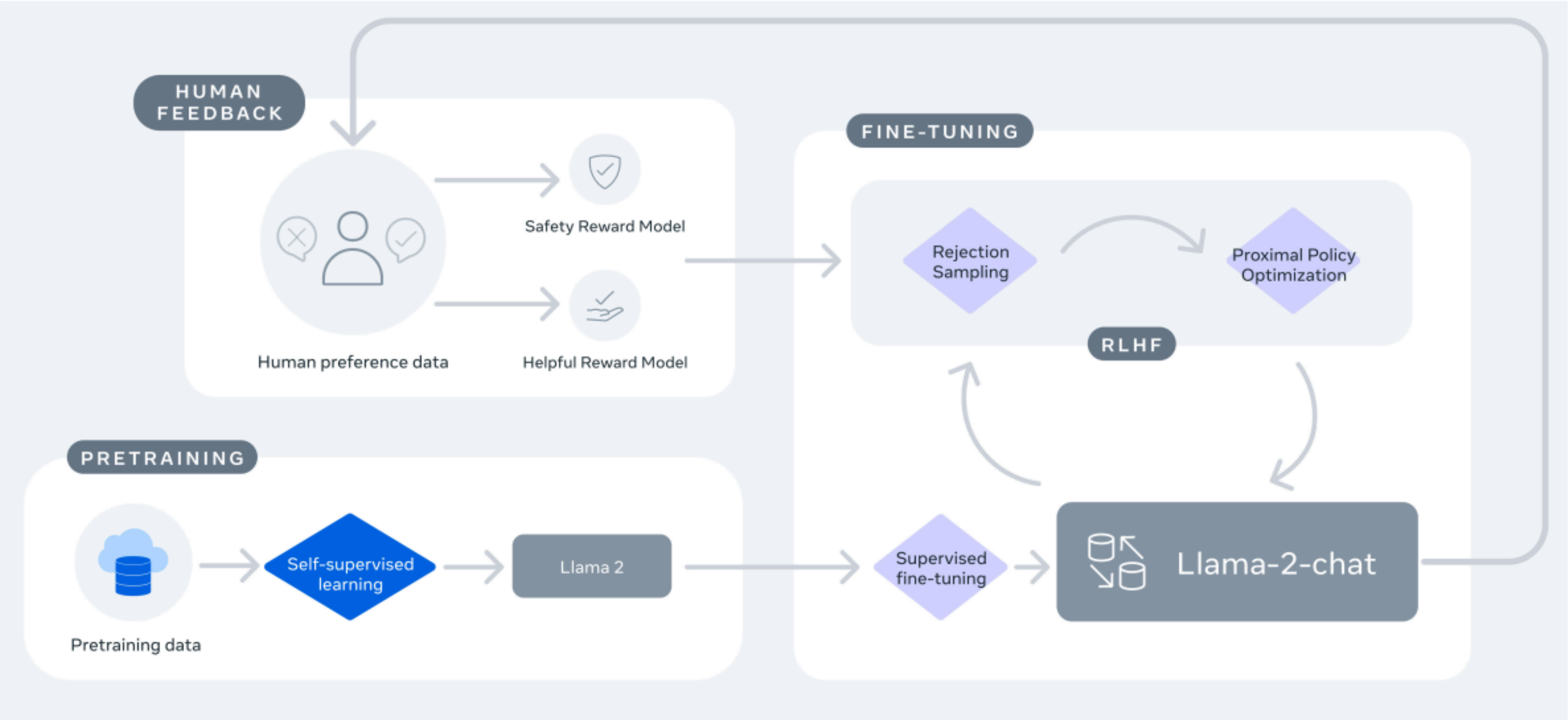
Fine-tuning
Supervised fine-tuning for instructions

Preference optimization
Align to human preferences

Pre-training can be broken into stages (mid-training)

Post-training

Training recipe for LLMs



Llama 2: Open Foundation and Fine-Tuned Chat Models [Touvron et al. 2023]

Brief History of Pre-training

1960 to 2015

- Singular Value Decomposition (1960s):
  - Take matrix $M \in |V| \times |V|$ of word co-occurrence counts
  - Use SVD to map $M = USV^T$ truncate to $|V| \times k$ initial singular values
  - Use truncated U use as word embeddings.
- Word2Vec/GloVe (2010):
  - Continuous Bag of Words (CBOW) - context words predict target word
  - Skip-gram - target word predicts each context word

Semi-supervised Sequence Learning

Andrew M. Dai
Google Inc.
adai@google.com

Quoc V. Le
Google Inc.
qvl@google.com

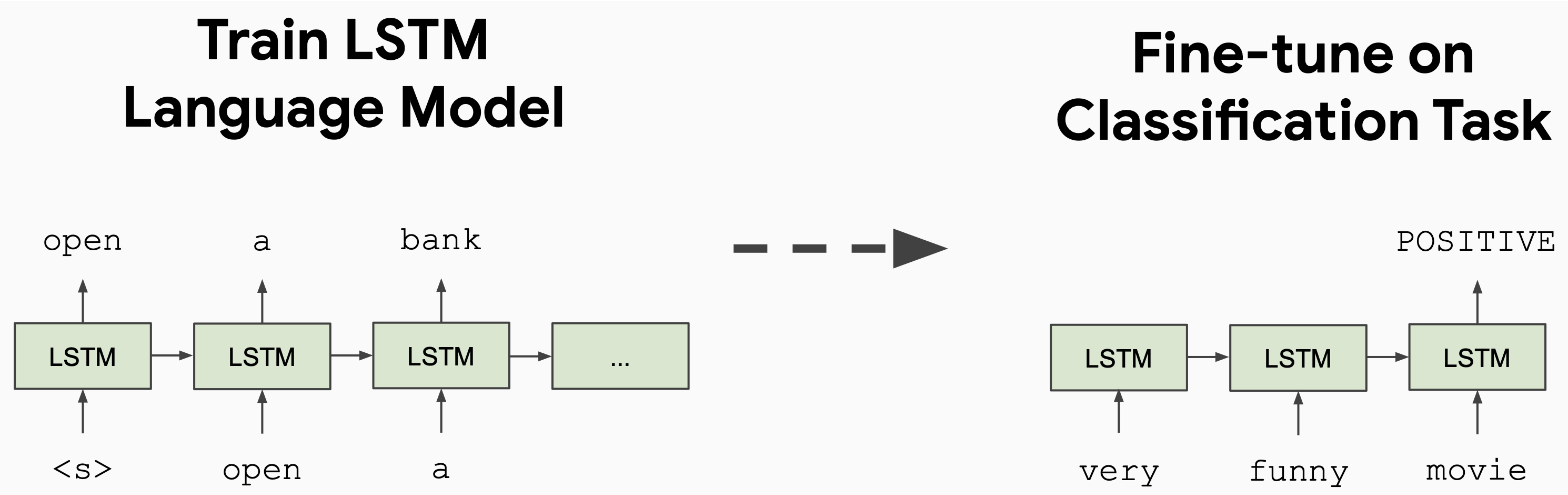


Fig from J. Devlin BERT slides https://arxiv.org/abs/1511.01432 **Nov 2015**

Deep contextualized word representations

ELMO

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

https://arxiv.org/abs/1802.05365 **Oct 2017**

ELMO

Fig from J. Devlin BERT slides

https://arxiv.org/abs/1802.05365

GPT models

GPT

- *Improving language understanding by generative pre-training* [Radford et al, 2018]
- Large language model with transformers with supervised fine-tuning
  - different model for each task
- Trained on BooksCorpus (800M words), 117M parameters (12 layers)

GPT-2

- *Language Models are Unsupervised Multitask Learner* [Radford et al, 2019]
- Model all tasks as sequence completion with special tokens indicating task
- Trained on WebText (40B words), 1.5B parameters (48 layers)
- No fine-tuning, demonstrated few-shot learning

GPT-3

- *Language Models are Few-Shot Learners* [Brown et al, 2020]
- Trained on Web+Books+Wikipedia (300B words), 175B parameters (96 layers)
- Demonstrated zero-shot and few-shot prompting abilities

GPT models (after GPT-3)

OpenAI

InstructGPT and GPT-3.5 [2022]

- Align responses to human feedback
- Instruction fine-tuning
- Reinforcement learning from human feedback
- Used in initial ChatGPT

- Supervised fine-tuning on human conversations
- Data where human will pretend to be user or AI assistant

- Human rank generated output
- Use reinforcement learning to improve generation

GPT-4 [March 2023]

- Multimodal with images and text (GPT-4V)
- Larger, better model (estimated 1.7 trillion parameters)
- Turbo [Nov 2023] - longer context (128K)

GPT-4o (omni) [May 2024]

- Multimodal with audio, images and text (GPT-4V)
- Real-time processing and generation

o1 [September 2024], o3 [January 2025?] - Reasoning

Development of Open LLMs

Closed LLMs

- **GPT (OpenAI)**
- Claude (Anthropic)
- Gemini (Google)

Open weights

- **LLaMa (Meta)**
- Mistral (Mistral AI)
- Qwen (Alibaba)
- Gemma (Google)

Open weights + data

- **OLMo (AI2)**
- DCLM
- Amber
- BLOOM
- Pythia

Open weights + partial data

- StableLM
- Zamba
- Falcon

Pre-training Transformers

Representation Learning

Preliminaries

Word structure and subword models

- NLP used to model the vocabulary in simplistic ways based on English
- Tokenize based on spaces into a sequence of "words"
- All novel words at test time were mapped to [UNK] (unknown token)

Byte Pair Encoding algorithm

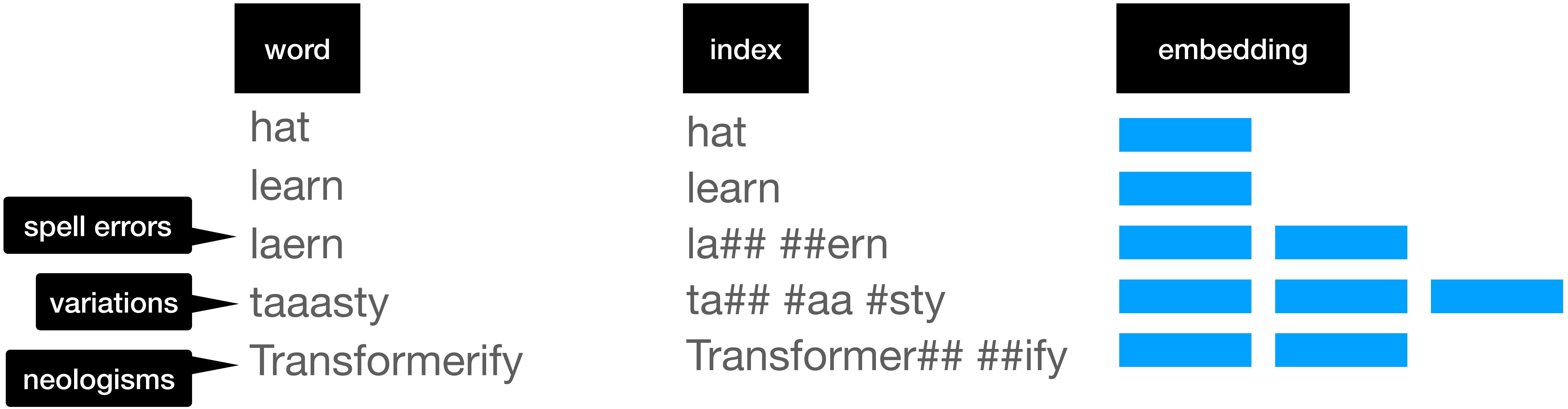
- Learn a vocabulary of parts of words (subwords)
- Vocabulary of subwords is produced before training a model on the training dataset (larger the better)
- At training and test time the vocabulary is split up into a sequence of known subwords
- Byte Pair Encoding (BPE) algorithm (takes max merges as input)
  - Init subwords with individual characters/bytes and "end of word" token.
  - Using the training data find most common adjacent subwords, merge and add to list of subwords
  - Replace all pairs of characters with new subword token; iterate until max merges

See bpe.ipynb

https://arxiv.org/abs/1508.07909

Word structure and subword models

- Common words are kept as part of the vocabulary (ignore morphology)
- Rarer words are split up into subword tokens
- In the worst case, words are split up into characters (or bytes)

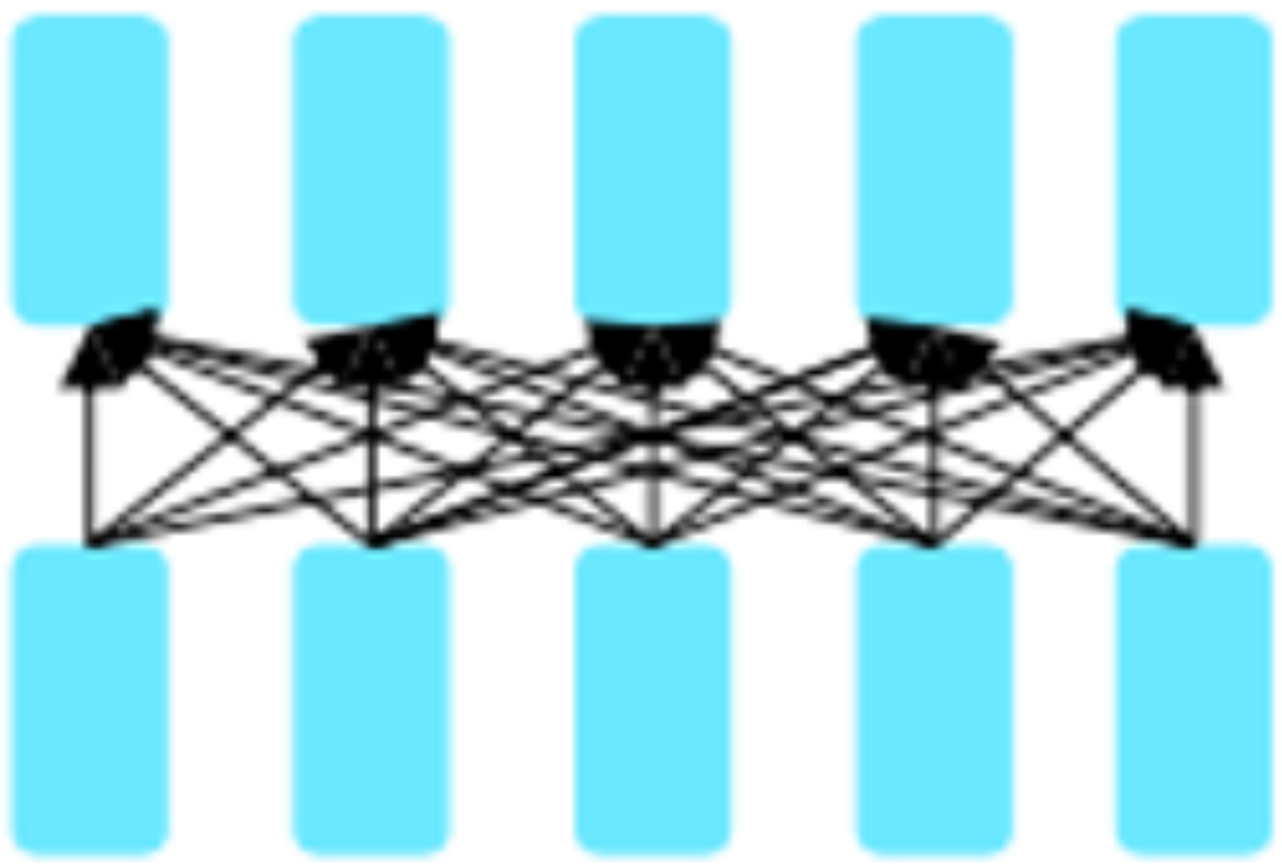


cs224n-2023-lecture9-pretraining.pdf

Transformers for pretraining

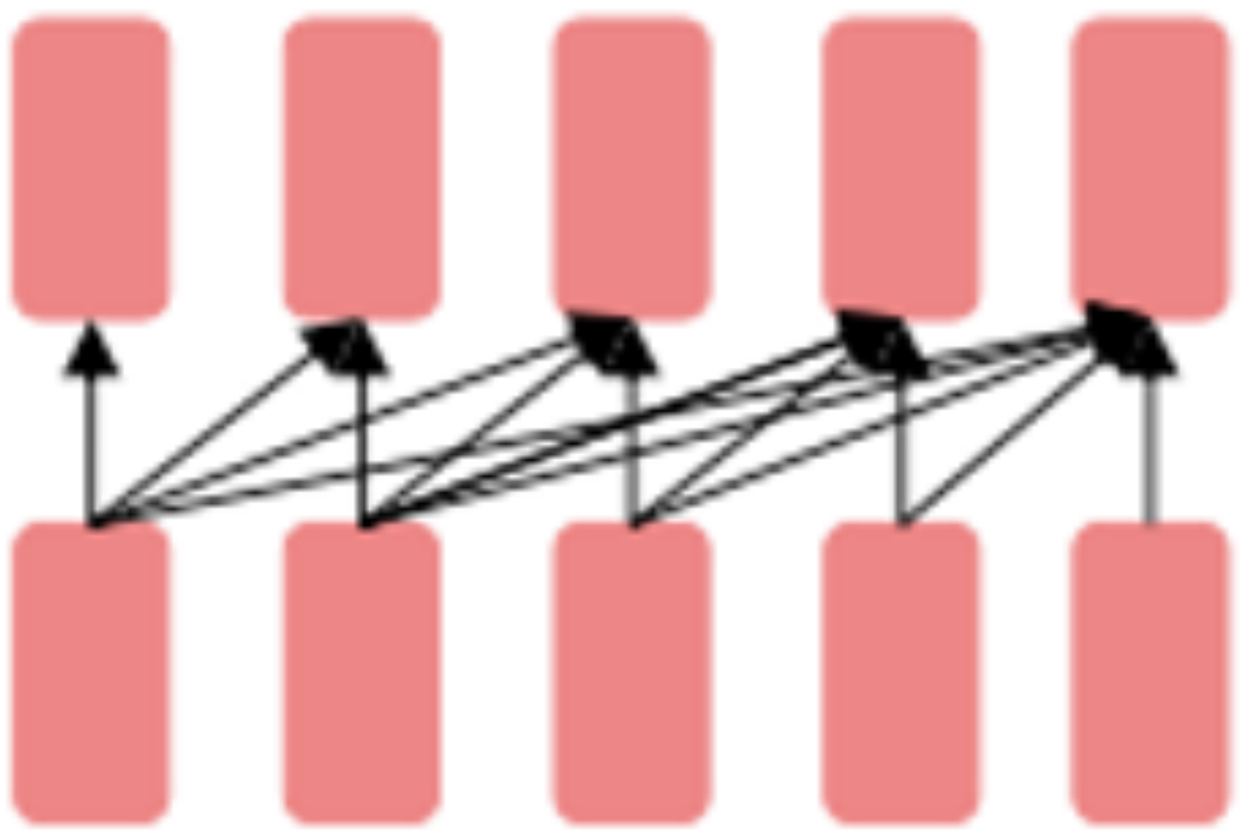
- Self-supervised Transformer based models shattered language understanding benchmarks in NLP in 2018.
- Trained on large text corpus with self-supervised objectives and then transferred.

Encoder only



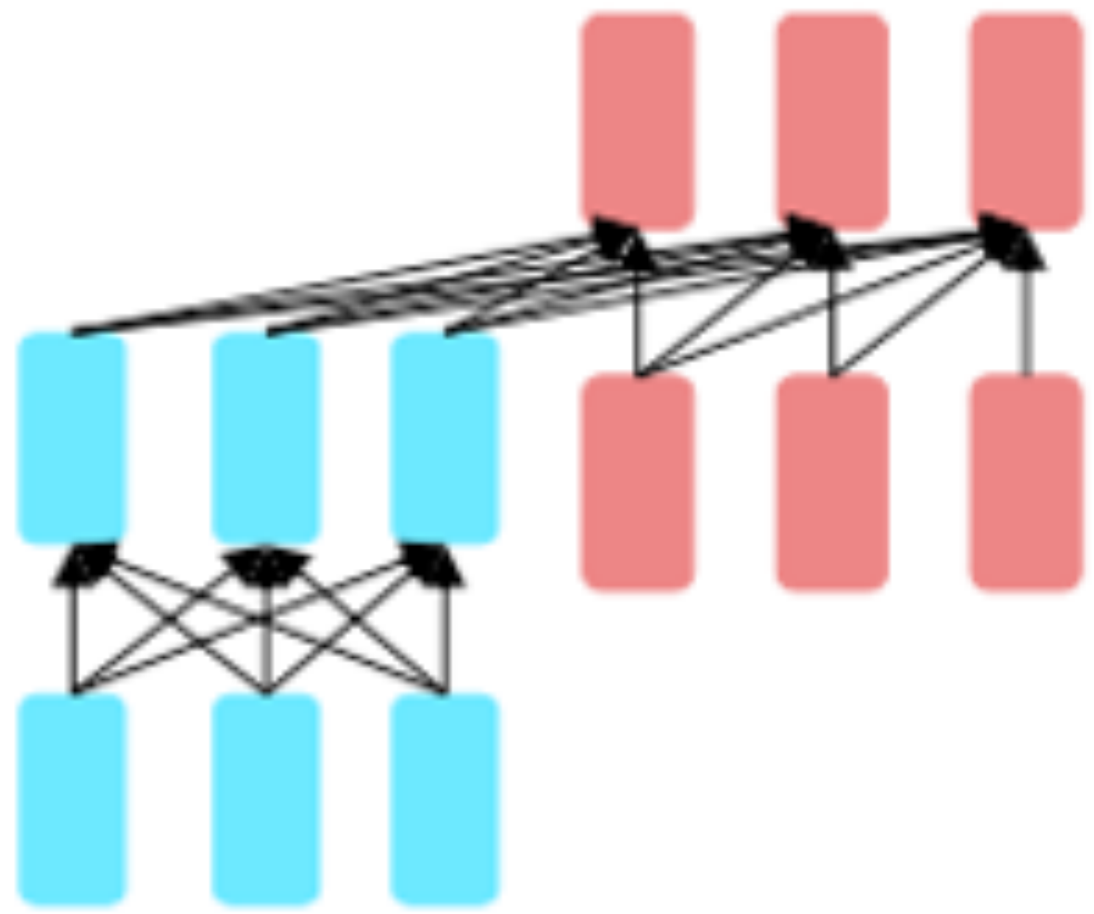
- Masked language models
- Bidirectional context
- BERT + variants (e.g. RoBERTa)

Decoder only



- Language models
- Can't condition on future words, good for generation
- GPT, LLaMa, PaLM

Encoder-Decoder

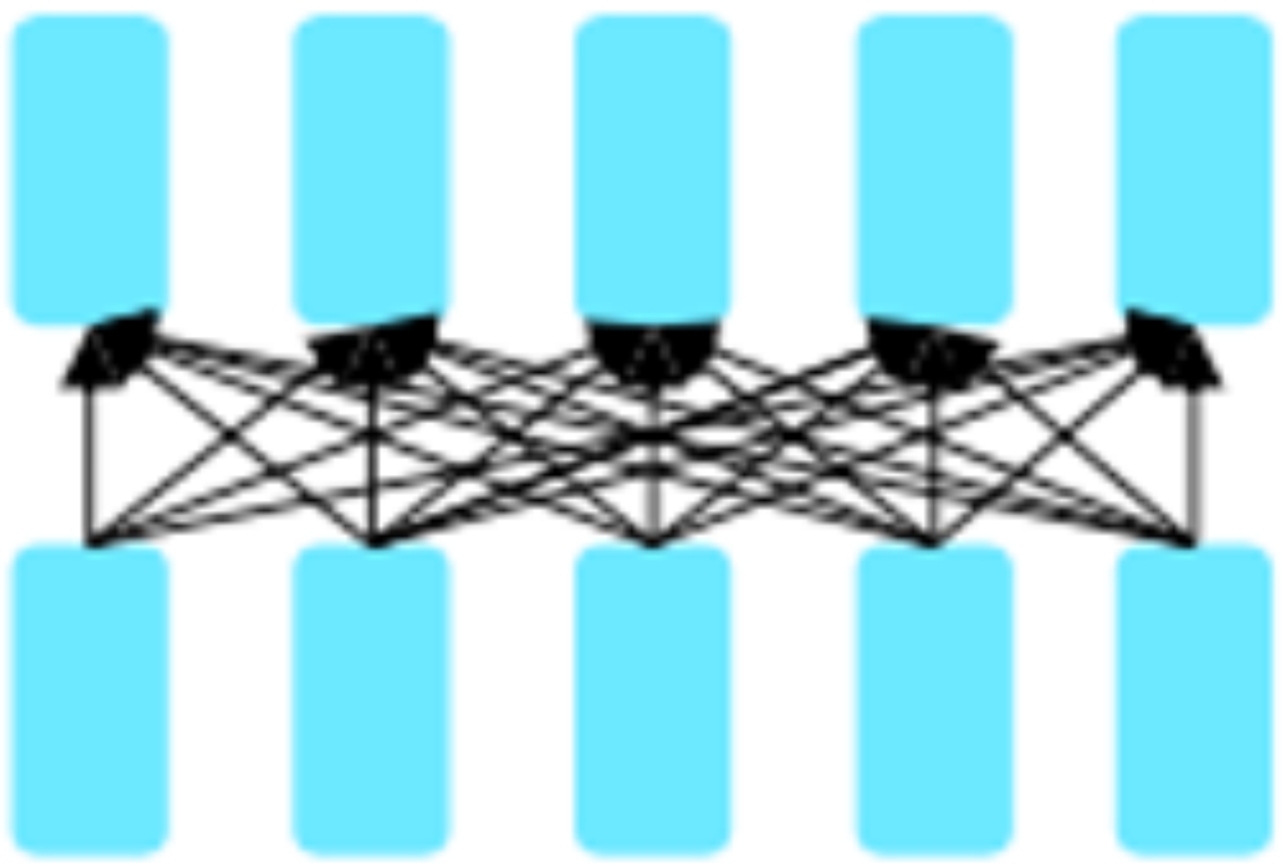


- Combine benefits of both
- Original Transformer, UniLM, BART, T5

Transformers for pretraining

- Self-supervised Transformer based models shattered language understanding benchmarks in NLP in 2018.
- Trained on large text corpus with self-supervised objectives and then transferred.

Encoder only



- Masked language models
- Bidirectional context
- BERT + variants (e.g. RoBERTa)

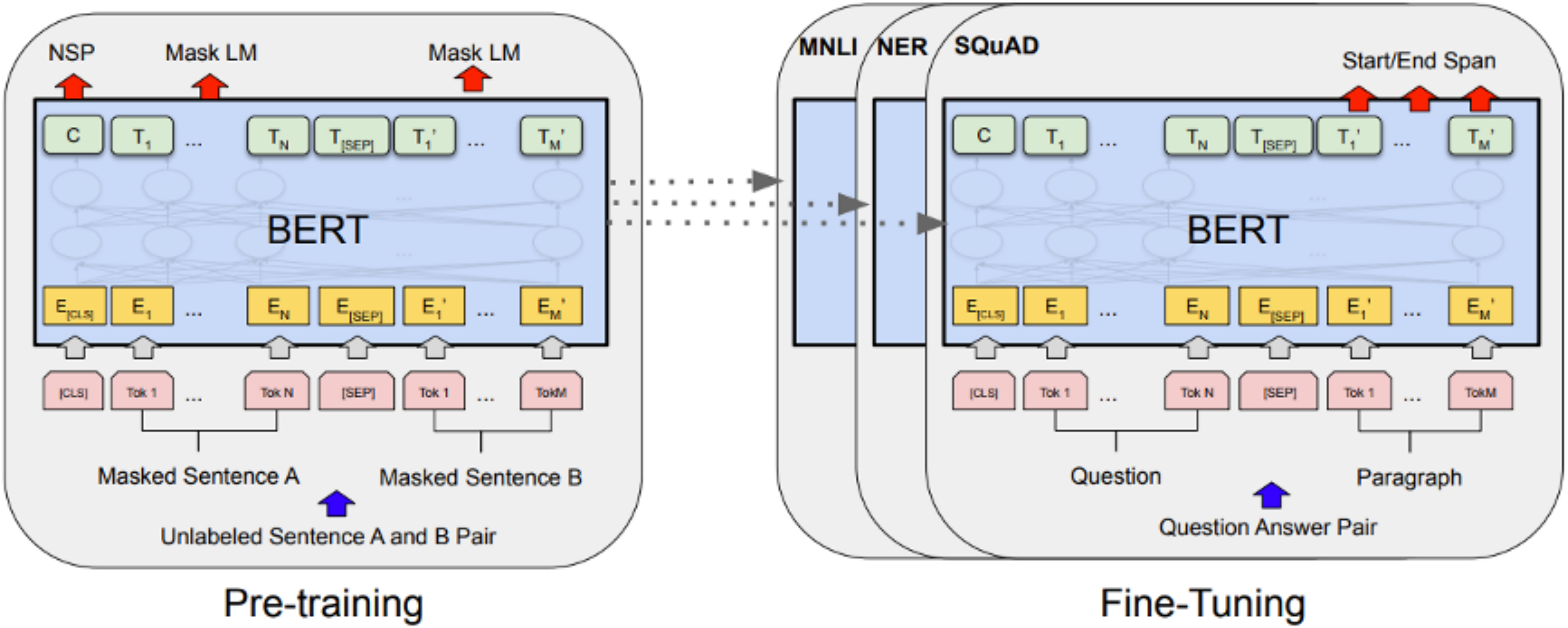
Decoder only

- Language models
- Can't condition on future words, good for generation
- GPT, LLaMa, PaLM

Encoder-Decoder

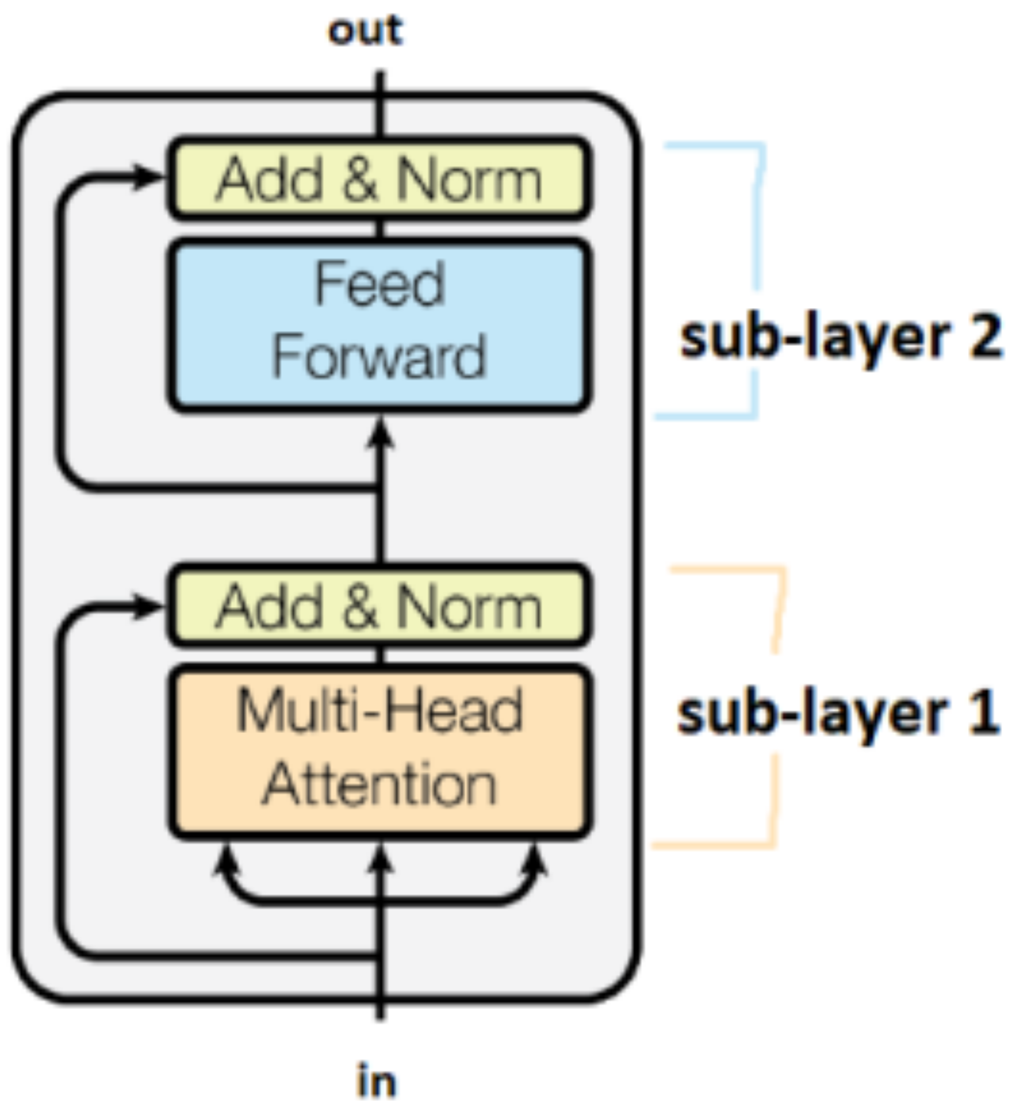
- Combine benefits of both
- Original Transformer, UniLM, BART, T5

Pre-training and fine-tuning

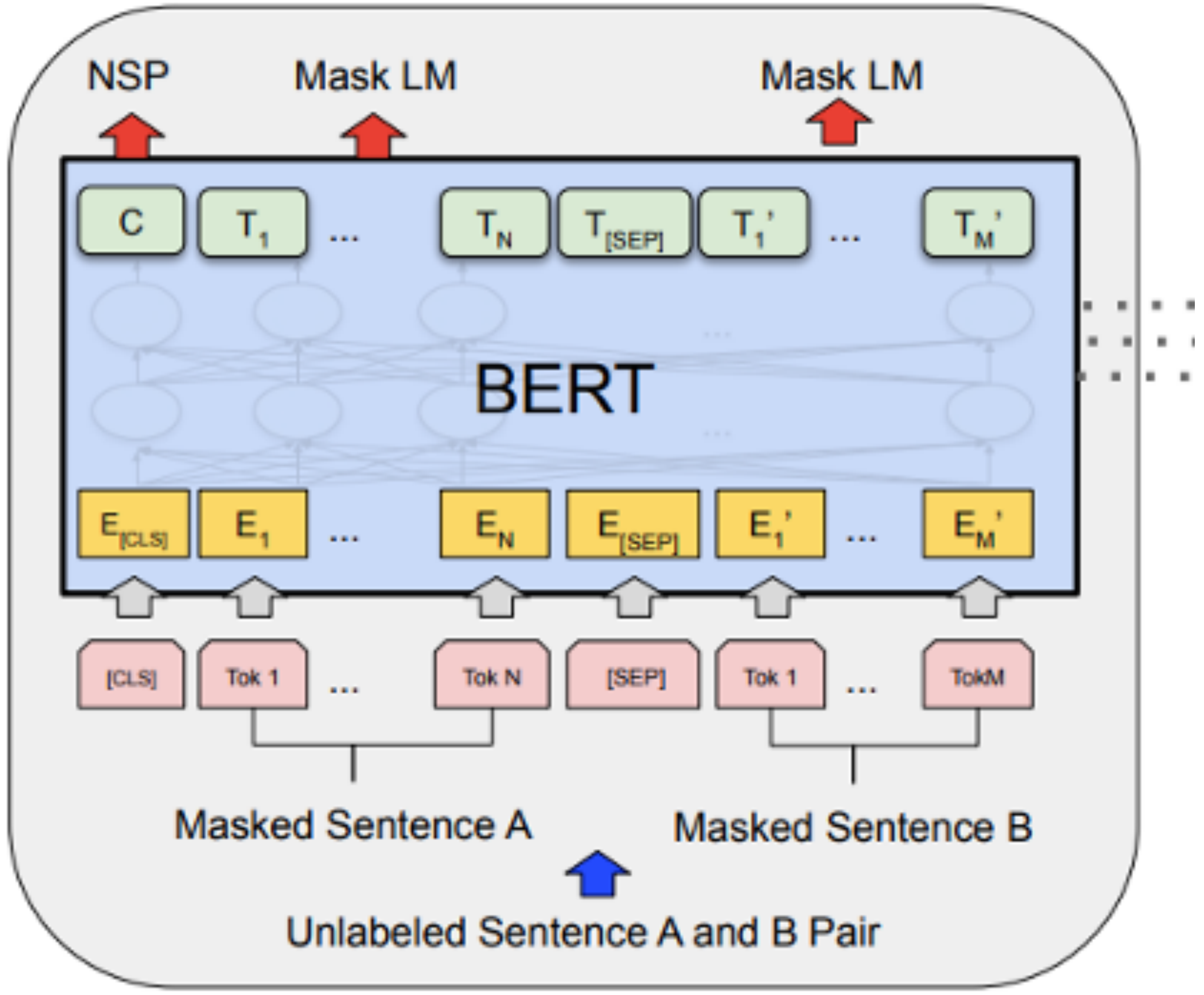


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/pdf/1810.04805.pdf

BERT



- Transformer Encoder
- Two training objectives
  - Masked Language Modeling
  - Next Sentence Prediction



Pre-training

Masked language models (MLMs)

Mask 15% of tokens

Example: `my dog is hairy`, we replace the word hairy

- 80% of time: replace word with `[MASK]` token

  `my dog is [MASK]`

- 10% of time: replace word with random word

  `my dog is apple`

- 10% of time: keep word unchanged to bias representation toward actual observed word

  `my dog is hairy`

RoBERTa

- Train with more data and for more epochs
  - Vocabulary size of 50K subword units vs 30K for BERT
  - Larger batch size and more training data
- No need for NSP

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

pretrain with **1024 V100 GPUs** for ~1 day

RoBERTa: A Robustly Optimized BERT Pretraining Approach
Liu et al, UW and Facebook, arXiv 2019

RoBERTa

- Train with more data and for more epochs
  - Vocabulary size of 50K subword units vs 30K for BERT
  - Larger batch size and more training data
- No need for NSP

Dynamic masking (masking changes)

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

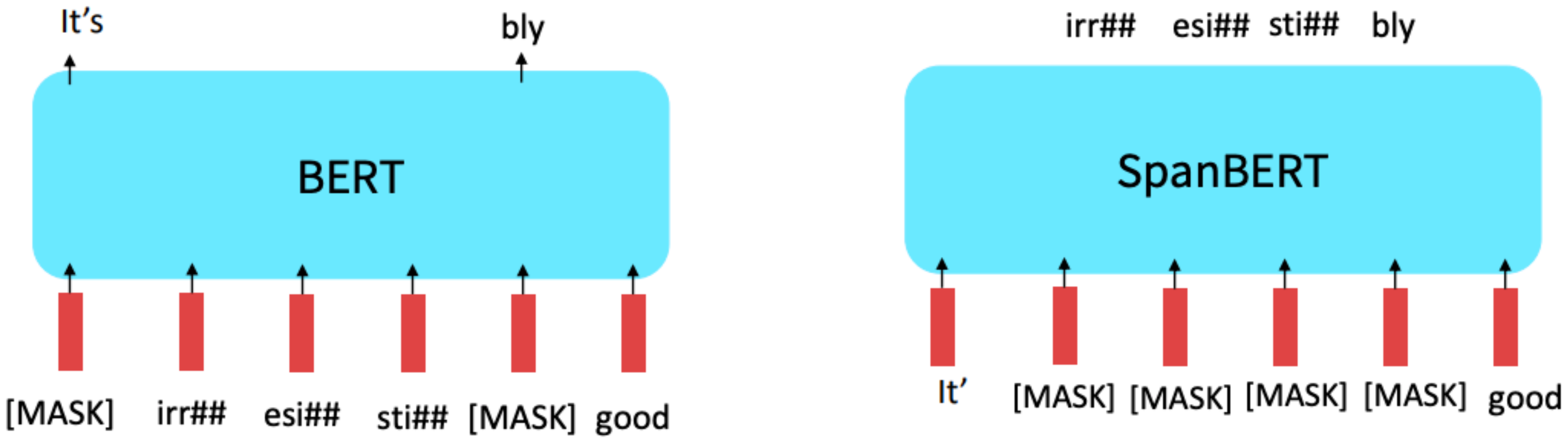
Better results with careful reimplementation.
Mean over 5 random seeds.

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

RoBERTa: A Robustly Optimized BERT Pretraining Approach
Liu et al, UW and Facebook, arXiv 2019

SpanBERT

- Mask out spans!



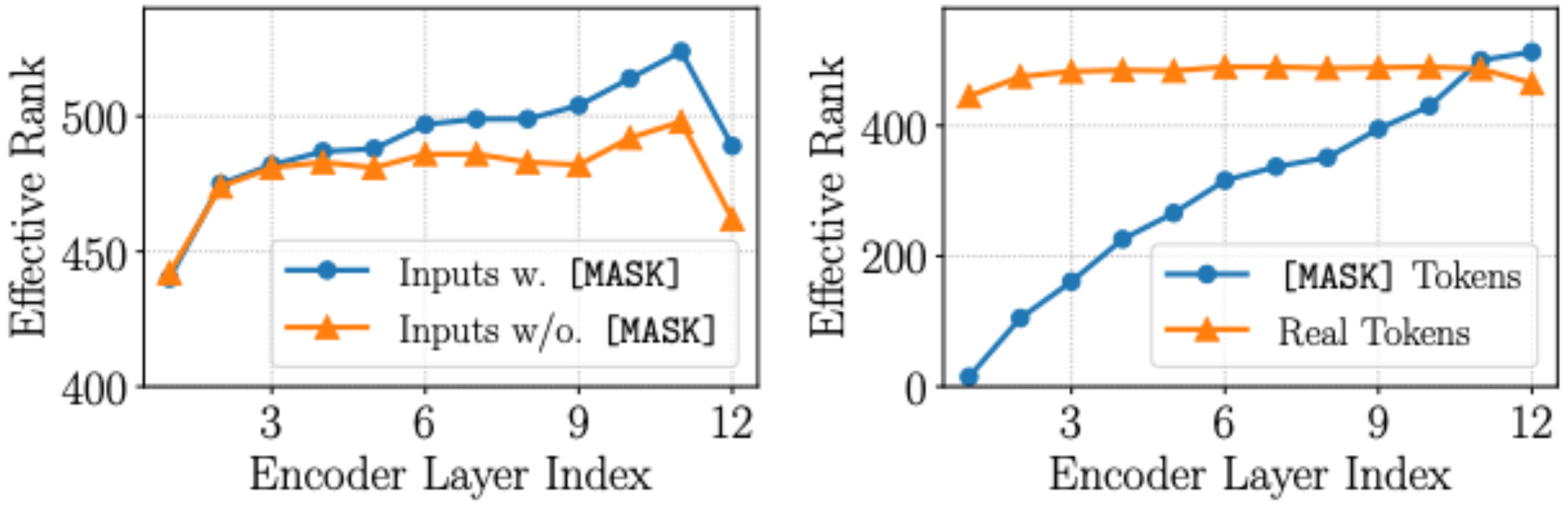
	NewsQA	TriviaQA	SearchQA	HotpotQA	Natural Questions	Avg.
Google BERT	68.8	77.5	81.7	78.3	79.9	77.3
Our BERT	71.0	79.0	81.8	80.5	80.5	78.6
Our BERT-1seq	71.9	80.4	84.0	80.3	81.8	79.7
SpanBERT	**73.6**	**83.6**	**84.8**	**83.0**	**82.5**	**81.5**

Table 2: Performance (F1) on the five MRQA extractive question answering tasks.

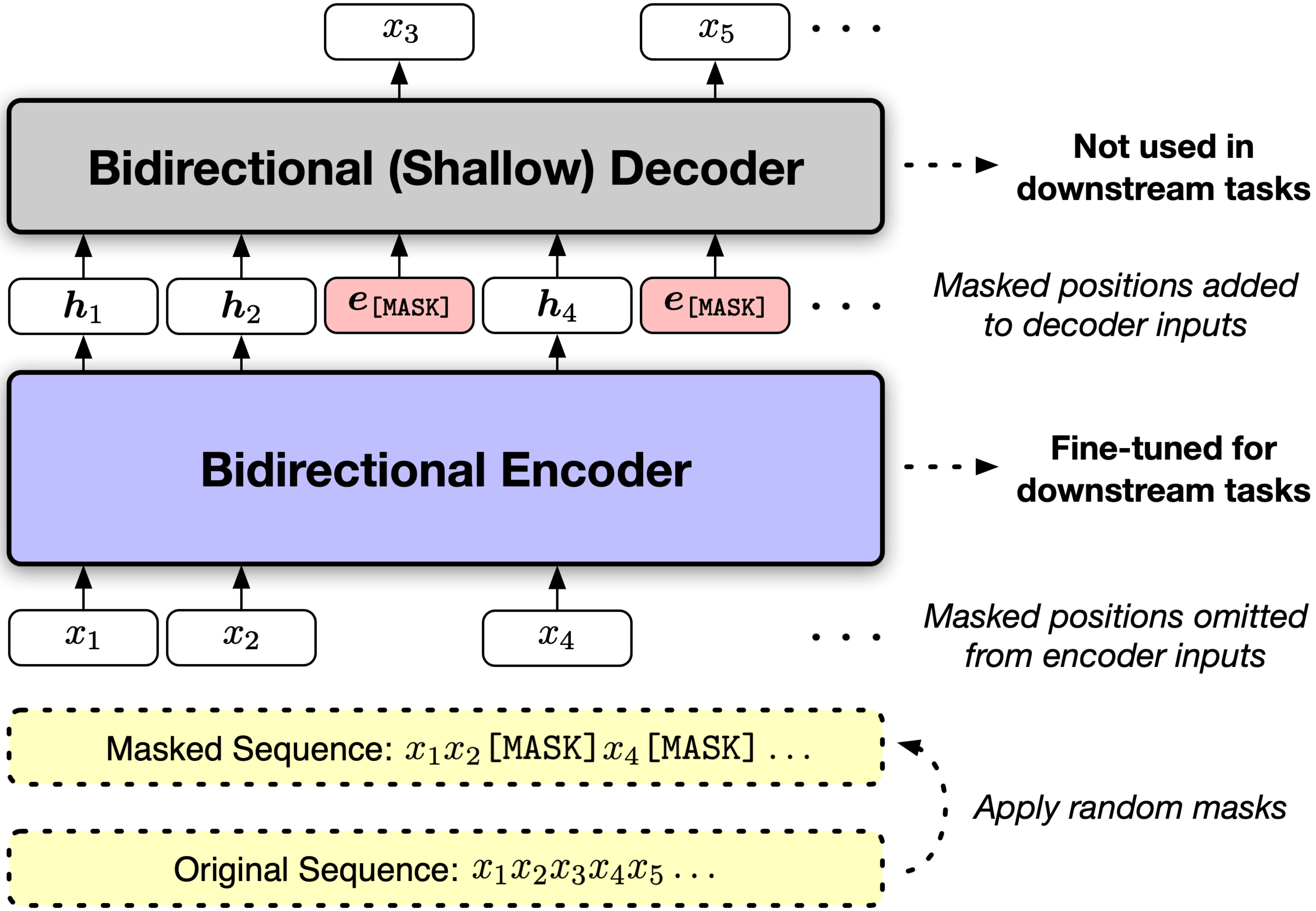
SpanBERT: Improving Pre-training by Representing and Predicting Spans
Joshi et al, TACL 2019

MAE-LM (Masked Autoencoder LM)

- [MASK] tokens are not observed in downstream tasks
- Model capacity wasted for [MASK] tokens
- Only feed non-masked tokens into encoder, have separate decoder (discarded) that predicts masked tokens



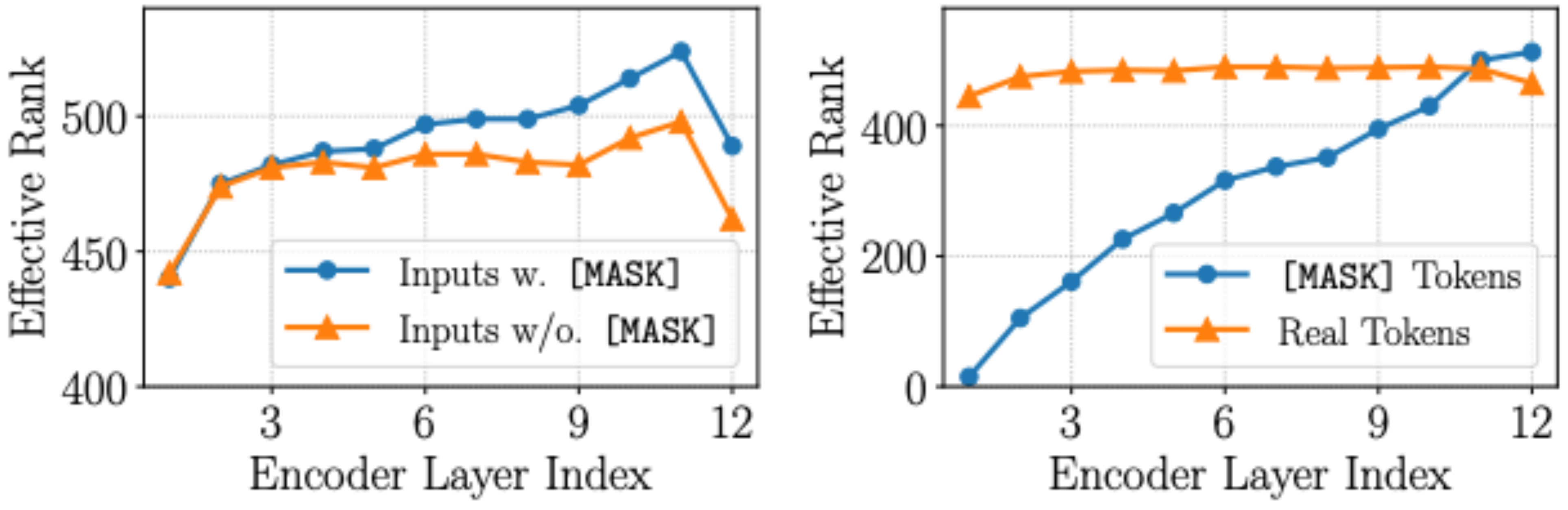
(a) (b)



Representation Deficiency in Masked Language Modeling [Meng et al. 2024]

MAE-LM (Masked Autoencoder LM)

- [MASK] tokens are not observed in downstream tasks
- Model capacity wasted for [MASK] tokens
- Only feed non-masked tokens into encoder, have separate decoder (discarded) that predicts masked tokens



(a) (b)

Table 2: Ablations evaluated with GLUE average scores. The setting of MAE-LM$_{\text{base}}$ is: enc. w/o. `[MASK]`; aligned position encoding w. relative position encoding; bi. self-attention; 4 layer, 768 dimension.

Group	Setting	GLUE
Original	MAE-LM$_{\text{base}}$	86.1
Naive	enc. w. `[MASK]` (*i.e.*, MLM)	85.2
	enc. w. `[MASK]` + dec.	85.1
Handling `[MASK]`	enc. w. `[MASK]`, dec. resets `[MASK]`	85.9
	random replace w. real token	85.1
Position Encoding	misaligned position encoding	86.0
	no relative position encoding	86.1
Decoder Attention	bi. self-attention + cross-attention	85.4
	uni. self-attention + cross-attention	85.5
	cross-attention	86.0
Decoder Size	2 layer, 768 dimension	85.8
	6 layer, 768 dimension	84.8
	4 layer, 512 dimension	85.8
	4 layer, 1024 dimension	85.5

Representation Deficiency in Masked Language Modeling [Meng et al. 2024]

ALBERT

https://arxiv.org/abs/1909.11942

Lan+ 2019

- Factorized embedding parameterization
  - Use small embedding size (128) and project to Transformer hidden size (1024) using a parameter matrix

ALBERT

https://arxiv.org/abs/1909.11942

- Cross-layer parameter sharing
  - $h^{\ell+1}$ parameters are shared with h^{ℓ}

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS
Single-task single models on dev								
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**

ALBERT

https://arxiv.org/abs/1909.11942

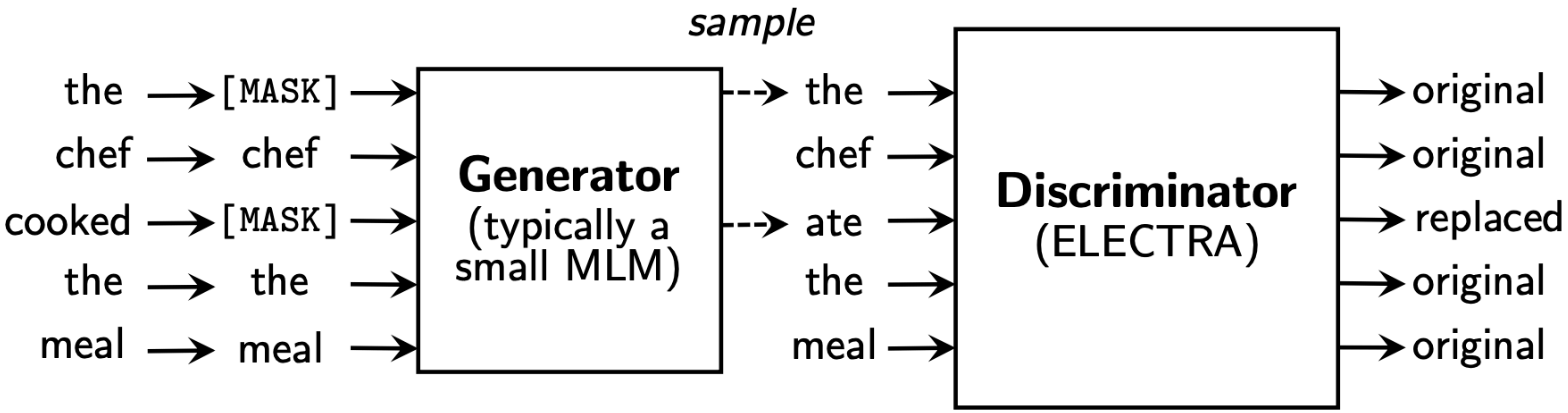
- Light on parameters; not necessarily faster than BERT

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

Discriminative training

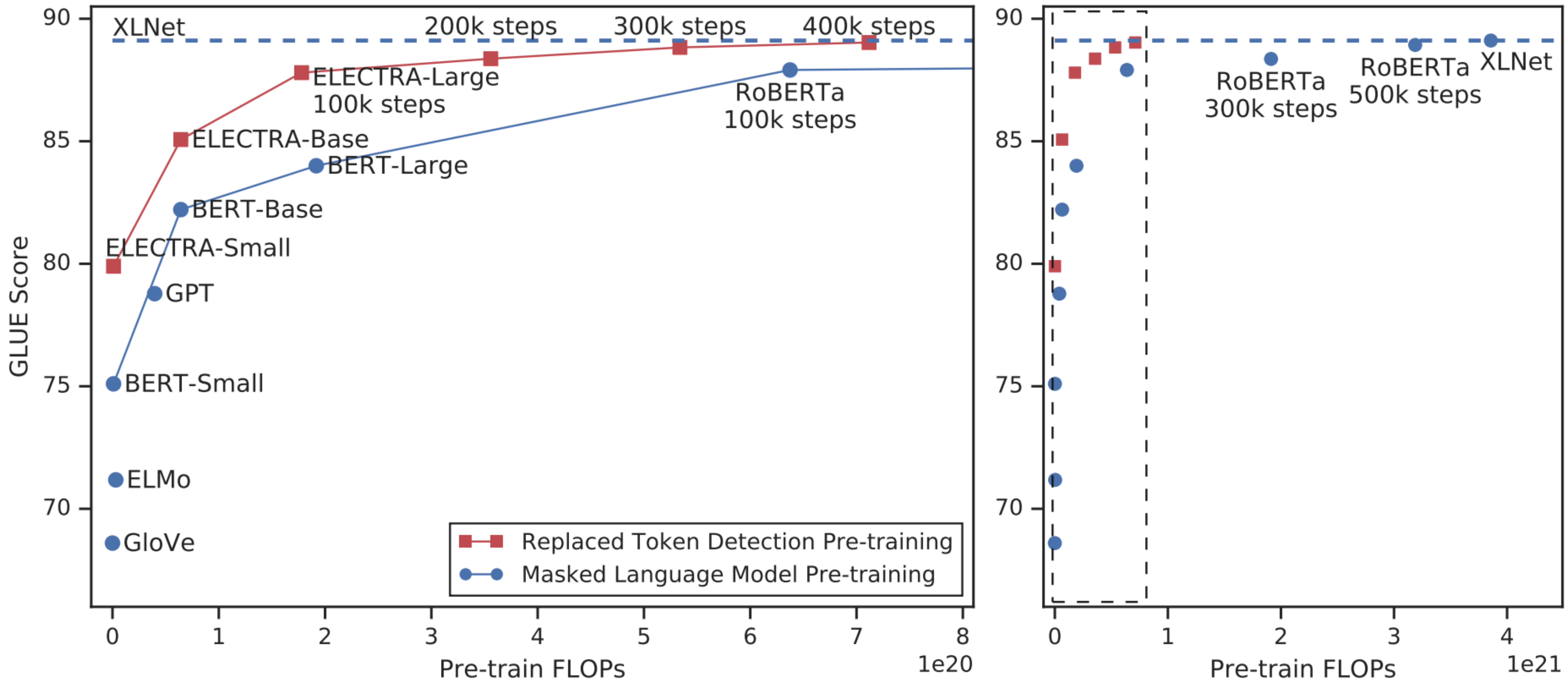
Loss is on all the training tokens vs just the masked ones, more compute efficient use of the training data

Train model to discriminate locally plausible text from real text



ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators
Clark et al, ICLR 2020

Discriminative training



ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators
Clark et al, ICLR 2020

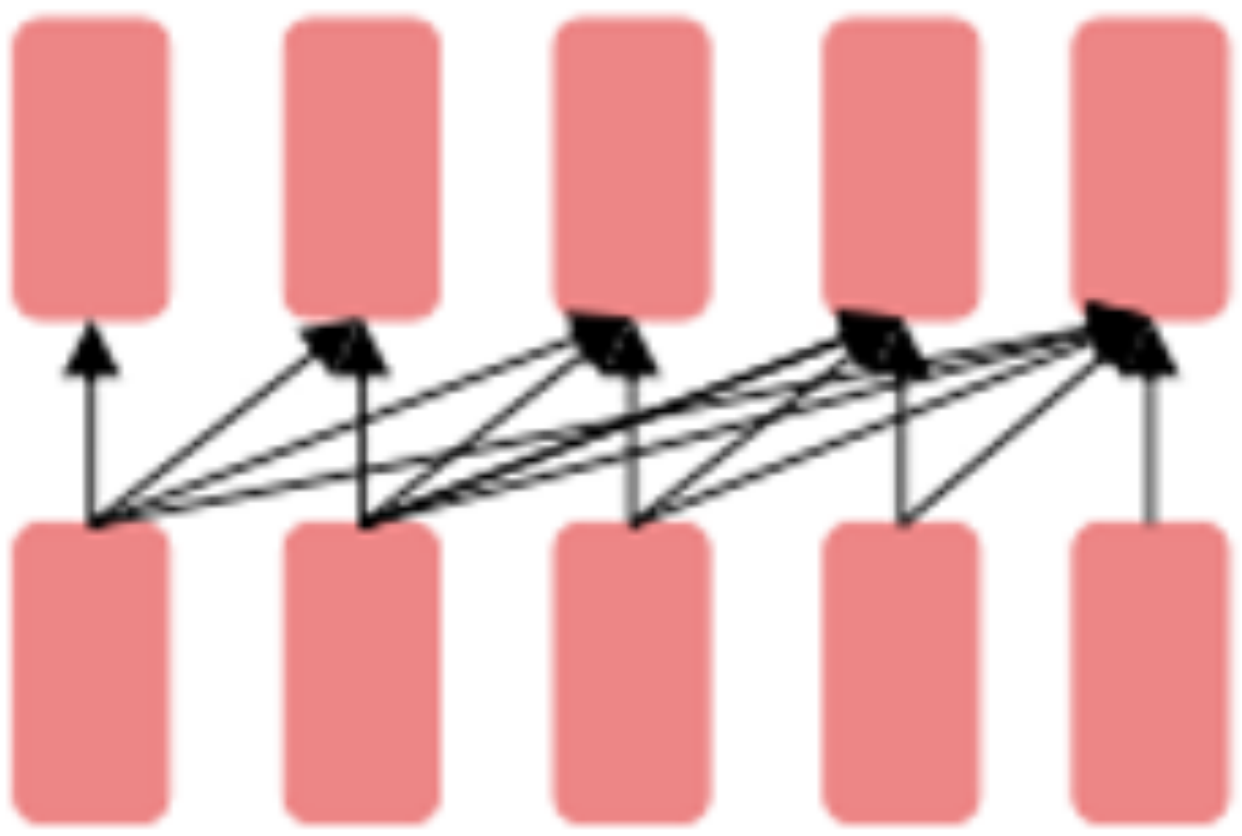
Transformers for pretraining

- Self-supervised Transformer based models shattered language understanding benchmarks in NLP in 2018.
- Trained on large text corpus with self-supervised objectives and then transferred.

Encoder only

- Masked language models
- Bidirectional context
- BERT + variants (e.g. RoBERTa)

Decoder only



- Language models
- Can’t condition on future words, good for generation
- GPT, LLaMa, PaLM

Encoder-Decoder

- Combine benefits of both
- Original Transformer, UniLM, BART, T5

Improving Language Understanding by Generative Pre-Training

GPT1

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

https://openai.com/research/language-unsupervised

Jun 2018

GPT1

Train Deep (12-layer) Transformer LM

Fine-tune on Classification Task

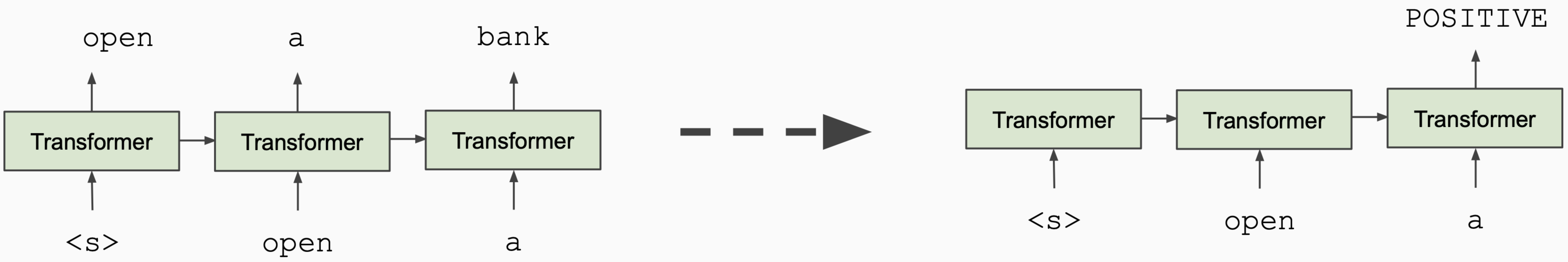


Fig from J. Devlin BERT slides

See also ULMFit: https://arxiv.org/abs/1801.06146

GPT1

Pre-training an autoregressive language model

BooksCorpus: 7K unpublished books (1B words)

- Start with a large amount of unlabeled data $\mathcal{U} = \{u_1, \ldots, u_n\}$
- Pre-training objective: Maximize the likelihood of predicting the next token
- $L_i(\mathcal{U}) = \sum_i \log P(u_i \mid u_{i-k}, \ldots, u_{i-1}; \Theta)$

$U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens

- This is equivalent to training a Transformer decoder
  - $h_0 = U\boxed{W_e} + W_p$
  - $h_\ell = \text{transformer\_block}(h_{\ell-1}) \forall \ell \in [1,n]$
  - $P(u) = \text{softmax}(h_n \boxed{W_e^T})$

n is the number of Transformer layers

W_e is the token embedding matrix

W_p is the position embedding matrix

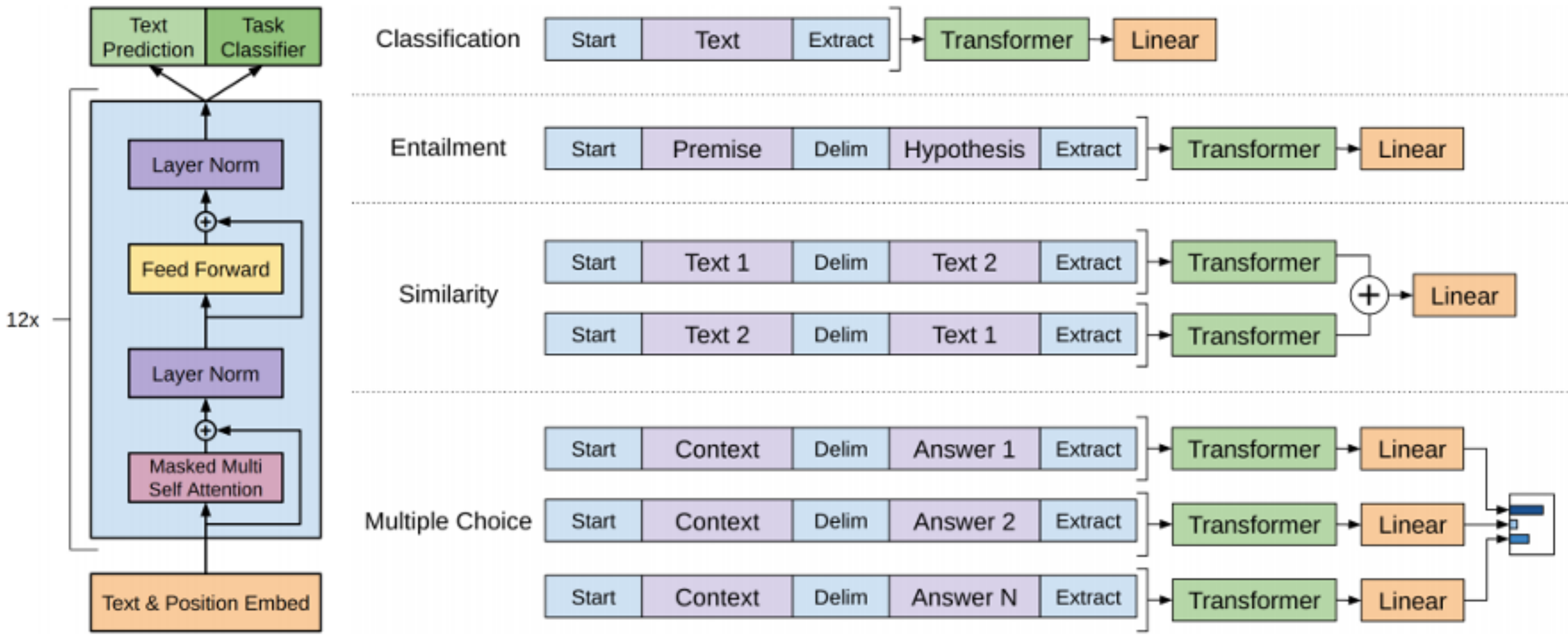
- Directionality is needed to generate a well-formed probability distribution

Dataset	Task	SOTA	GPT1
SNLI	Textual entailment	89.3	89.9
MNLI matched	Textual entailment	80.6	82.1
MNLI mismatched	Textual entailment	80.1	81.4
SciTail	Textual entailment	83.3	88.3
QNLI	Textual entailment	82.3	88.1
RTE	Textual entailment	61.7	56.0
STS-B	Semantic similarity	81.0	82.0
QQP	Semantic similarity	66.1	70.3
MRPC	Semantic similarity	86.0	82.3
RACE	Reading comprehension	53.3	59.0
ROCStories	Commonsense reasoning	77.6	86.5
COPA	Commonsense reasoning	71.2	78.6
SST-2	Sentiment analysis	93.2	91.3
CoLA	Linguistic acceptability	35.0	45.4
GLUE	Multi task benchmark	68.9	72.8

https://openai.com/research/language-unsupervised

GPT (Generative pretrained transformer)

- Unsupervised retraining: Standard language model loss
- Supervised fine-tuning: Use simple classifier (linear layer + softmax) trained to predict correct class (use combined loss)



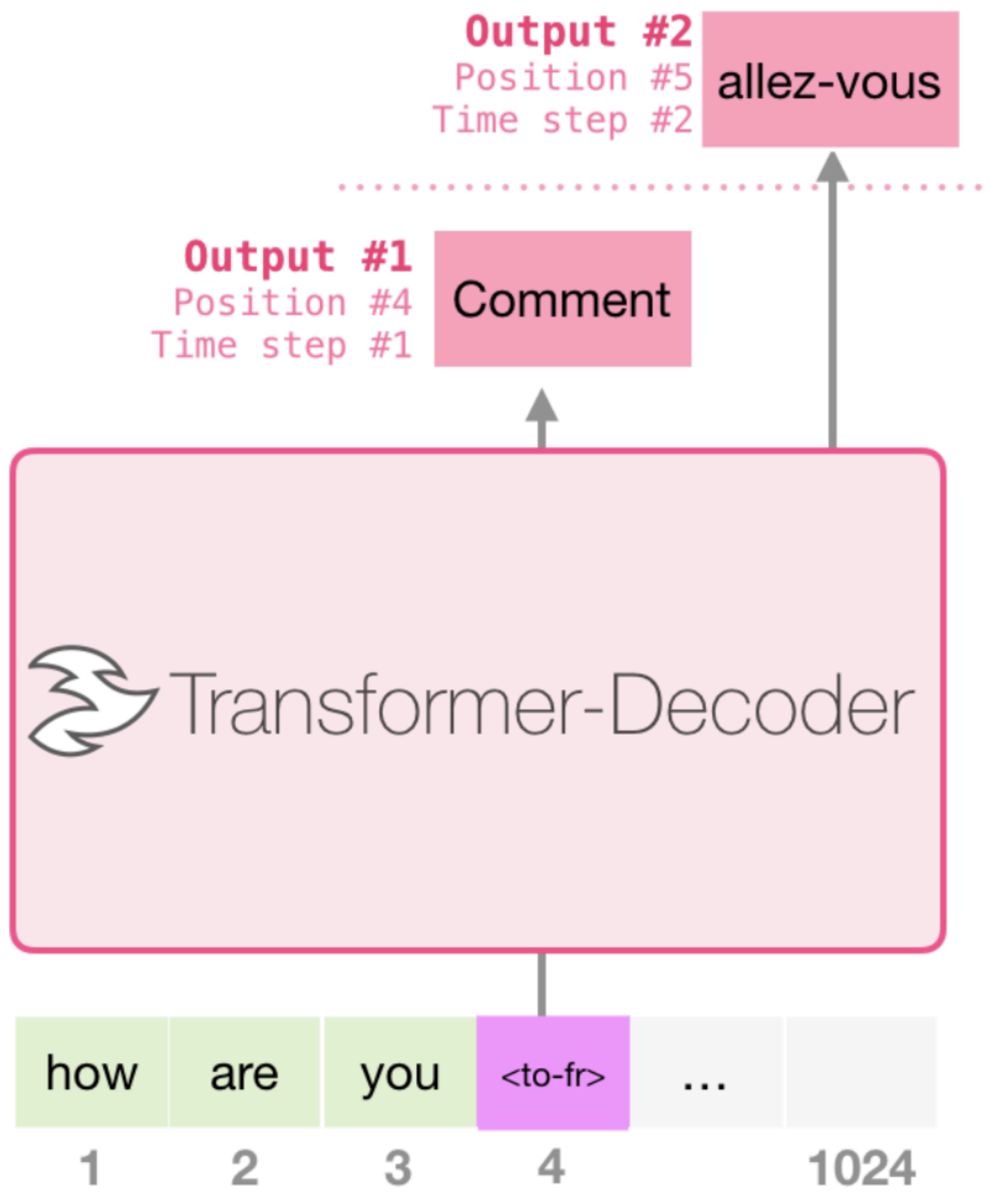
Improving language understanding by generative pre-training (Radford et al, 2018)

GPT-2

- Express all tasks a a language modelling task
- Training improvements
  - Improved initialization / additional layer normalization
  - Increased vocabulary / context /batch size

- Machine Translation

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



(figure credit: Jay Alammar http://jalammar.github.io/illustrated-gpt2/)

GPT-2

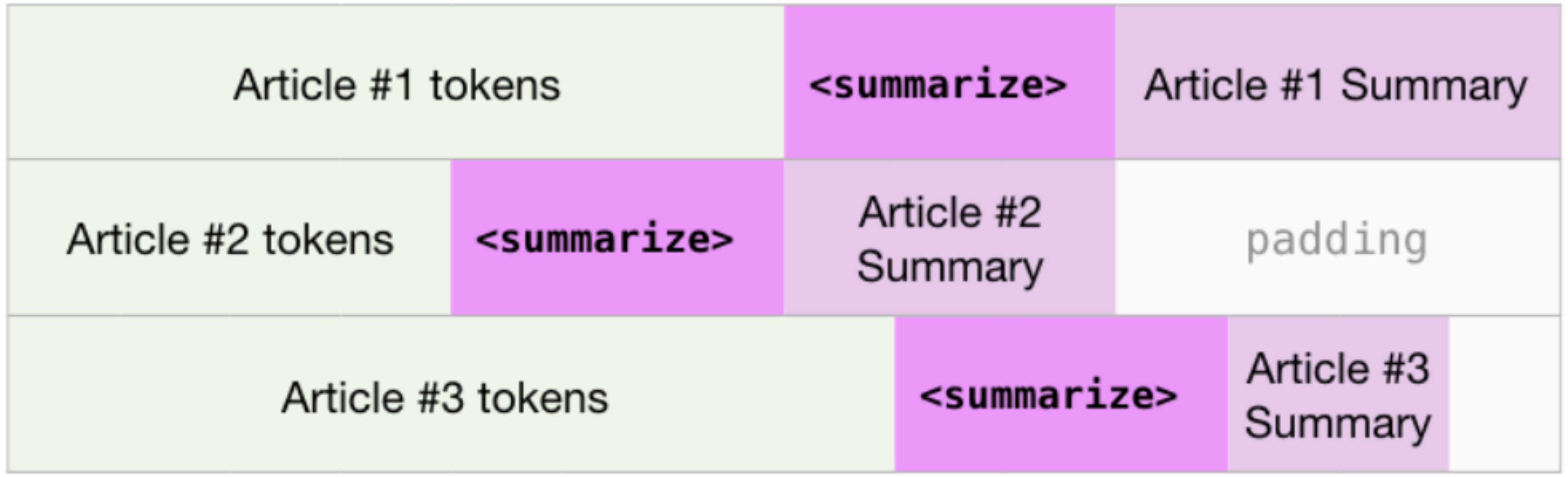
How can we use decoders for different tasks?

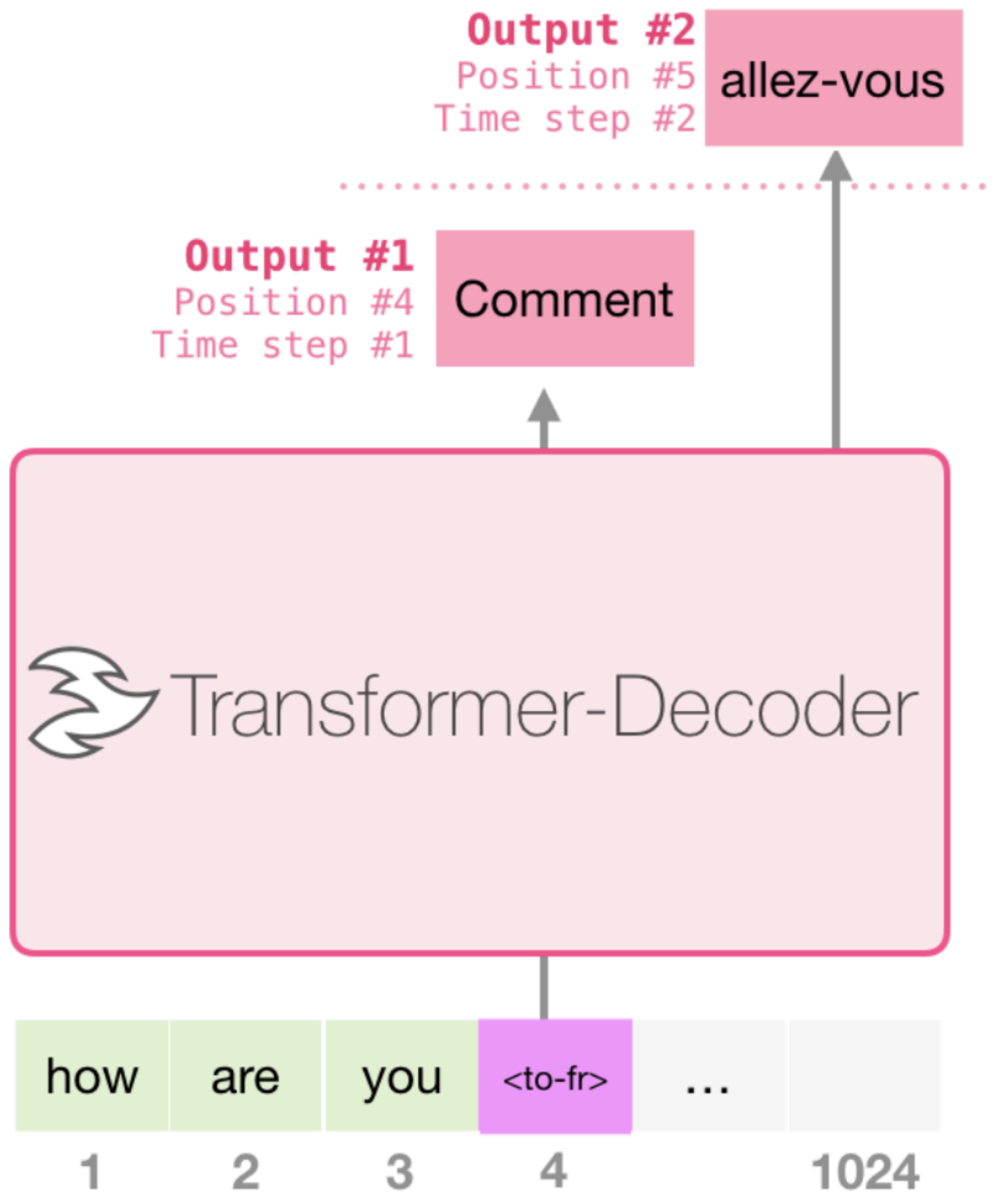
- Use special token to indicate task

Machine Translation

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				

Summarization





(figure credit: Jay Alammar http://jalammar.github.io/illustrated-gpt2/)

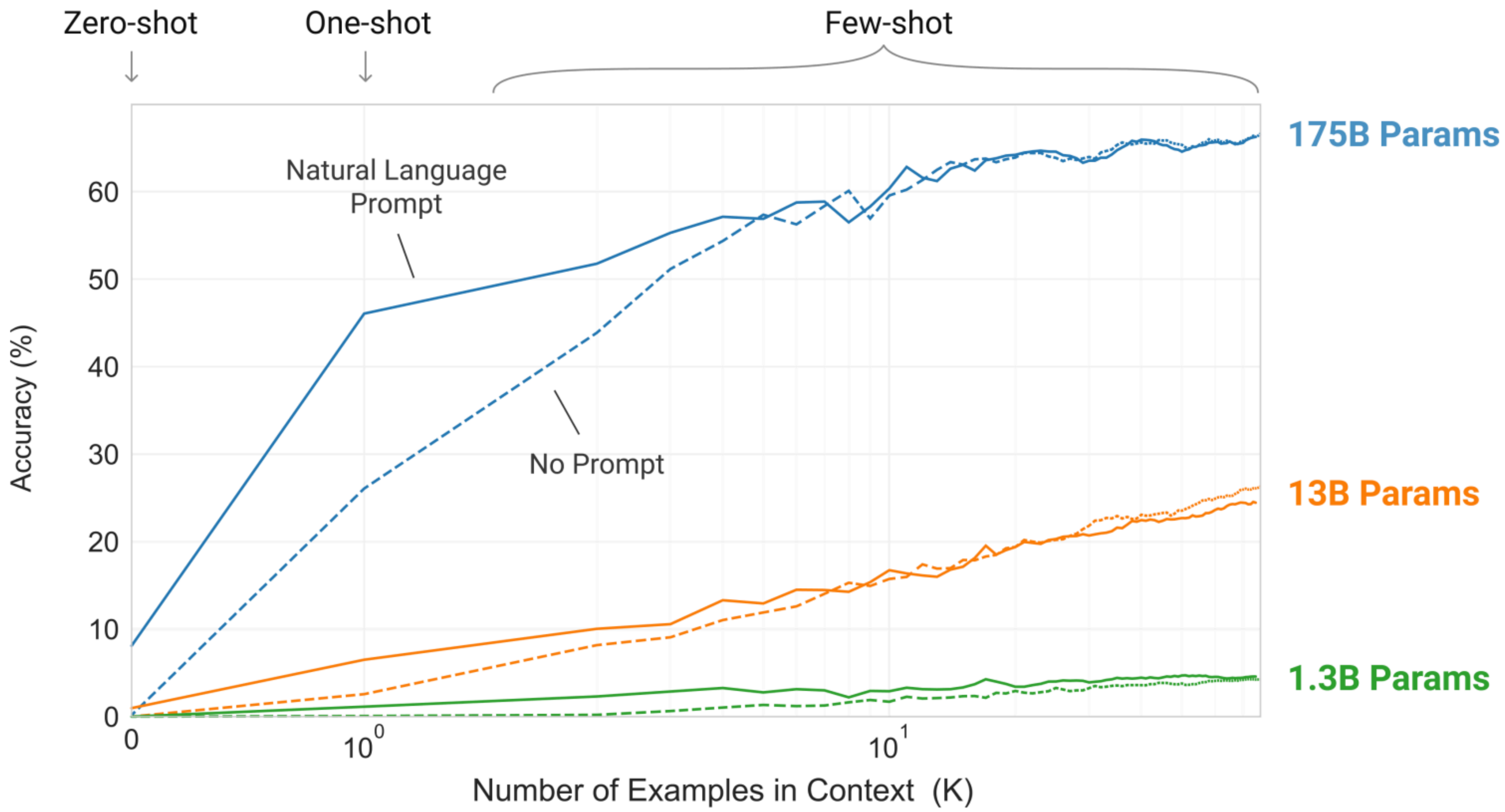
Language Models are Unsupervised Multitask Learner (Radford et al. 2019)

GPT-3: Few-shot learning

0 training examples

1 training examples

A few examples are provided at test time



Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`cheese =>`	*prompt*

One-shot

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

`Translate English to French:`	*task description*
`sea otter => loutre de mer`	*example*
`cheese =>`	*prompt*

Few-shot

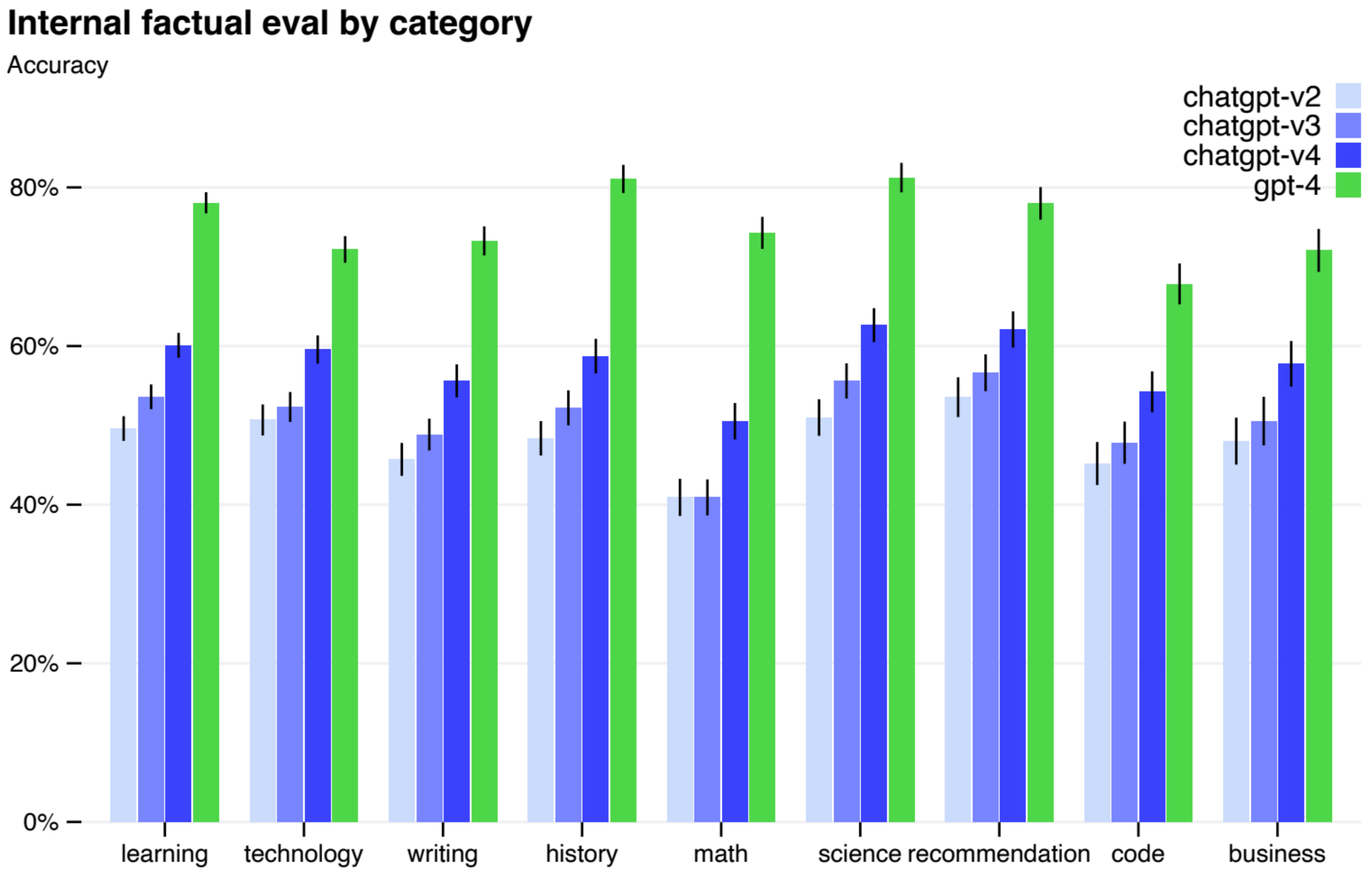
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`sea otter => loutre de mer`	*examples*
`peppermint => menthe poivrée`	
`plush girafe => girafe peluche`	
`cheese =>`	*prompt*

Language Models are Few-Shot Learners (Brown et al. OpenAI, 2020)

GPT-4

- Growing performance for ChatGPT versions

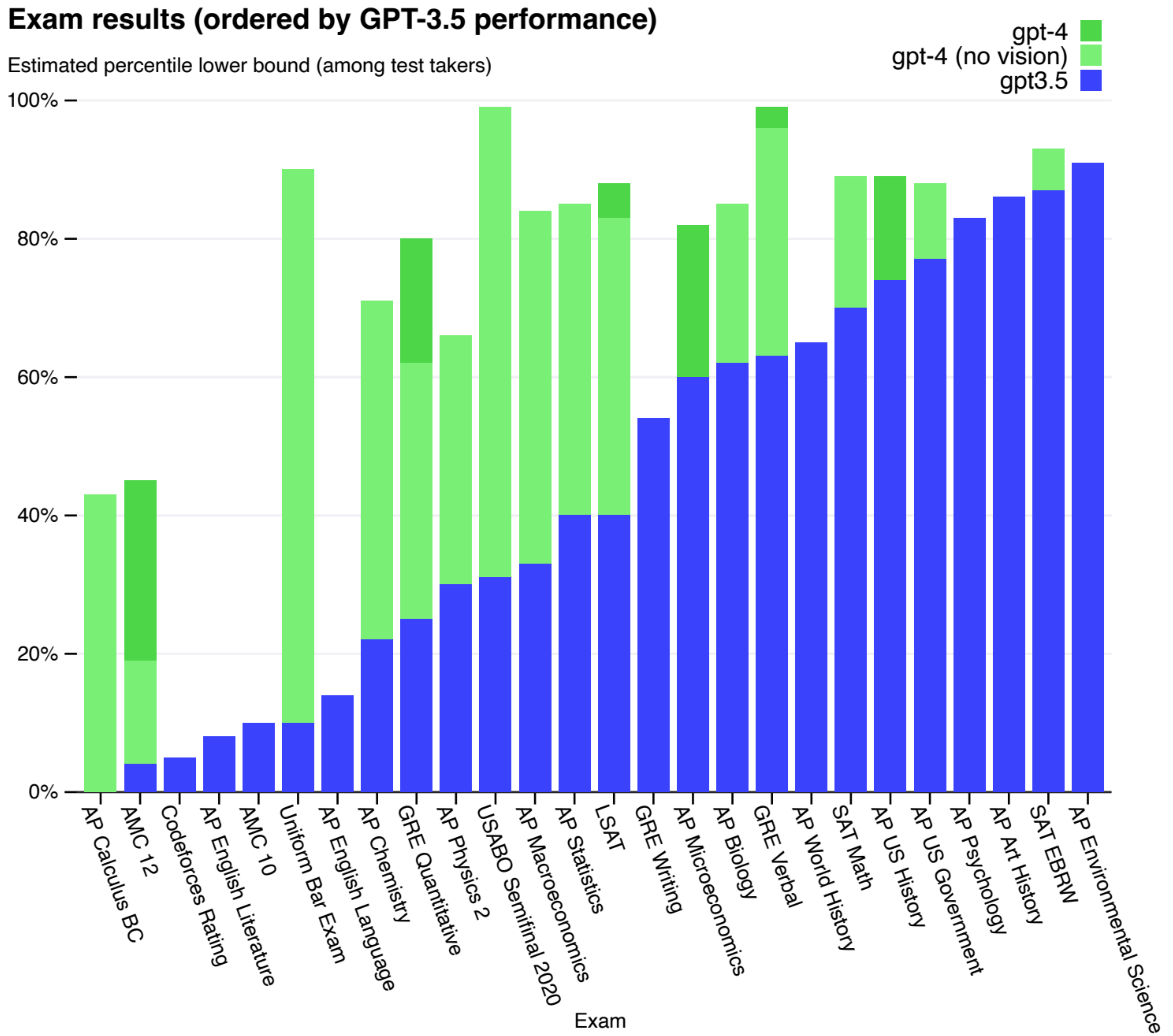


https://openai.com/research/gpt-4

GPT-4

- GPT-4 passing standardized tests

- Bar exam:
  - GPT-3.5 score in bottom 10%
  - GPT-4 score in top 10%



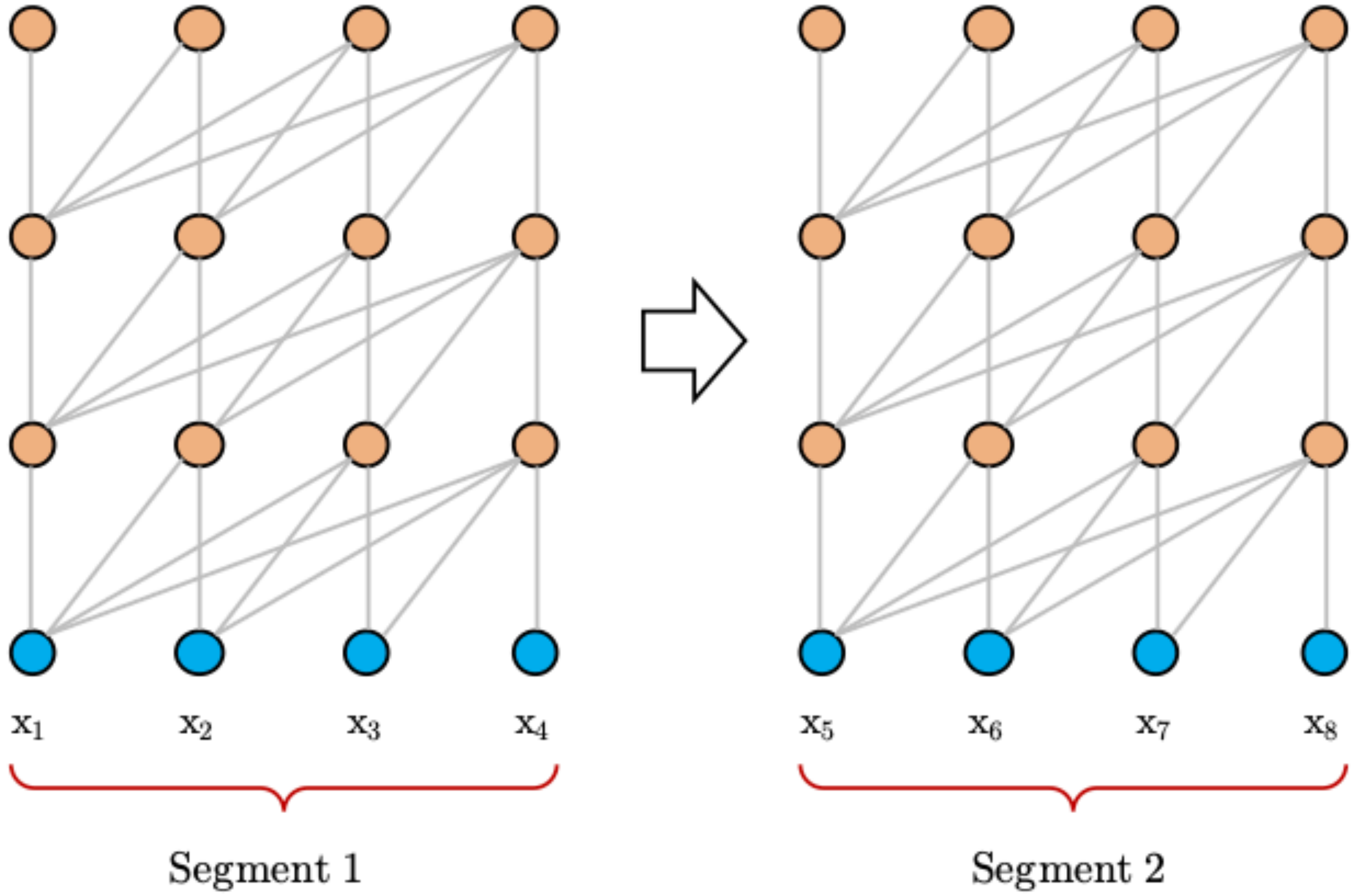
https://openai.com/research/gpt-4

Transformer-XL

https://arxiv.org/abs/1901.02860

Dai+ 2019

- Vanilla Model



(a) Train phase.

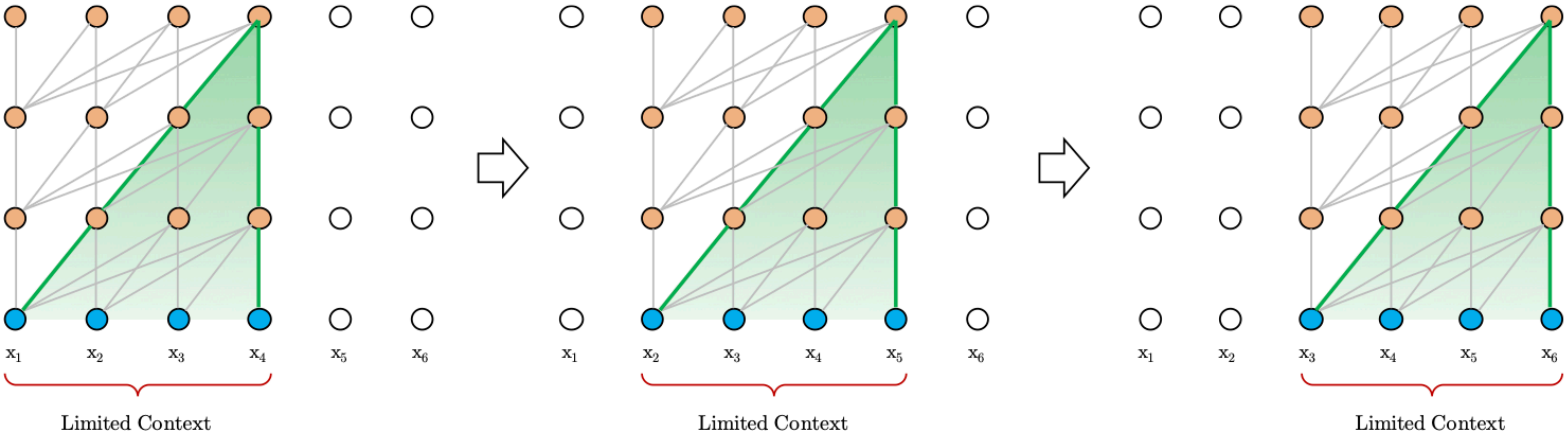
Transformer-XL

https://arxiv.org/abs/1901.02860

Dai+ 2019

Is there a better way to allow for long context?

- Vanilla Model



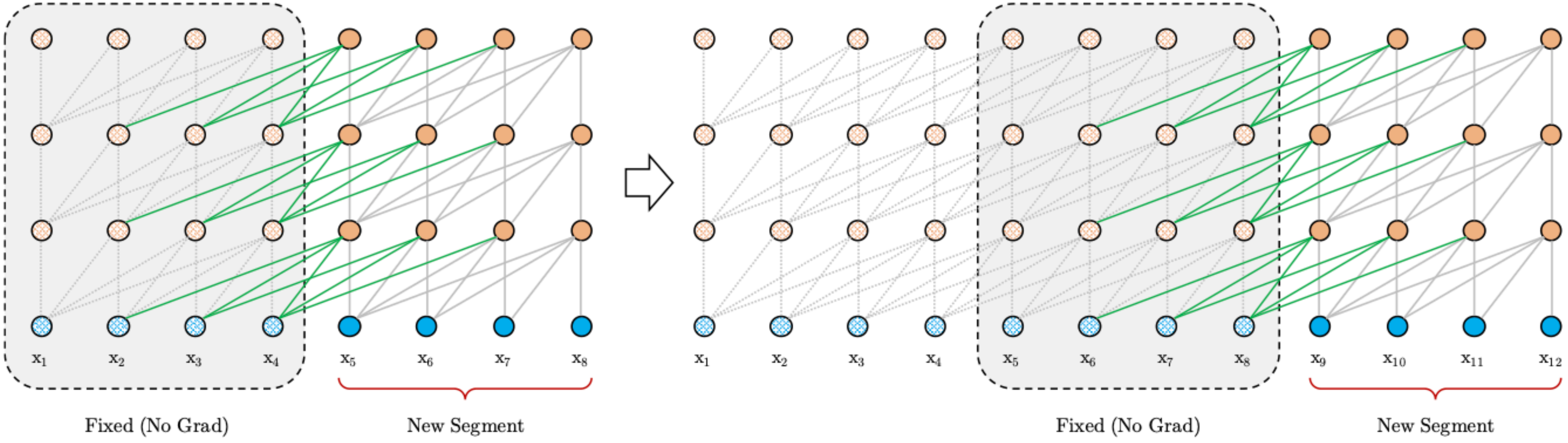
(b) Evaluation phase.

Transformer-XL

https://arxiv.org/abs/1901.02860

Dai+ 2019

- Autoregressive LM (different from GPT)
- segment level recurrence (reuse states) + relative positional embeddings



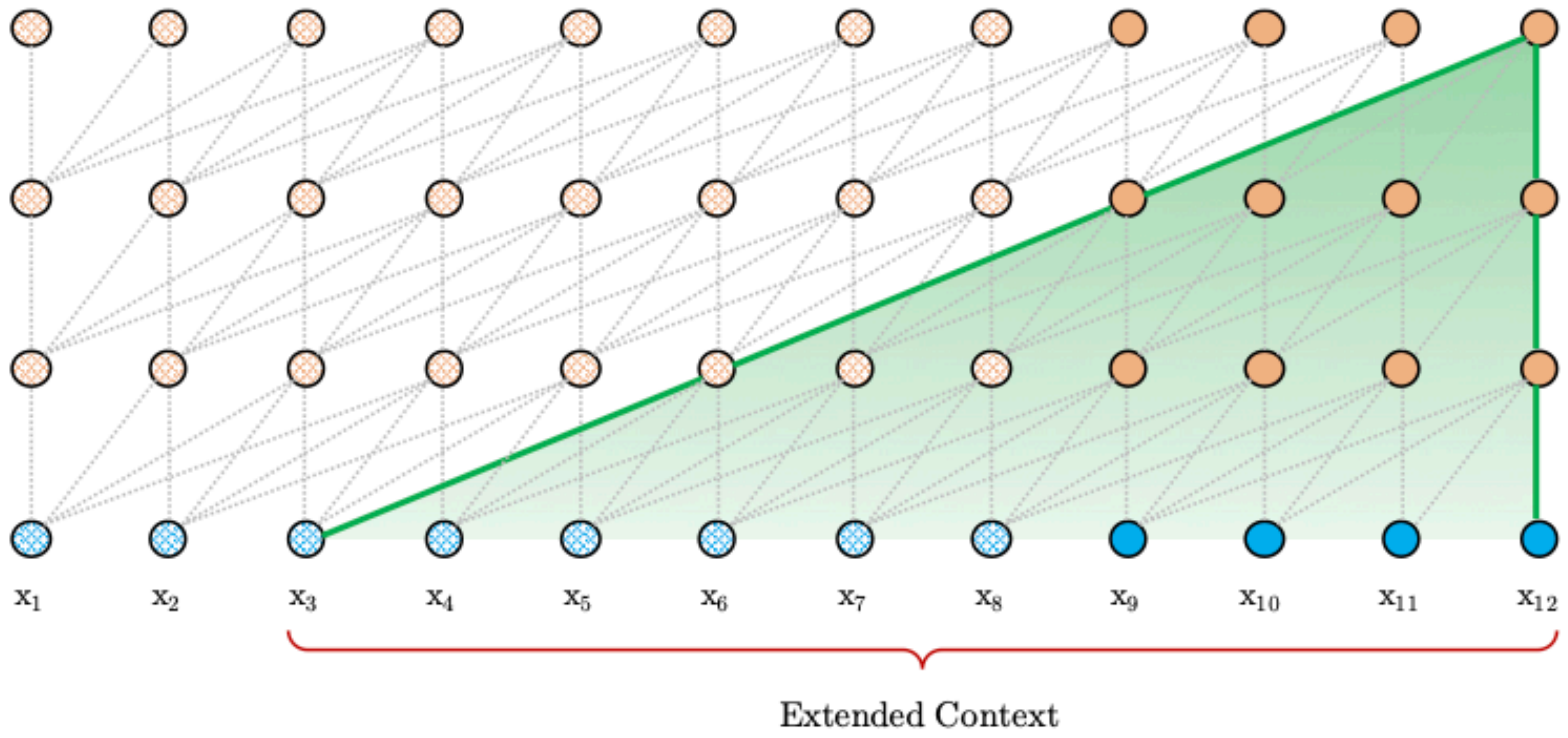
(a) Training phase.

Transformer-XL

https://arxiv.org/abs/1901.02860

Dai+ 2019

- Autoregressive LM (different from GPT)



(b) Evaluation phase.

XLNet

https://arxiv.org/abs/1906.08237

Yang+ 2019

- Autoregressive model for masked language modelling
  - Uses permutations (factorization order) to provide context
  - Allows for context from both sides through permutation
  - Avoid [MASK] token that does not appear in downstream tasks

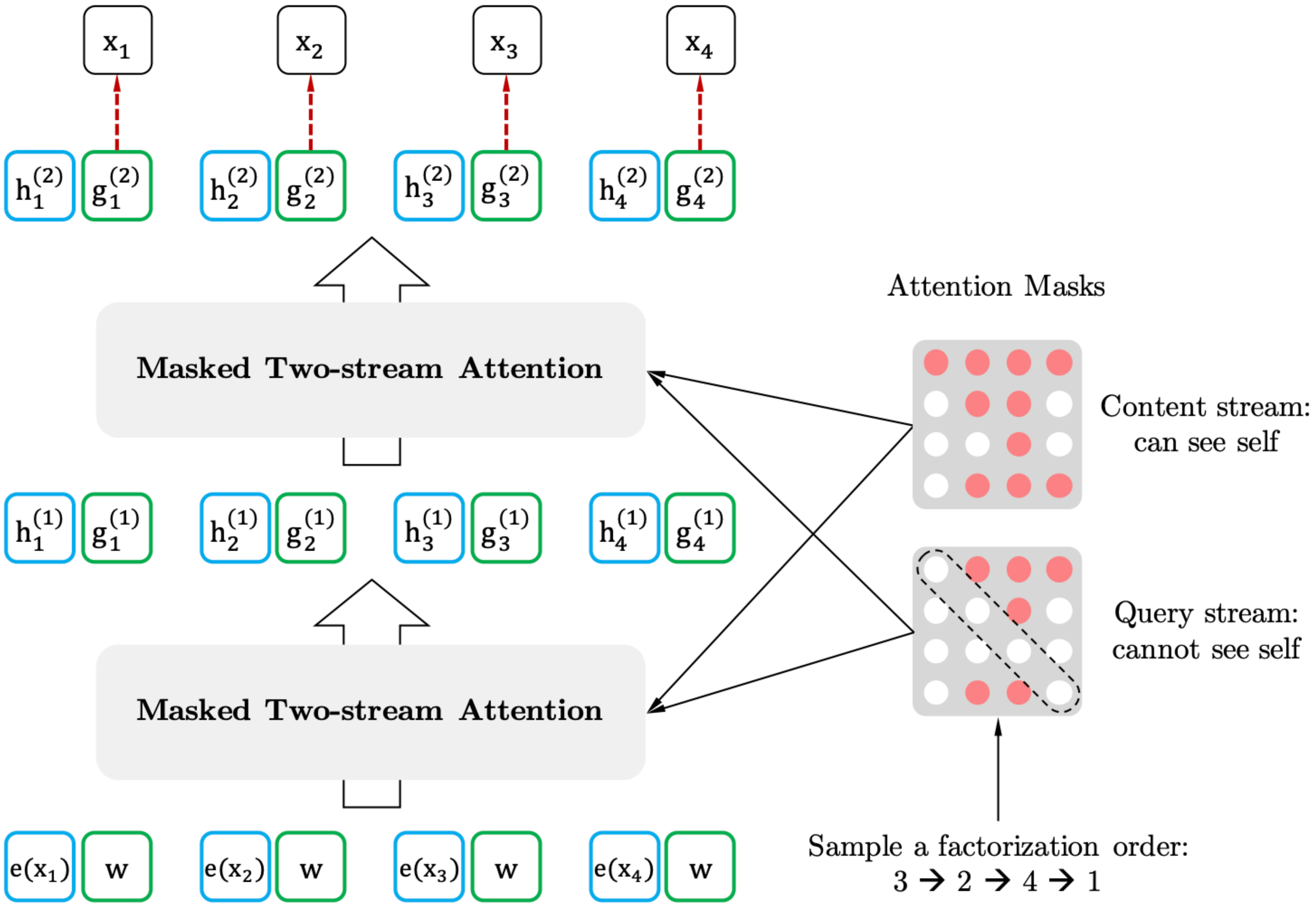
XLNet

https://arxiv.org/abs/1906.08237

Yang+ 2019

- Relative position embeddings (using auto-regressive TransformerXL)
  - Absolute attention: position 4 → 5; position 128 → 129
  - Relative attention: position $t \rightarrow (t-1)$
- Mask prediction over all token positions using permutation on factorization order (sample a factorization order: 3 → 2 → 1 → 4)
  - Two stream self-attention: standard and query on [MASK] token
  - Permute only factorization order, not sequence order

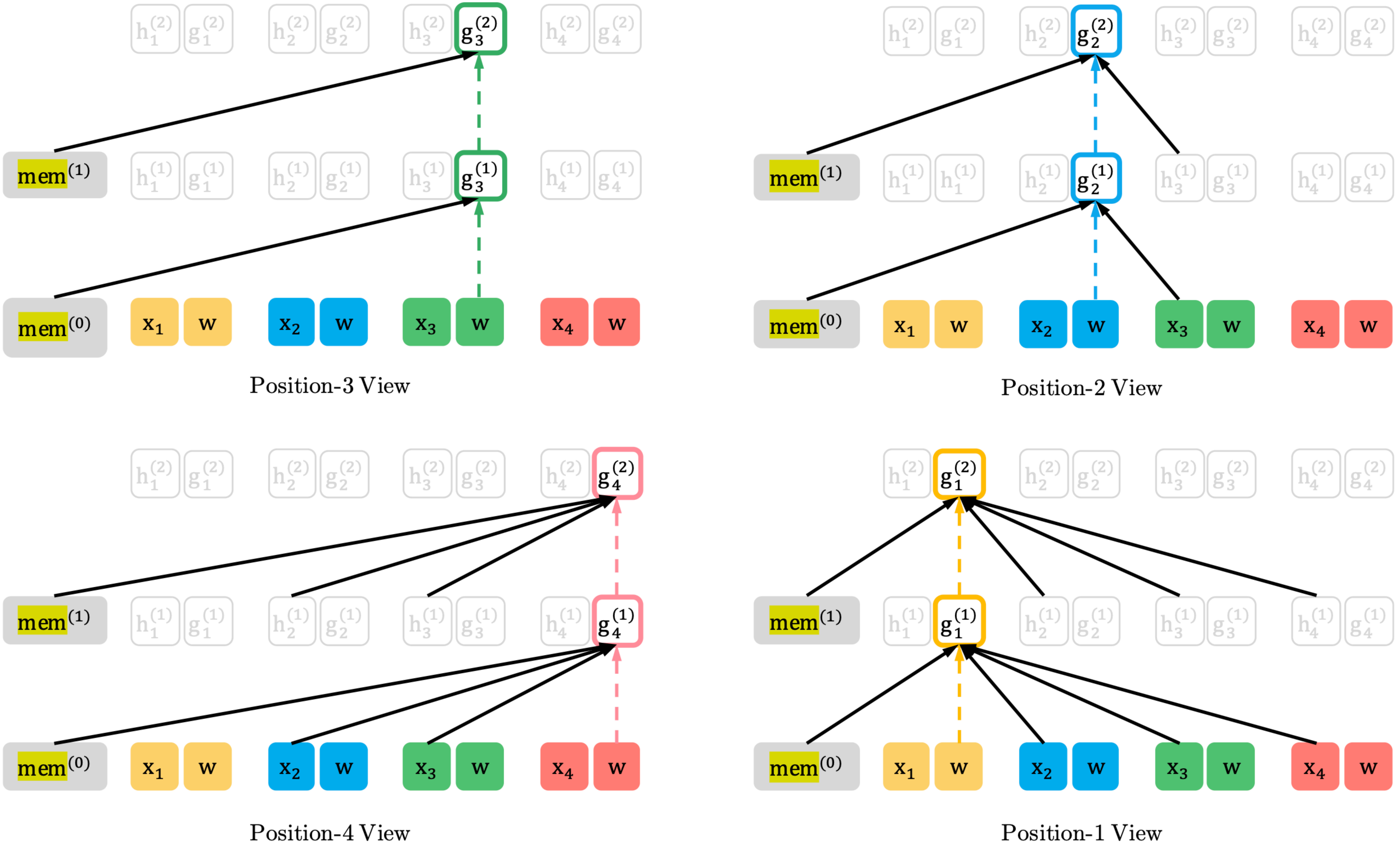
XLNet



x_1
x_2
x_3
x_4
$h_1^{(2)}$ $g_1^{(2)}$ $h_2^{(2)}$ $g_2^{(2)}$ $h_3^{(2)}$ $g_3^{(2)}$ $h_4^{(2)}$ $g_4^{(2)}$
Masked Two-stream Attention
$h_1^{(1)}$ $g_1^{(1)}$ $h_2^{(1)}$ $g_2^{(1)}$ $h_3^{(1)}$ $g_3^{(1)}$ $h_4^{(1)}$ $g_4^{(1)}$
Masked Two-stream Attention
$e(x_1)$ w $e(x_2)$ w $e(x_3)$ w $e(x_4)$ w
Attention Masks
Content stream:
can see self
Query stream:
cannot see self
Sample a factorization order:
3 → 2 → 4 → 1

XLNet

Split View of the Query Stream
(Factorization order: 3 → 2 → 4 → 1)



XLNet

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
Single-task single models on dev								
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**

Transformers for pretraining

- Self-supervised Transformer based models shattered language understanding benchmarks in NLP in 2018.
- Trained on large text corpus with self-supervised objectives and then transferred.

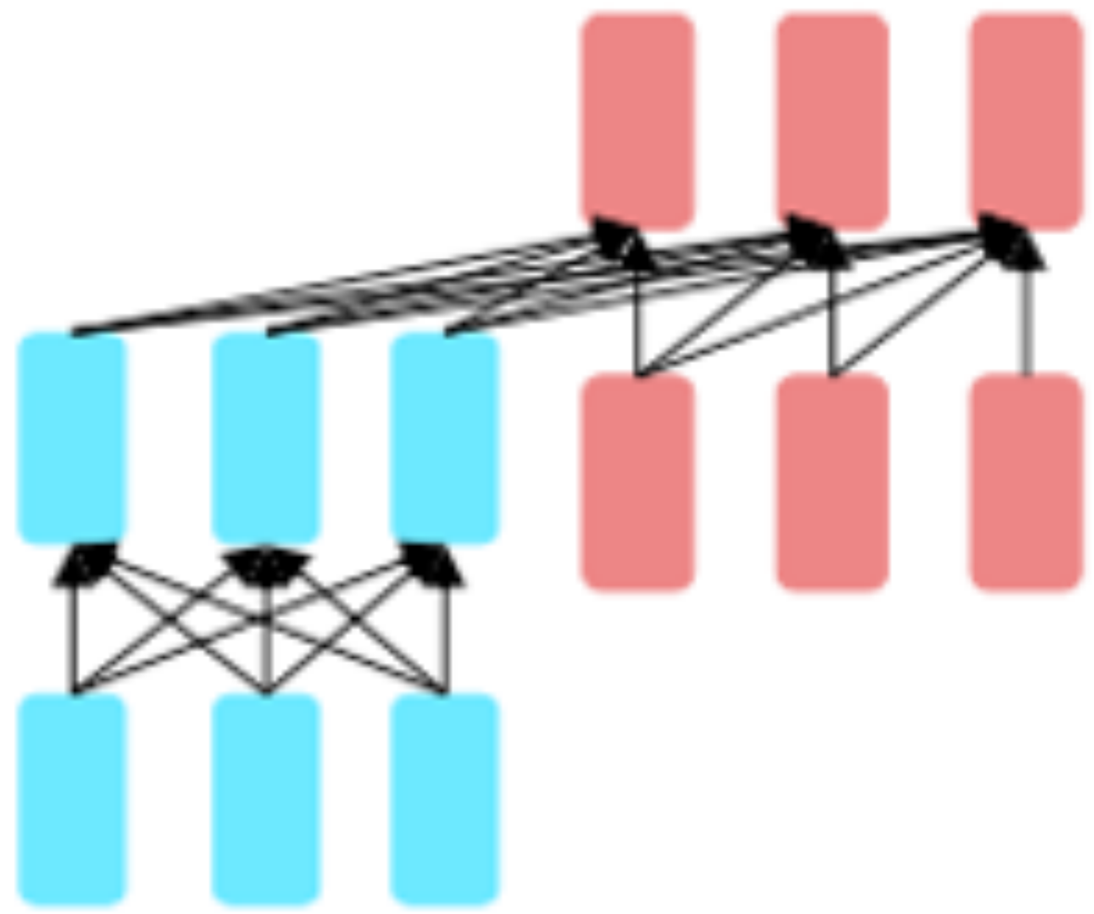
Encoder only

- Masked language models
- Bidirectional context
- BERT + variants (e.g. RoBERTa)

Decoder only

- Language models
- Can't condition on future words, good for generation
- GPT, LLaMa, PaLM

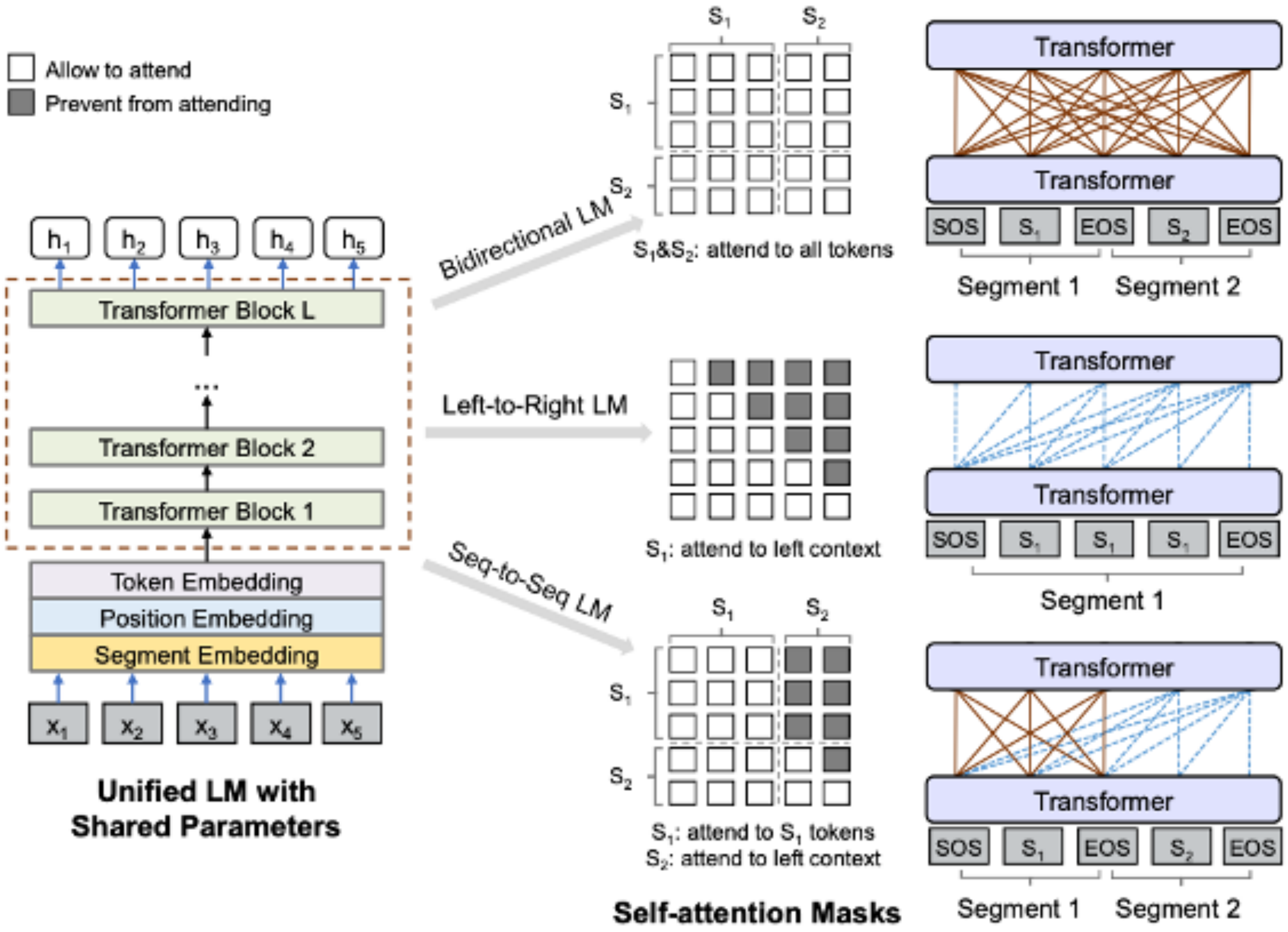
Encoder-Decoder



- Combine benefits of both
- Original Transformer, UniLM, BART, T5

Encoder-Decoder pretraining

- Combine advantages of both encoder and decoder
- Seq2Seq LM with masking
- Next sentence prediction



Unified Language Model Pre-training for Natural Language Understanding and Generation [Dong et al, NeurIPS 2019]

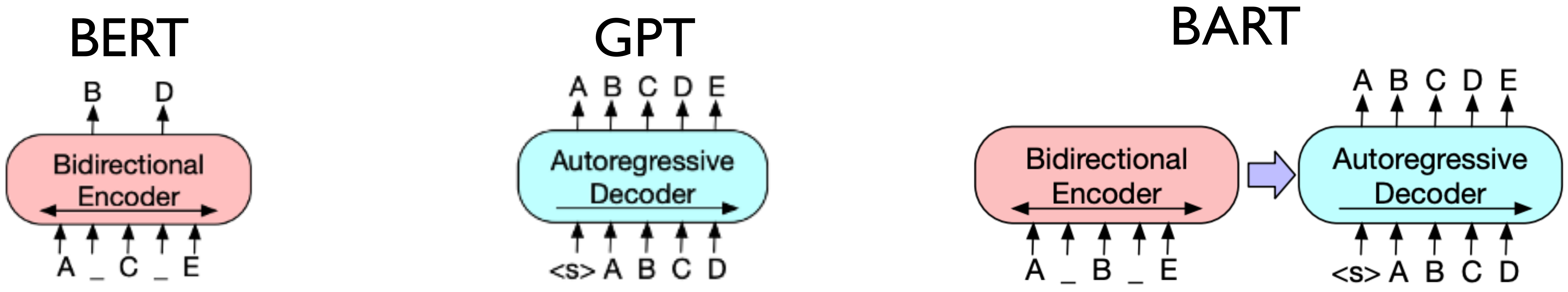
UniLM v1

- Combine benefits of BERT (encoder) and GPT (decoder)

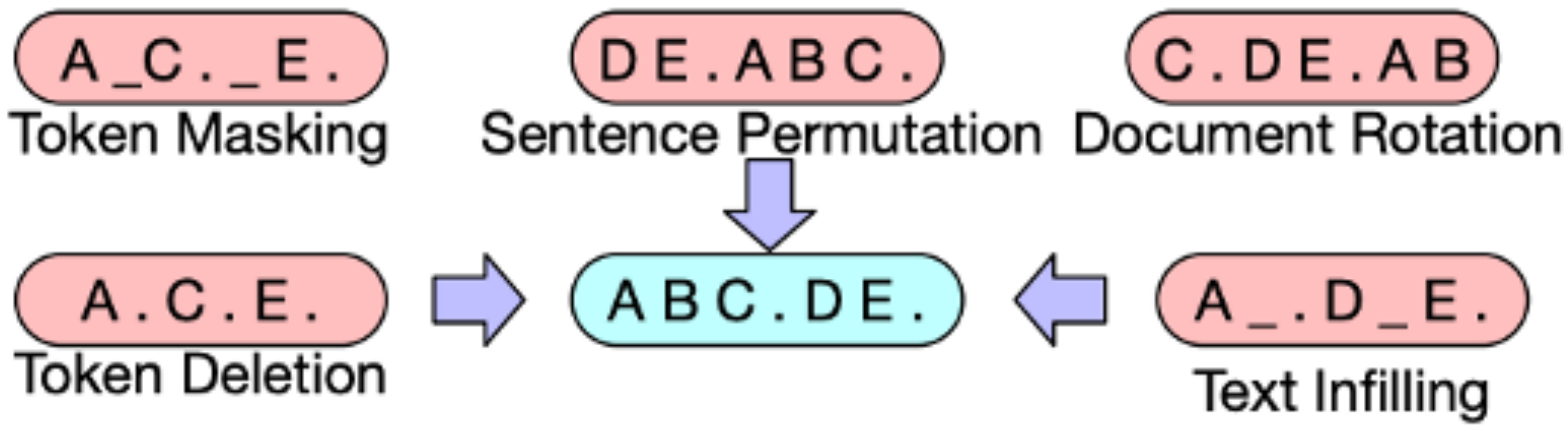
Model	CoLA MCC	SST-2 Acc	MRPC F1	STS-B S Corr	QQP F1	MNLI-m/mm Acc	QNLI Acc	RTE Acc	WNLI Acc	AX Acc	**Score**
GPT	45.4	91.3	82.3	80.0	70.3	82.1/81.4	87.4	56.0	53.4	29.8	72.8
$\text{BERT}_{\text{LARGE}}$	60.5	**94.9**	89.3	86.5	**72.1**	86.7/**85.9**	**92.7**	70.1	65.1	39.6	80.5
UNILM	**61.1**	94.5	**90.0**	**87.7**	71.7	**87.0/85.9**	**92.7**	**70.9**	65.1	38.4	**80.8**

Unified Language Model Pre-training for Natural Language Understanding and Generation
Dong et al, Microsoft, NeurIPS 2019

BART: Denoising seq2seq training



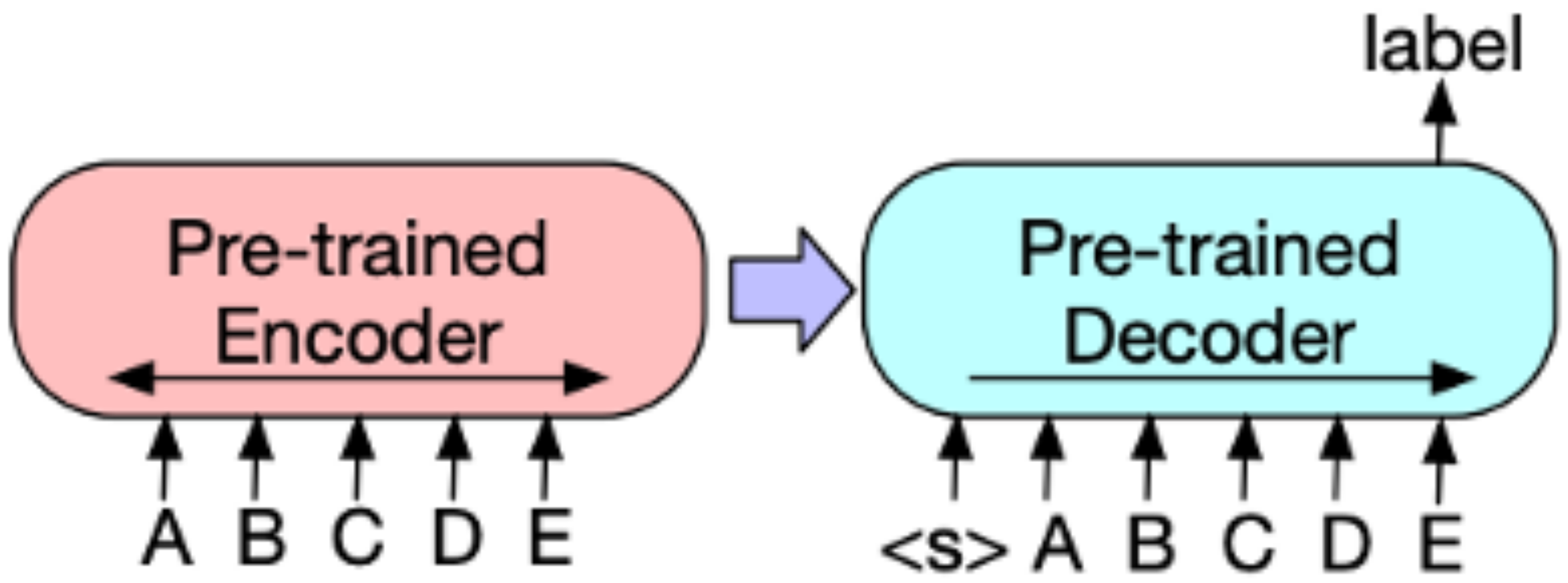
- Combine benefits of BERT (encoder) and GPT (decoder)
- More flexibility in noise generation



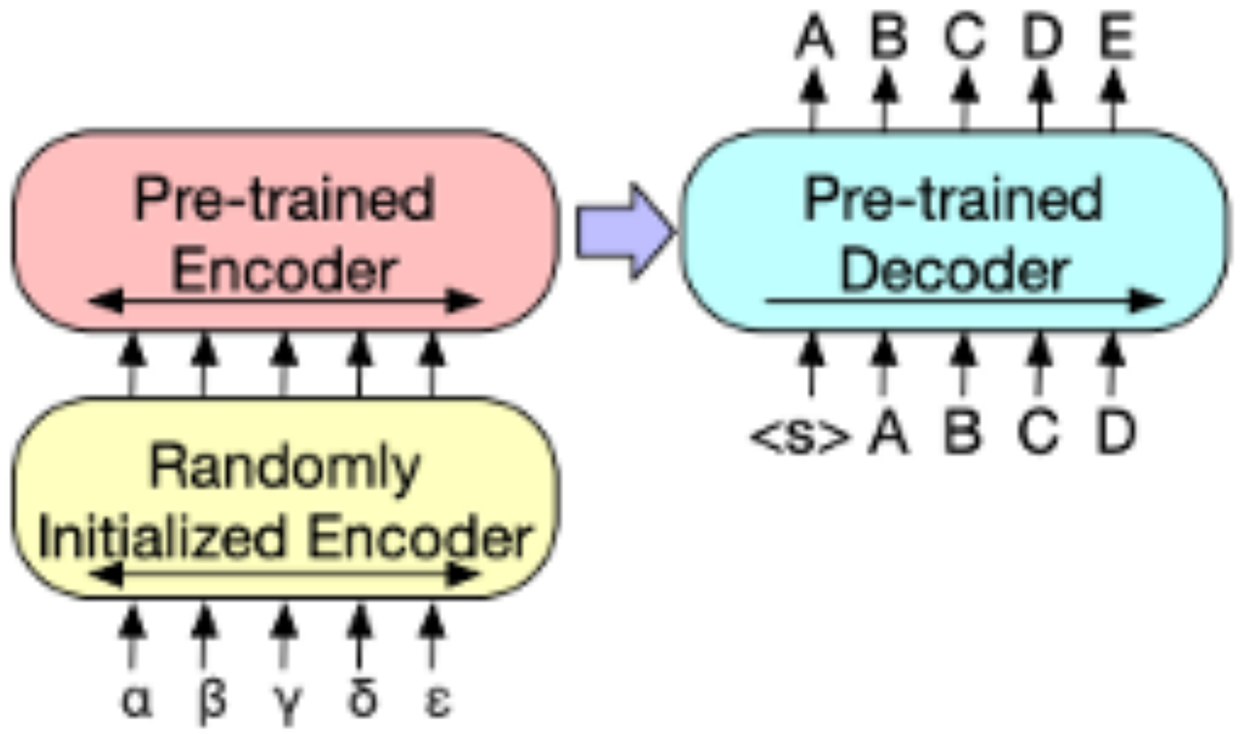
BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
Lewis et al, Facebook AI, ACL 2020

BART: Denoising seq2seq training

Classification



Machine Translation



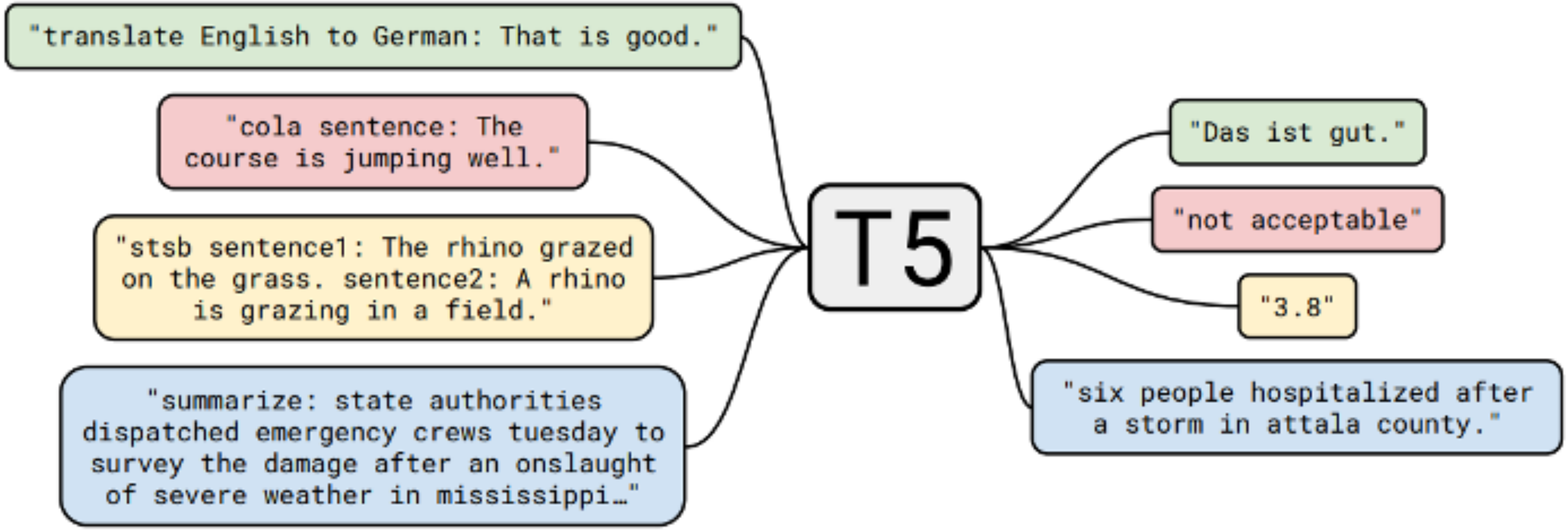
	SQuAD 1.1 EM/F1	SQuAD 2.0 EM/F1	MNLI m/mm	SST Acc	QQP Acc	QNLI Acc	STS-B Acc	RTE Acc	MRPC Acc	CoLA Mcc
BERT	84.1/90.9	79.0/81.8	86.6/-	93.2	91.3	92.3	90.0	70.4	88.0	60.6
UniLM	-/-	80.5/83.4	87.0/85.9	94.5	-	92.7	-	70.9	-	61.1
XLNet	**89.0**/94.5	86.1/88.8	89.8/-	95.6	91.8	93.9	91.8	83.8	89.2	63.6
RoBERTa	88.9/**94.6**	**86.5/89.4**	**90.2/90.2**	96.4	92.2	94.7	**92.4**	86.6	**90.9**	**68.0**
BART	88.8/**94.6**	86.1/89.2	89.9/90.1	**96.6**	**92.5**	**94.9**	91.2	**87.0**	90.4	62.8

BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
Lewis et al, Facebook AI, ACL 2020

T5: Text to Text Transfer Transformer

https://arxiv.org/abs/1910.10683

- Treat all NLP problems as encoding text and generating text
- Trained on cleaned up version of Common Crawl



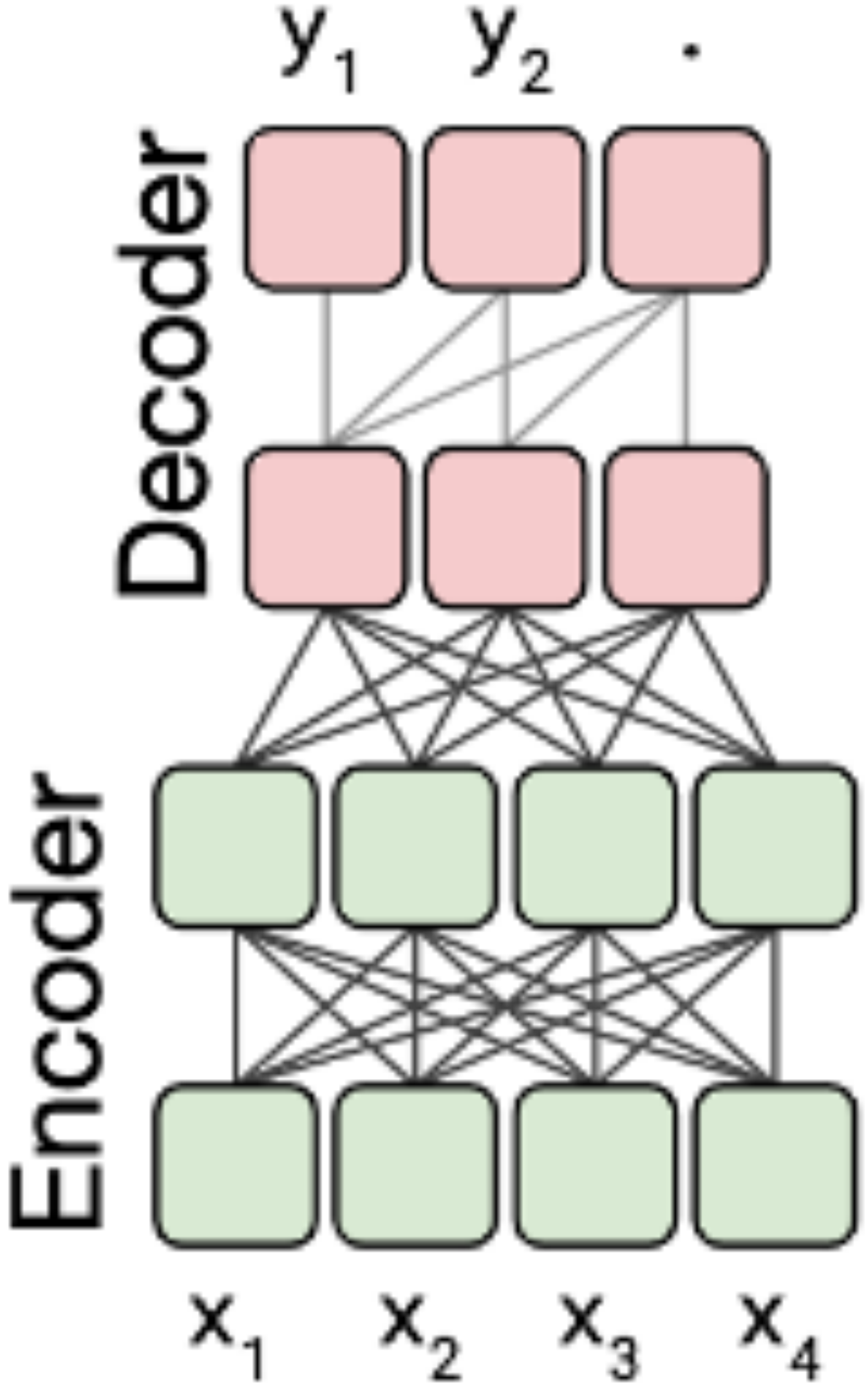
Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [Raffel et al, Google, JMLR 2020]

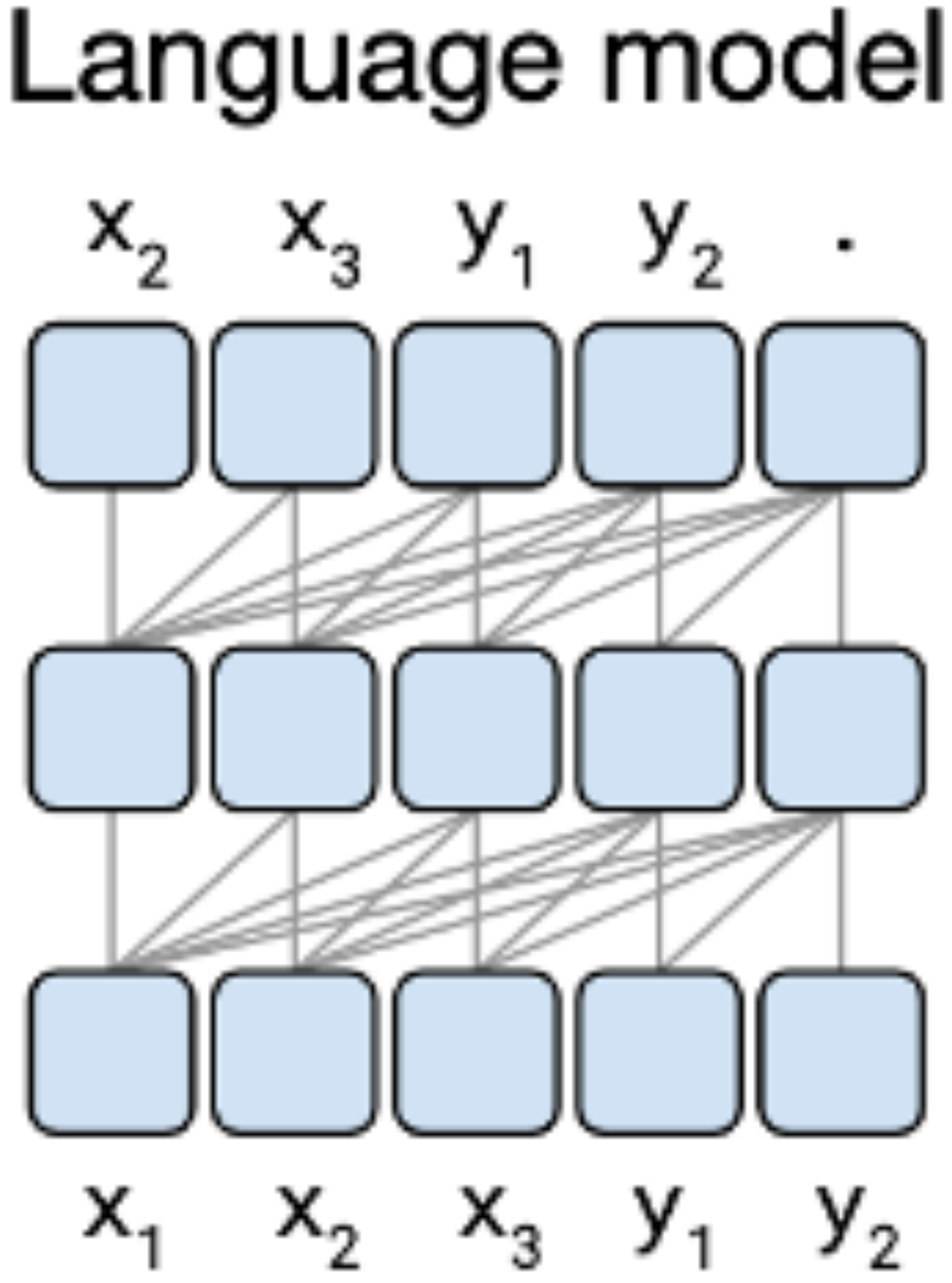
T5: Text to Text Transfer Transformer

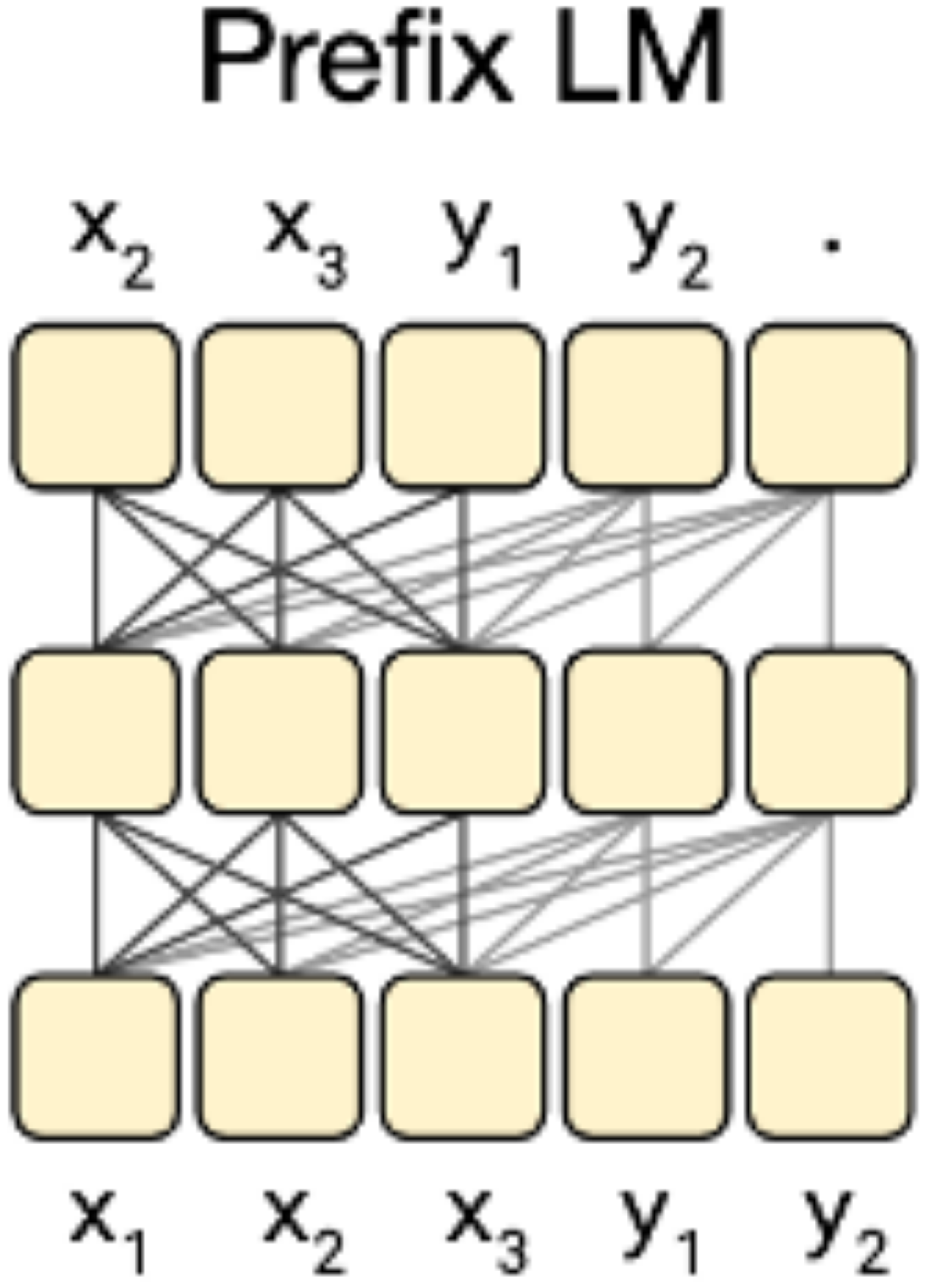
Normally: Separate parameters for encoder/decoder

Causal masking only

Masking similar to encoder/decoder





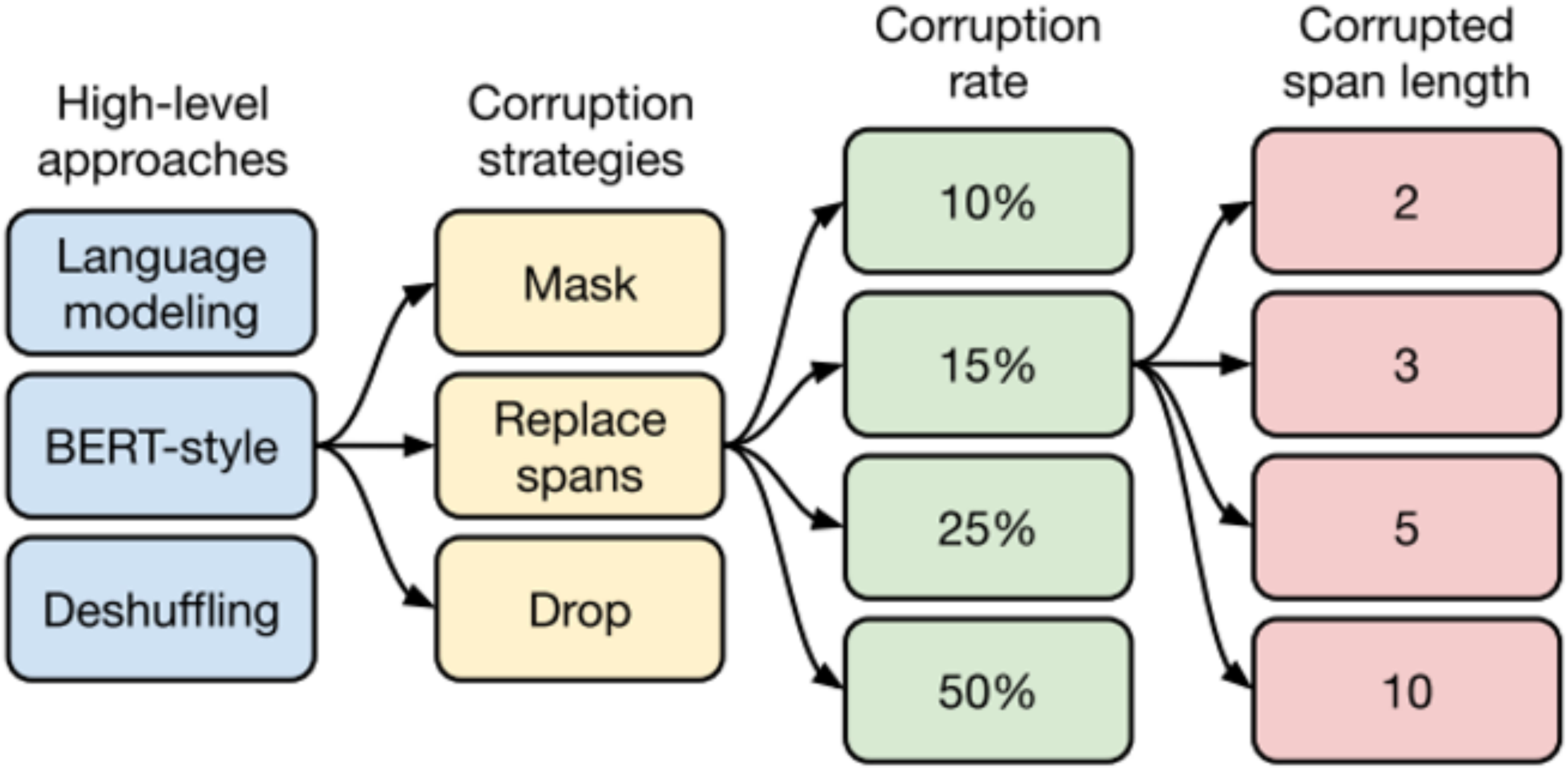


Can force sharing of parameters for encoder/decoder

Similar performance, less parameters

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [Raffel et al, Google, JMLR 2020]

T5: Text to Text Transfer Transformer



Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [Raffel et al, Google, JMLR 2020]

T5 (use both encoder and decoder)

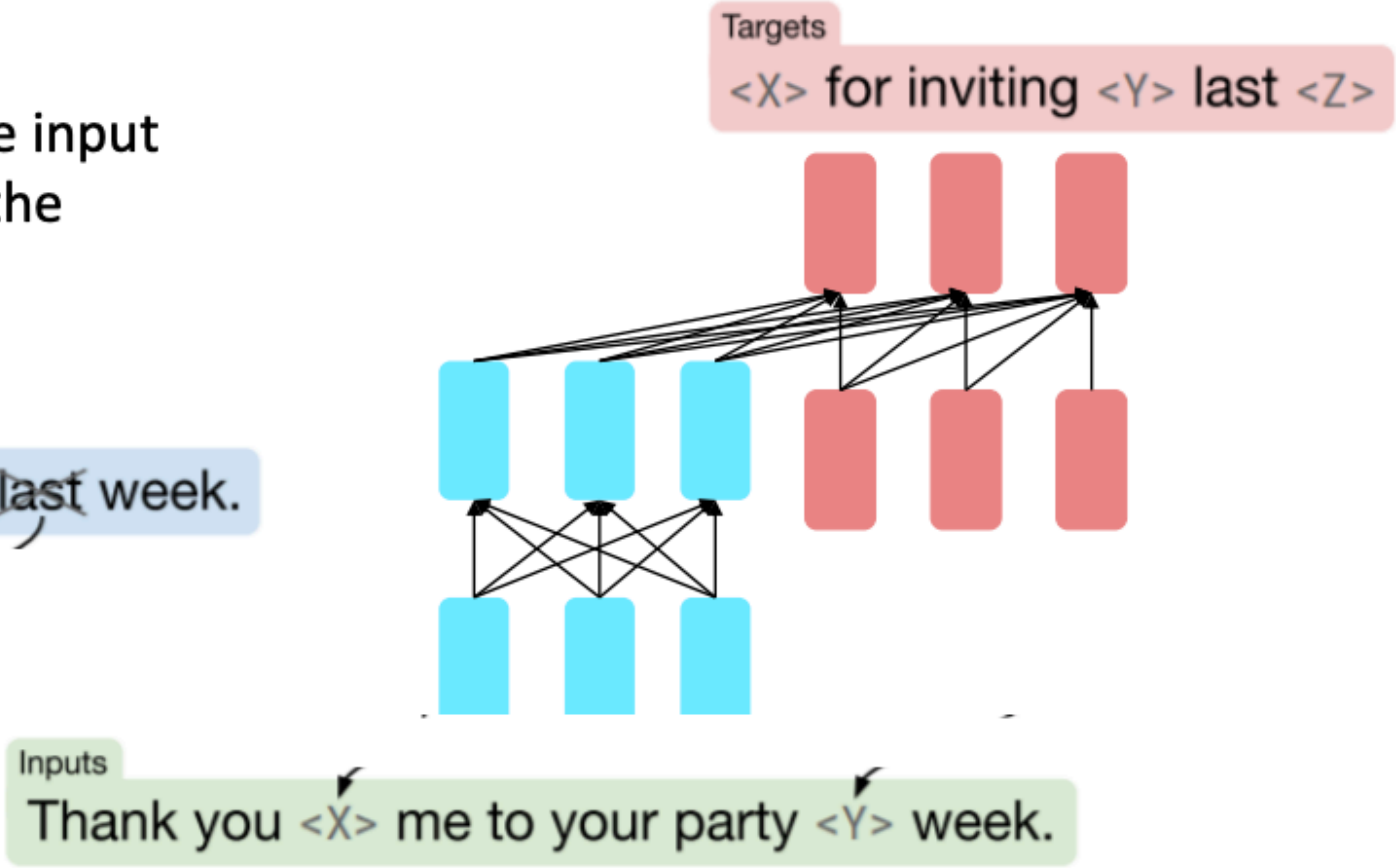
Span corruption works best

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.



Slide Credit: Stanford CS224n, John Hewitt

T5: Text to Text Transfer Transformer

Different corruption type

	Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Predict all	BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
	MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
Predict corrupted	★ 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**

Different corruption rate

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [Raffel et al, Google, JMLR 2020]

T5 (use both encoder and decoder)

Raffel et al., 2018 found encoder-decoders to work better than decoders for their tasks, and span corruption (denoising) to work better than language modeling.

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 summary

https://arxiv.org/abs/1910.10683

Raffel+ 2019

- Ablation study on many aspects of pre-training and fine-tuning
    - Model size (bigger is better; 11B parameters)
    - Amount of training data (more is better)
    - Domain / cleanliness of training data [-ve]
    - Pre-training objective (e.g. span length of masked text) [-ve]
    - Ensemble models [-ve]
    - Fine-tuning recipe (e.g. only allow top k layers to fine-tune) [-ve]
    - Multi-task training [-ve]

Using pre-trained LLMs

Using LLMs for tasks

- So your language model can complete a sentence, but you may want to do different things
  - Classify whether a email is SPAM or NOT SPAM
  - Answer a question: when was Albert Einstein born?
  - Extract information from text
- If I give it a piece of text, how do I tell it whether I want to translate it French, summarize it, or make it into a poem?

Using LLMs for tasks

Develop specialized model for your task (with LM as part)

- Hookup appropriate inputs/outputs
- Fine-tuning parameters (include some LM parameters) for task

Try to use the LM network as it is (no extra network training)

- Zero-shot / few-shot prompting (in-context learning)

Try to have smaller LM to allow running on various devices

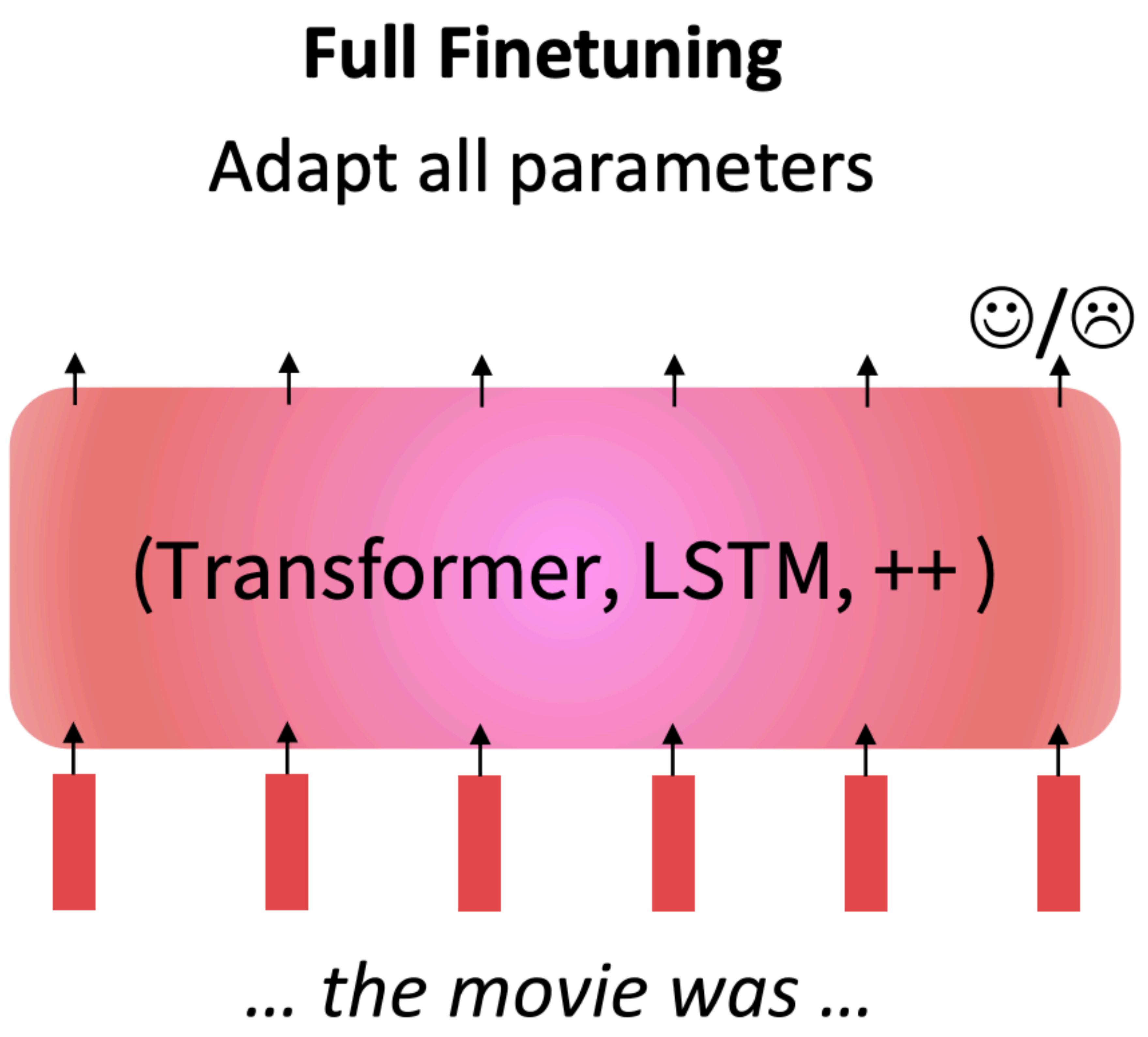
- Model distillation and pruning

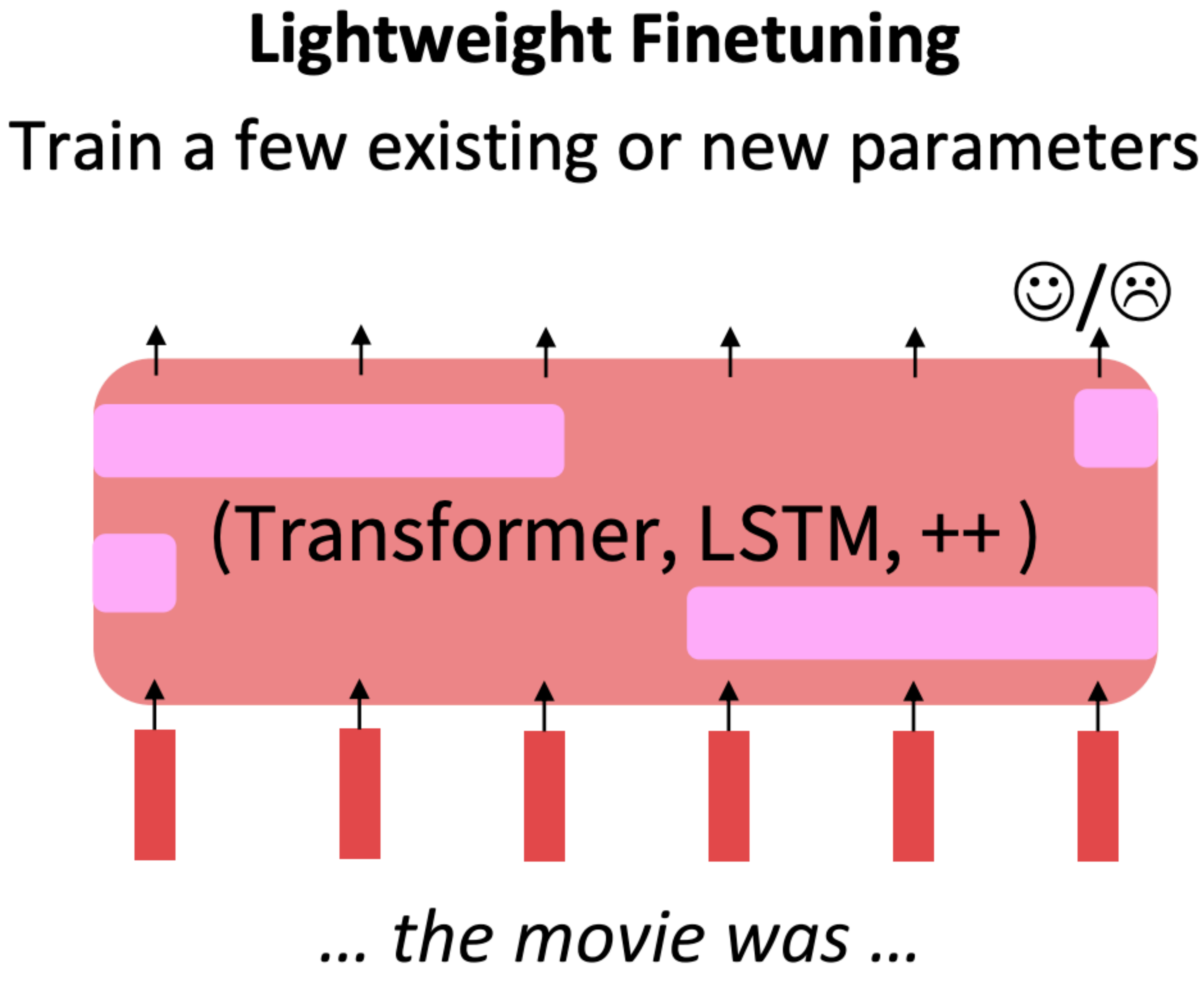
Different ways to fine-tune

- Parameter efficient fine-tuning (PEFT)
- Instruction tuning (fine-tune with instructions)
- Reinforcement learning with human feedback (train with modified objective that incorporates human preferences)

Full finetuning vs parameter efficient fine-tuning

- Finetuning every parameter in a pretrained model works well, but is memory-intensive.
- **Lightweight** finetuning methods adapt pretrained models in a constrained way.
- Leads to **less overfitting** and/or **more efficient finetuning and inference**.



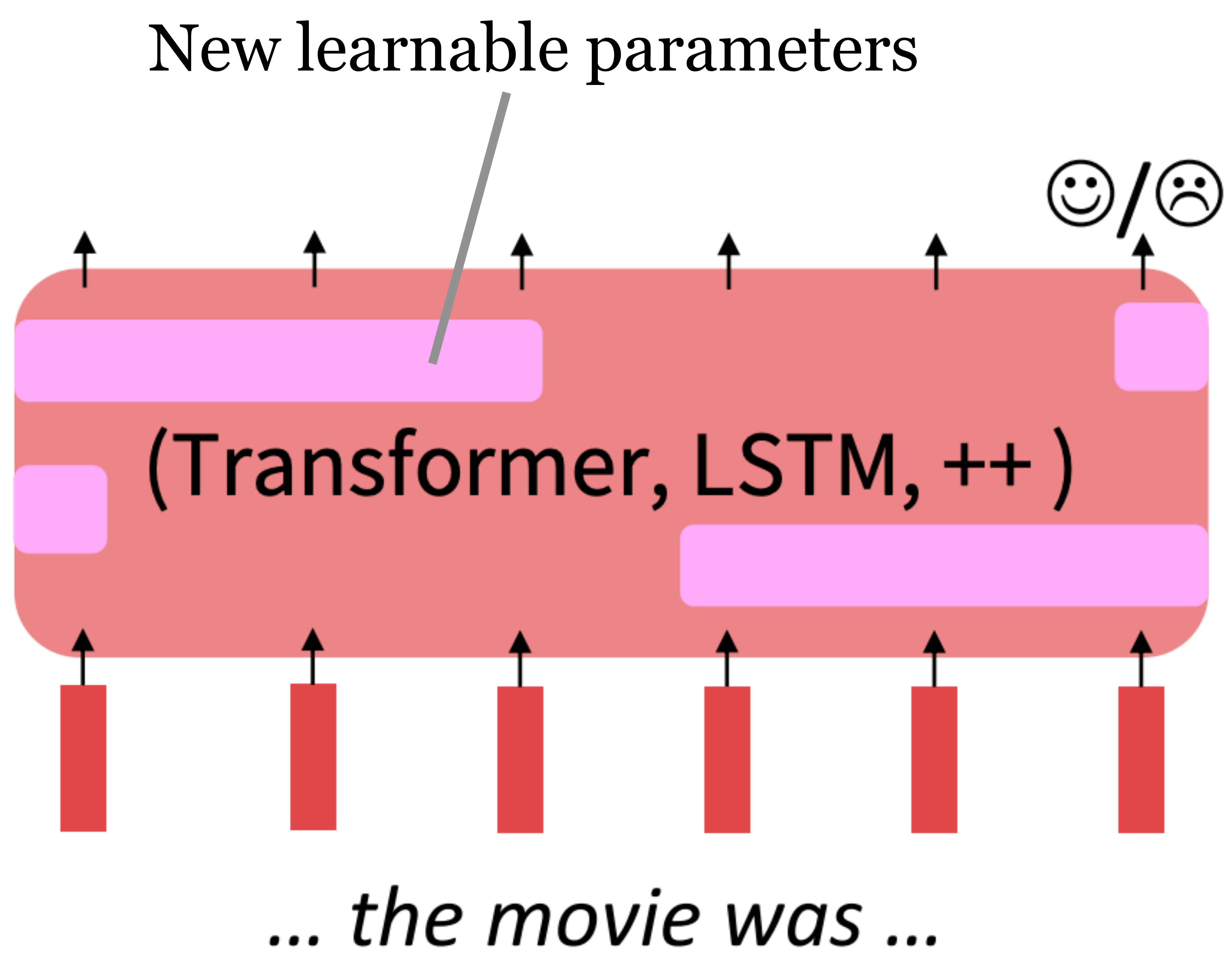


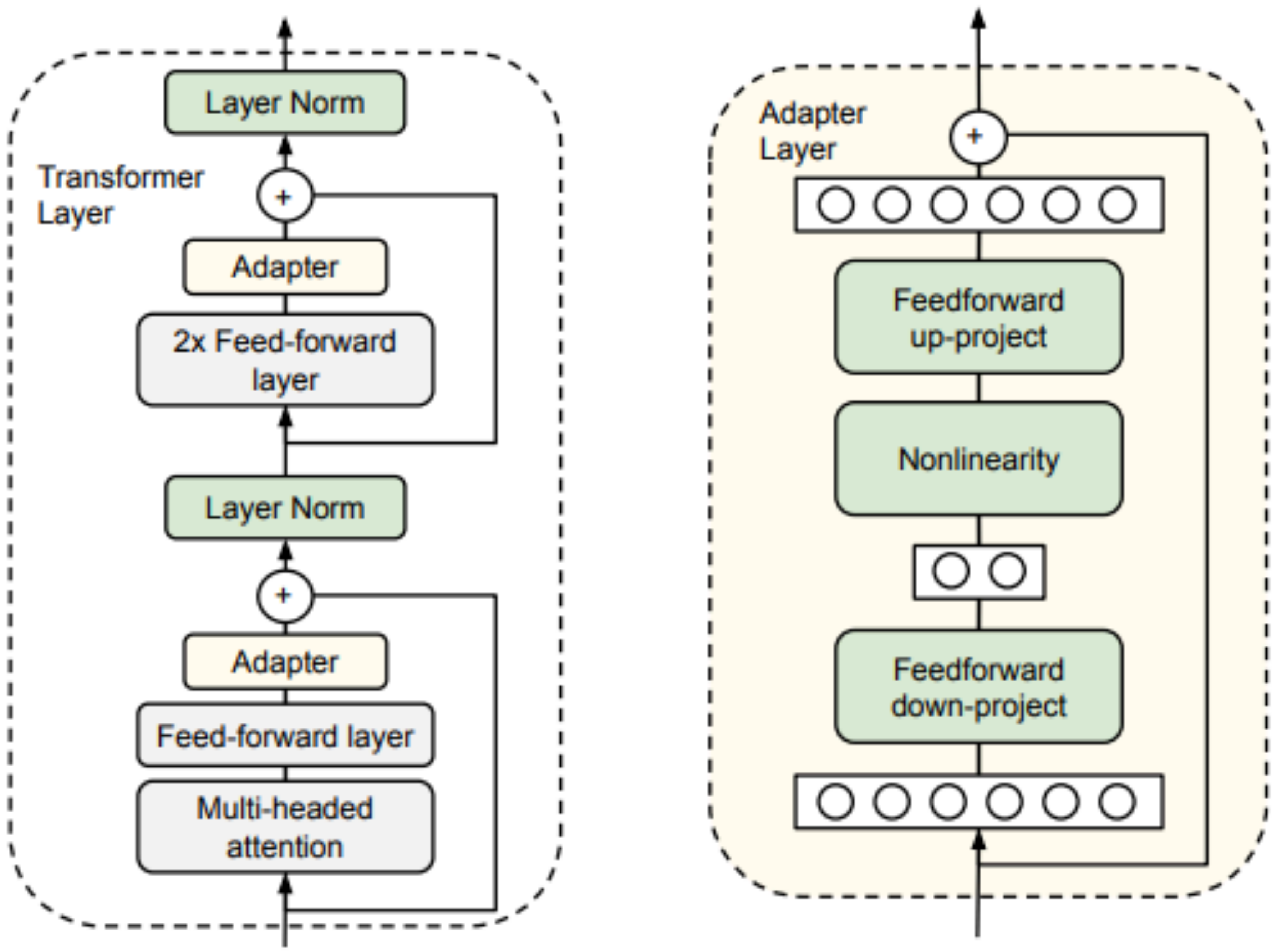
Slide Credit: Stanford CS224n, John Hewitt

[Liu et al., 2019; Joshi et al., 2020]

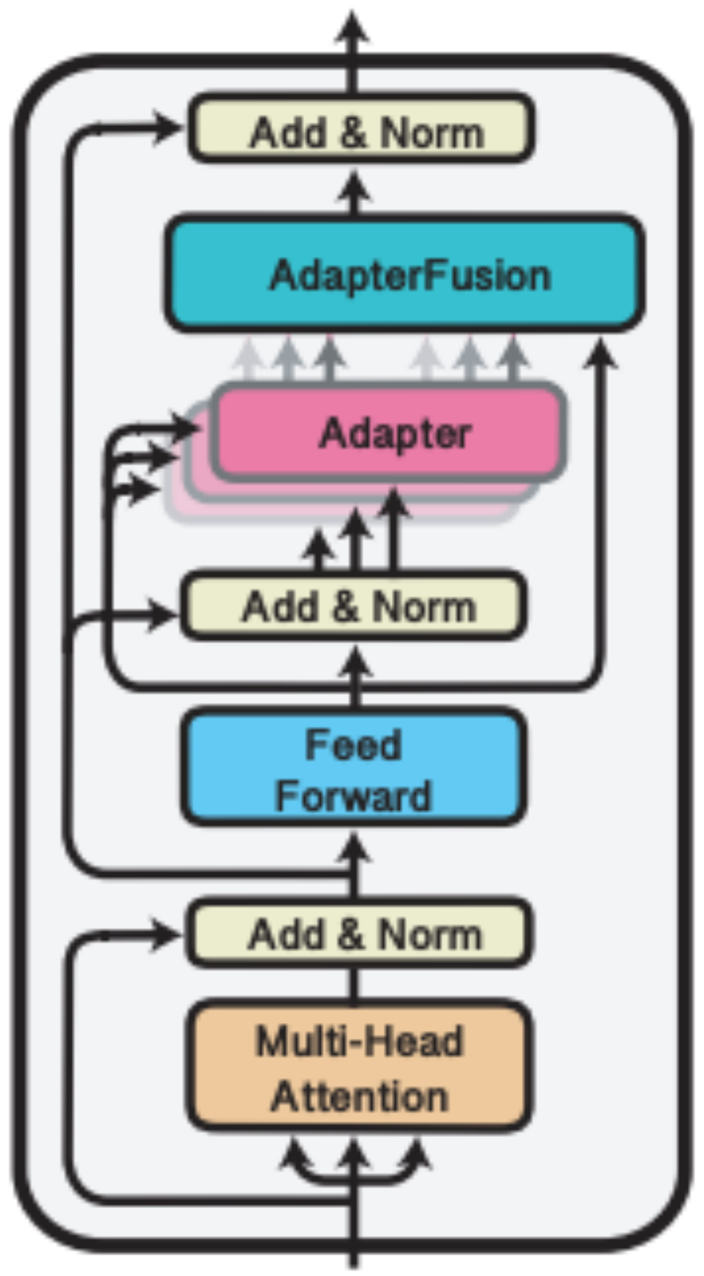
Parameter-Efficient Finetuning: Adapters

- Add lightweight network with new learnable parameters
- Only these parameters are fine-tuned, rest are frozen





[Houlsby et al., 2019]

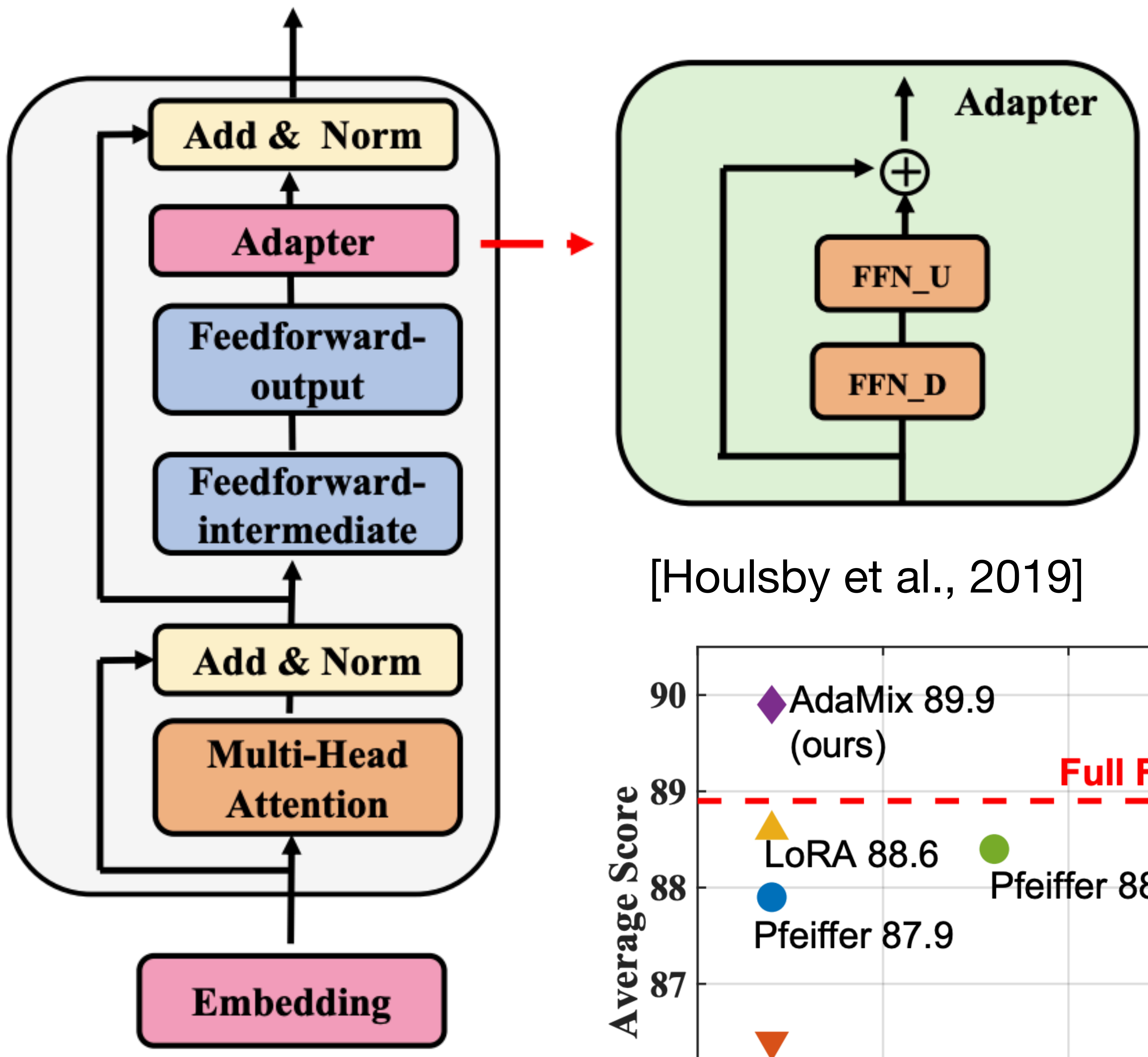


[Pfeiffer et al., 2021]

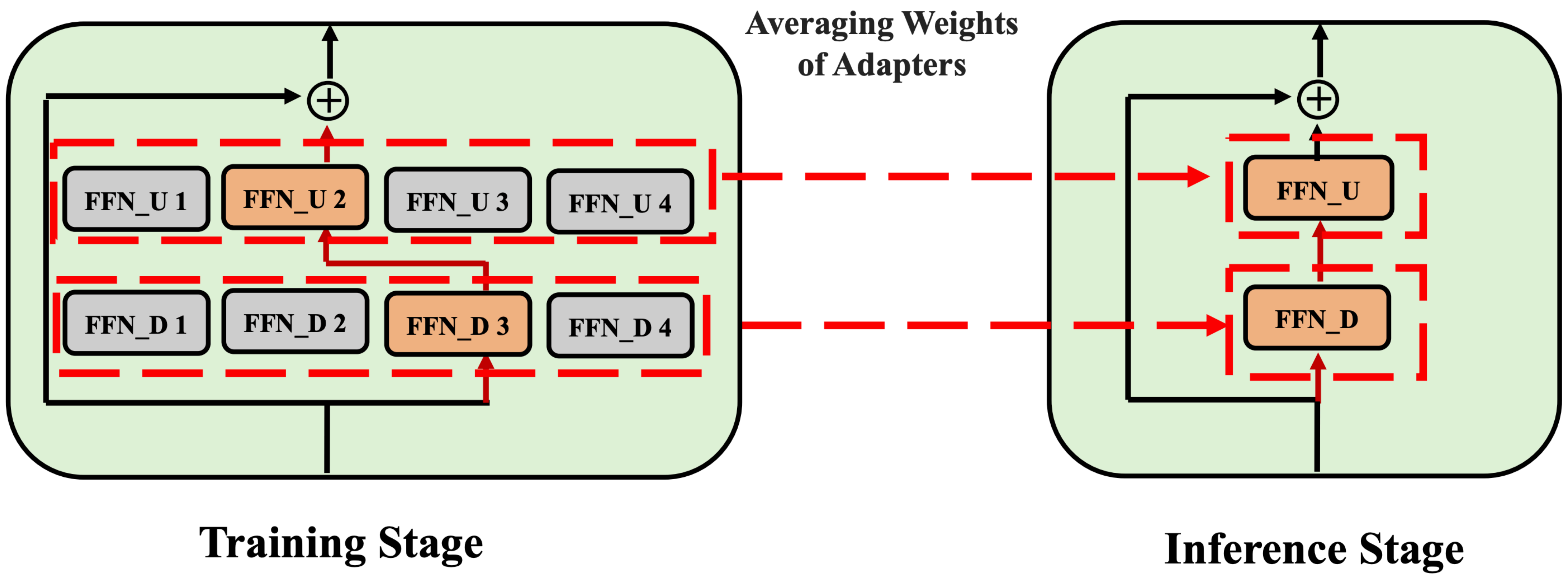
https://github.com/adapter-hub/adapter-transformers

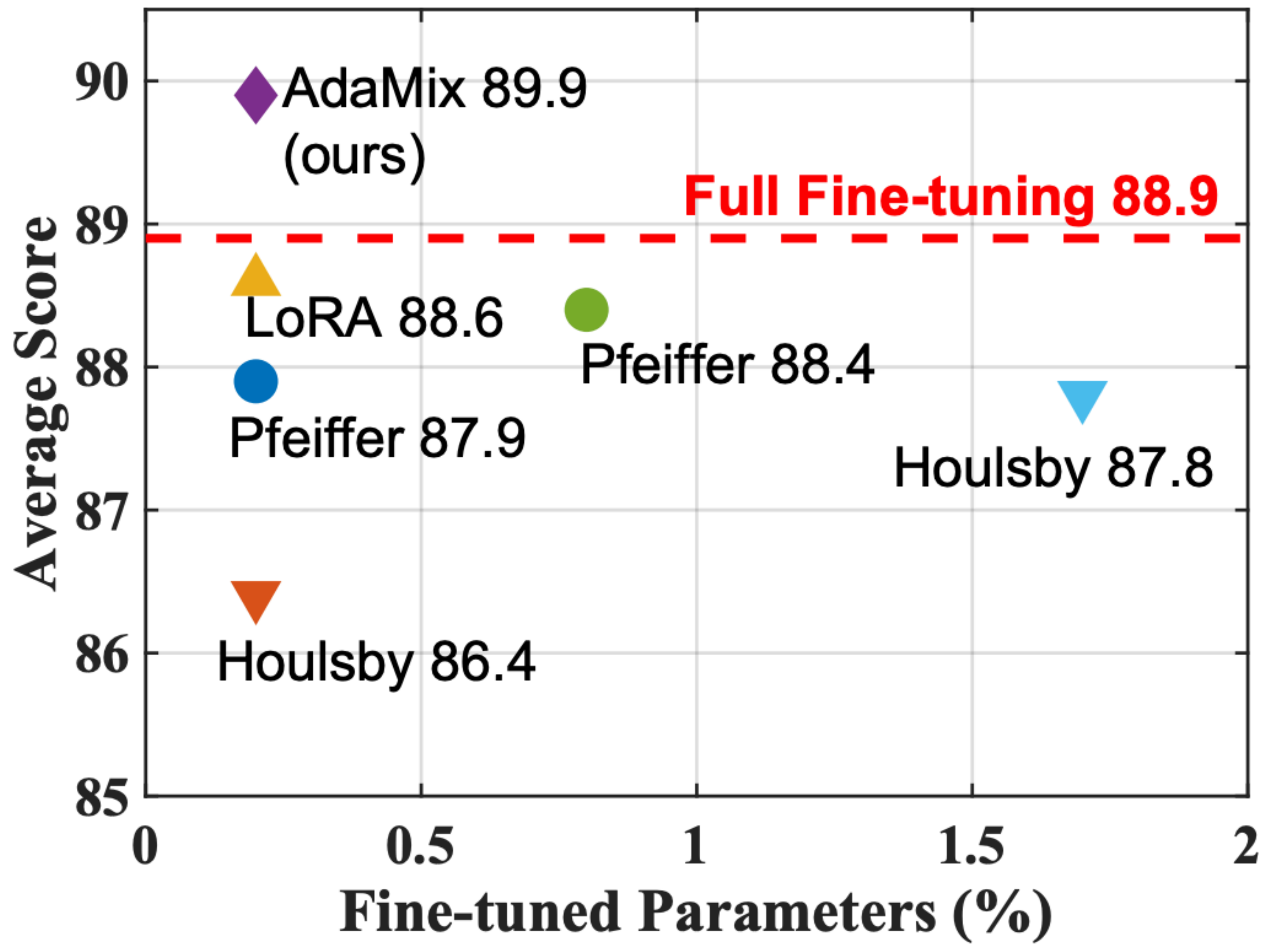
Parameter-Efficient Finetuning: Adapters

- **Mixture of adapters** - stochastically selected during training
- Average weights of adapters during inference



[Houlsby et al., 2019]



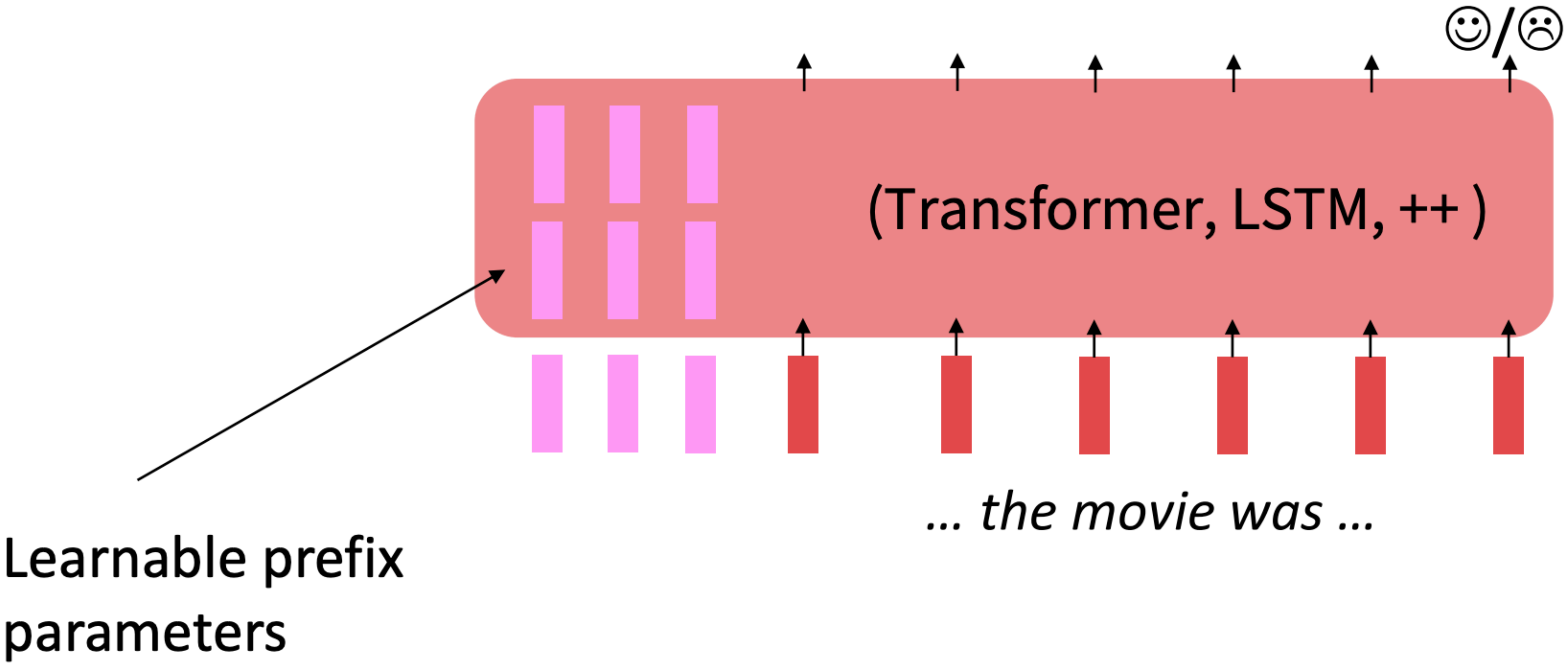


Performance on GLUE, fine-tuning of RoBERTa-large

[AdaMix, Wang et al., EMNLP 2022]

Parameter-Efficient Finetuning: Prefix-Tuning, Prompt tuning

- Prefix-Tuning adds a prefix of parameters, and freezes all pretrained parameters.
- The prefix is processed by the model just like real words would be.
- Advantage: each element of a batch at inference could run a different tuned model.

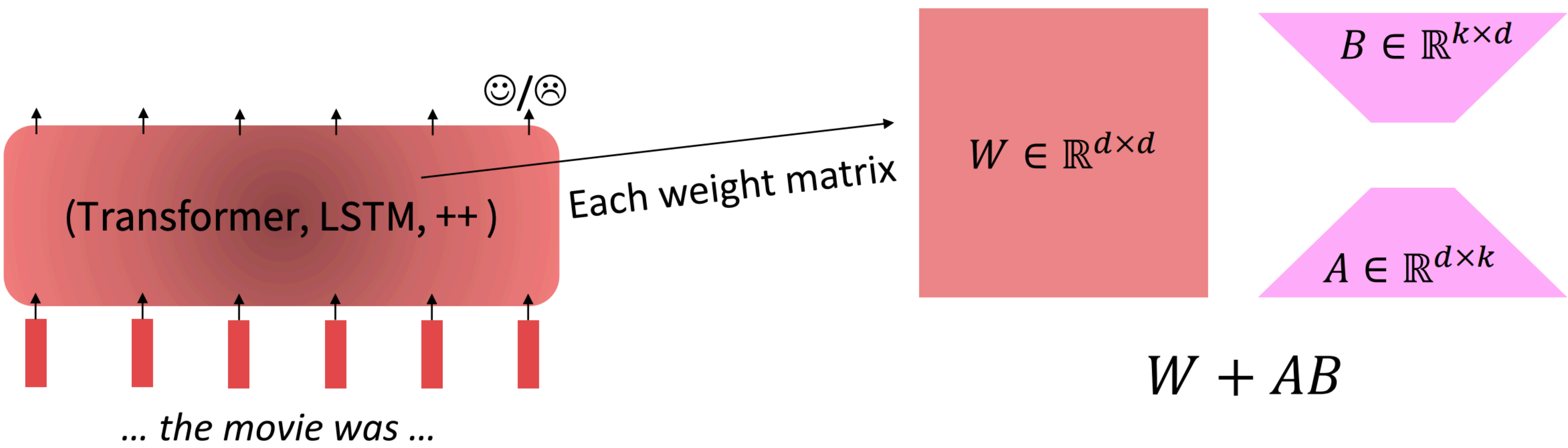


Slide Credit: Stanford CS224n, John Hewitt

[Li and Liang, 2021; Lester et al., 2021]

Parameter-Efficient Finetuning: Low-Rank Adaptation

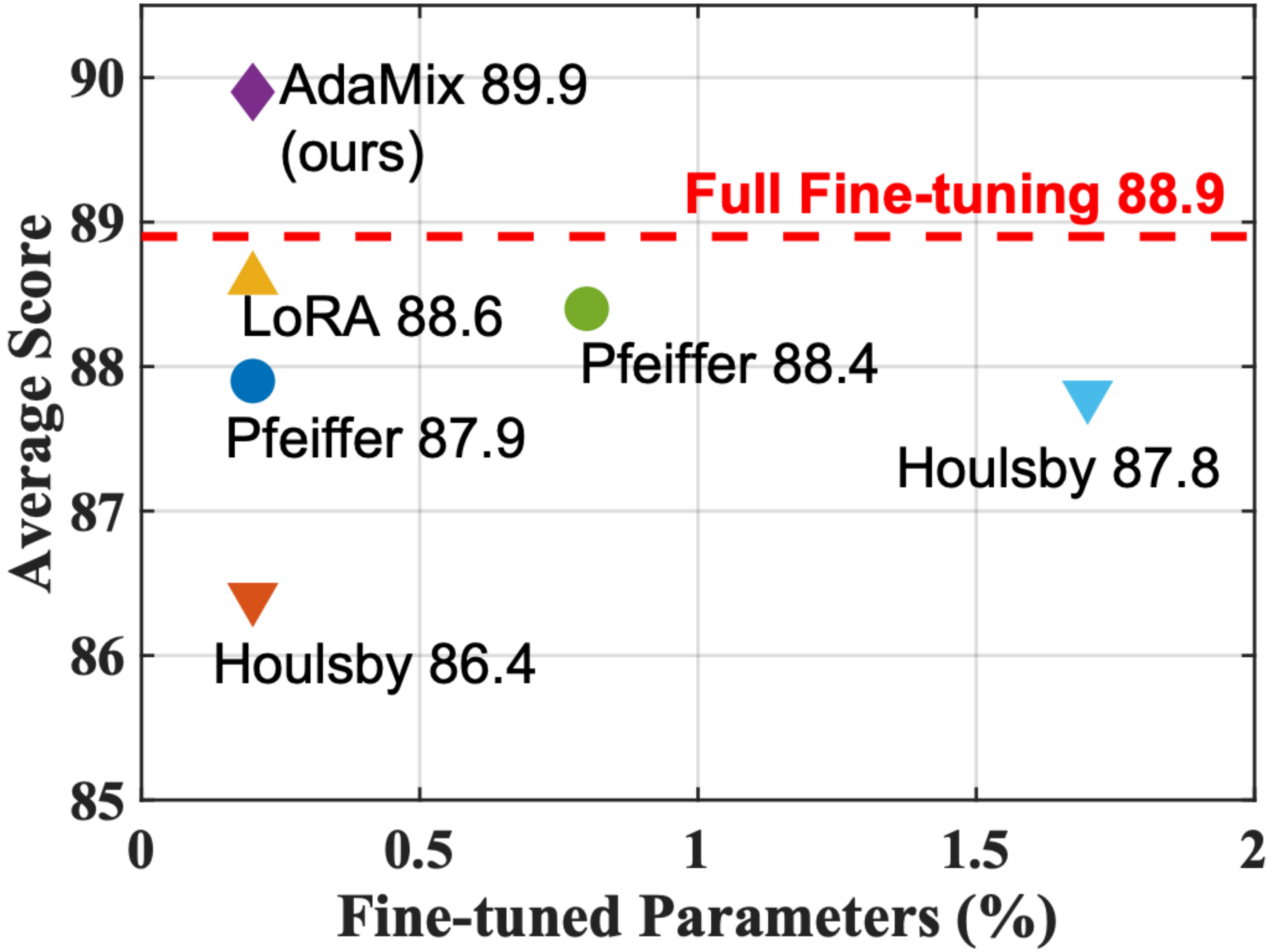
- Low-Rank Adaptation learns a low-rank "diff" between the pretrained and finetuned weight matrices.
- Easier to learn than prefix-tuning



$$W + AB$$

Parameter-Efficient Finetuning: Low-Rank Adaptation

Model	**#Param.**	**M A**	**'S-B ırson**	**Avg.**
Full Fine-tuning†	355.0M	9	**2.4**	88.9
Pfeiffer Adapter†	3.0M	9	2.1	88.4
Pfeiffer Adapter†	0.8M	9	1.9	87.9
Houlsby Adapter†	6.0M	8	1.0	87.8
Houlsby Adapter†	0.8M	9	1.5	86.4
LoRA†	0.8M	9	2.3	88.6
AdaMix Adapter	0.8M	**9**	**2.4**	**89.9**



Good performance by tuning just a fraction of the weights

[AdaMix, Wang et al., EMNLP 2022]

Going toward smaller powerful LMs

- Knowledge Distillation
  - DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. Sanh et al. NeurIPS Workshop 2019
  - TinyBERT: Distilling BERT for Natural Language Understanding. Jiao et al. Findings of ACL 2020
- Quantization
  - Q8BERT: Quantized 8bit BERT, Zafrir et al, NeurIPS Workshop 2019
- Model Pruning
  - Compressing BERT: Studying the effects of weight pruning on transfer learning. Gordon et al. Workshop of ACL 2020.