Advanced Natural Language Processing

Lecture 8: Pretrained Language Model

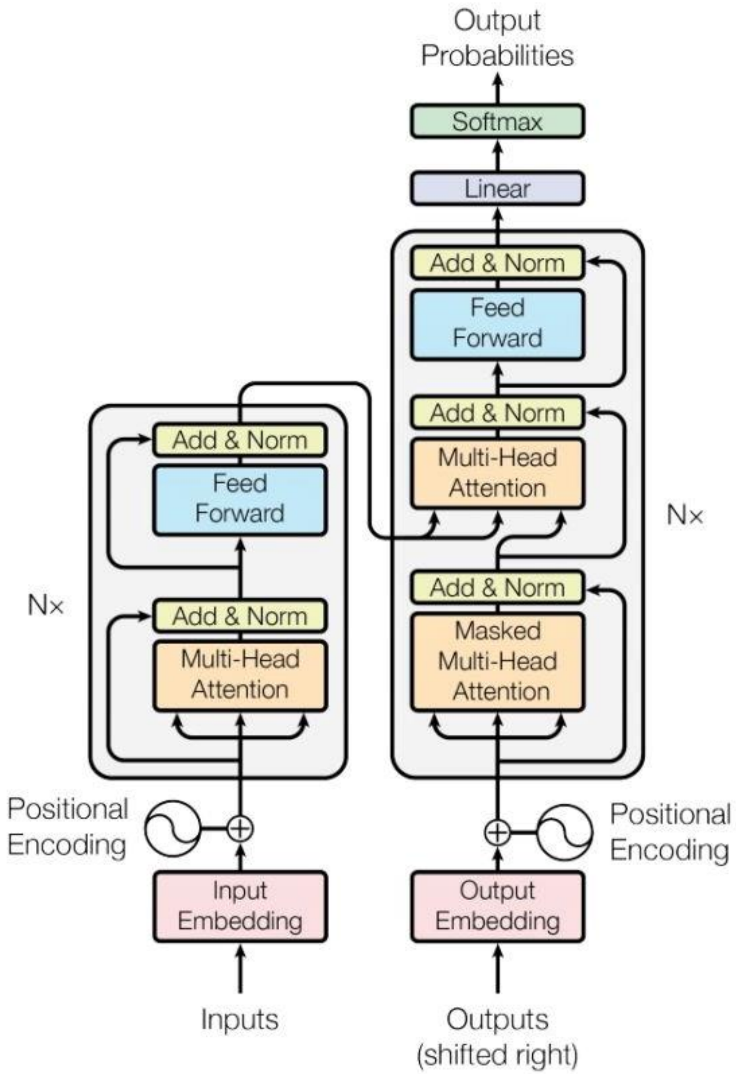
陈冠华 CHEN Guanhua

Department of Statistics and Data Science

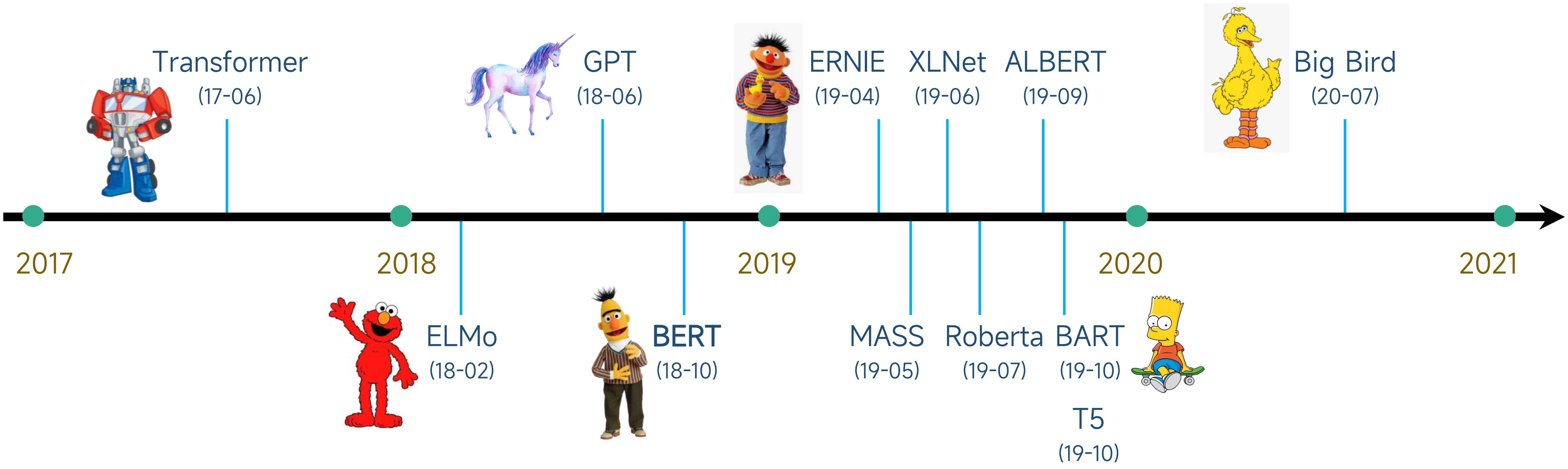
南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Pretrained Language Model

- Pretraining
  - The model is trained with unlabeled data in a self-supervised manner
  - E.g., masked language modeling, next token prediction
- Finetuning
  - Initialized with the pretrained model and trained with the supervised dataset of the target/downstream task



Timeline of Pretrained Language Models



Different Types of Pretrained Models

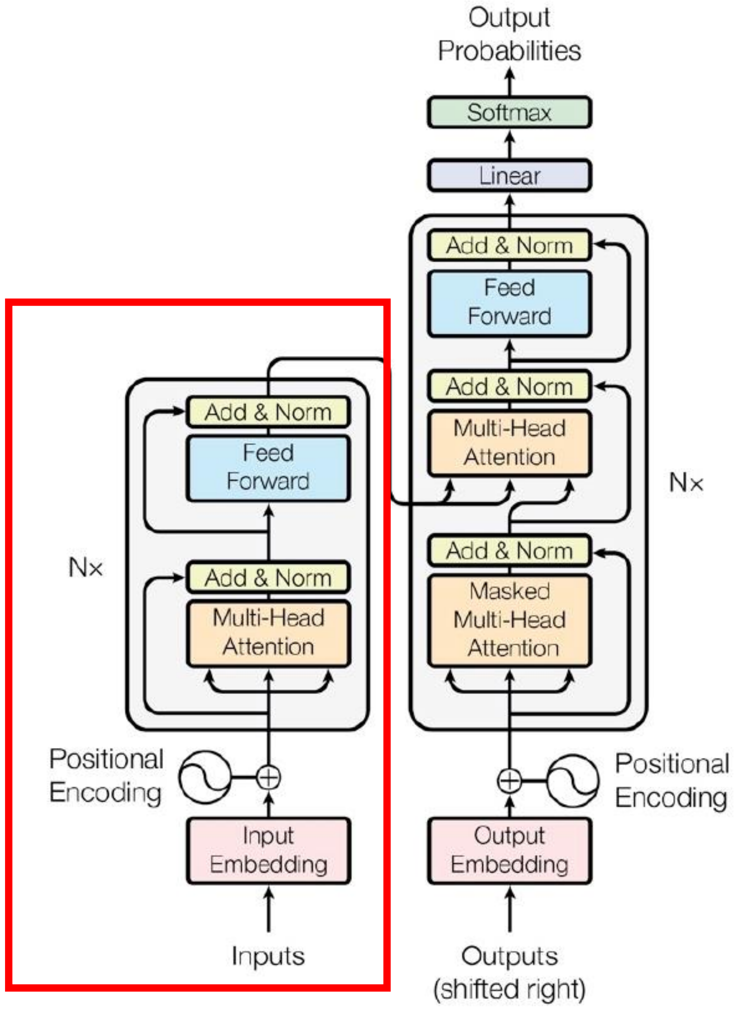
Encoder-only

Decoder-only

Encoder-Decoder

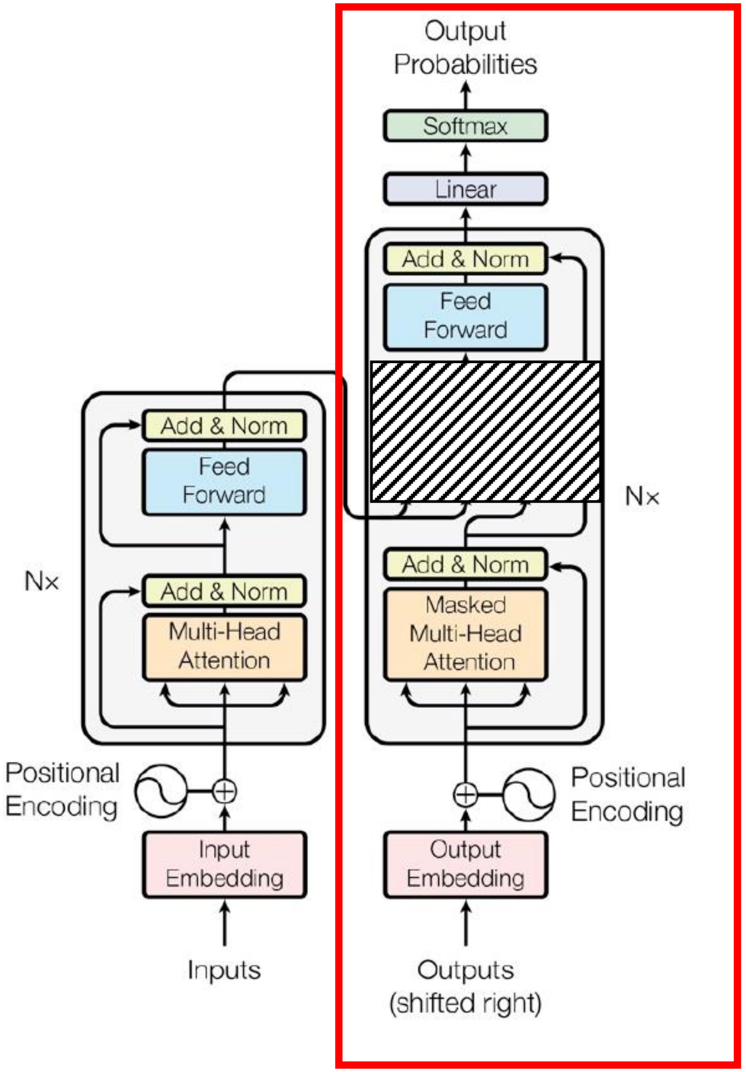
Pretrained Encoder

Encoder-only



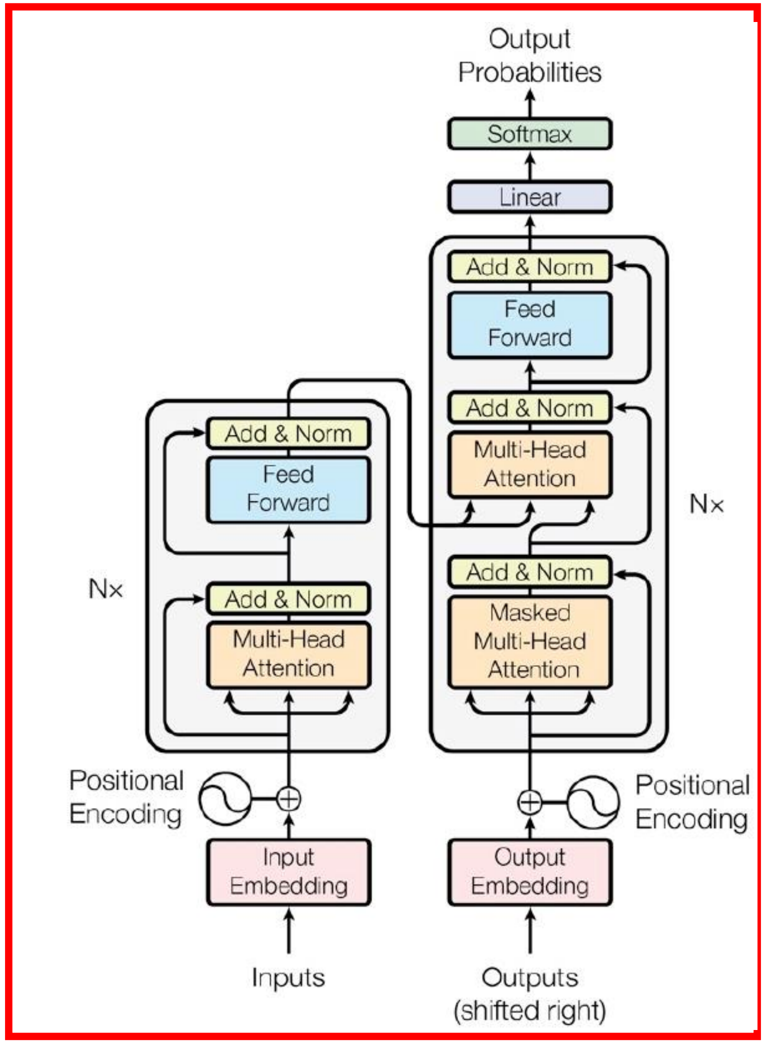
Pretrained Decoder

Decoder-only



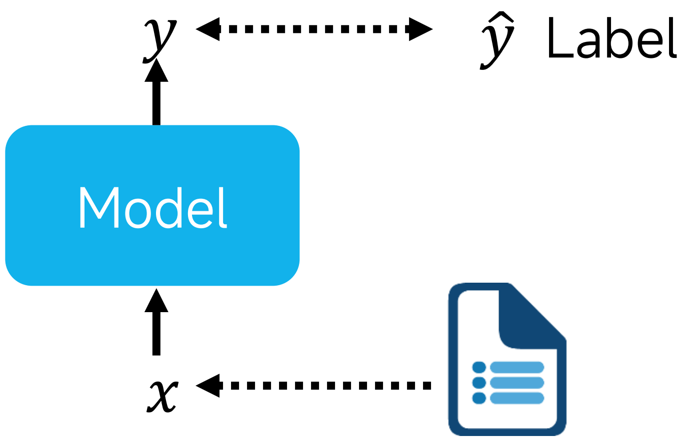
Pretrained Encoder-Decoder

Encoder-decoder

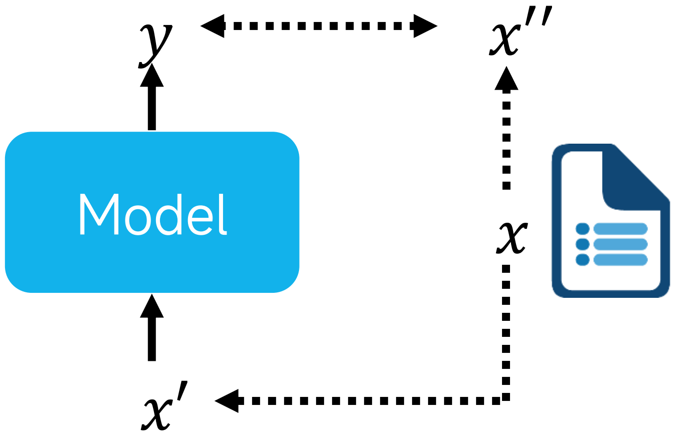


Self-Supervised Learning

Supervised learning



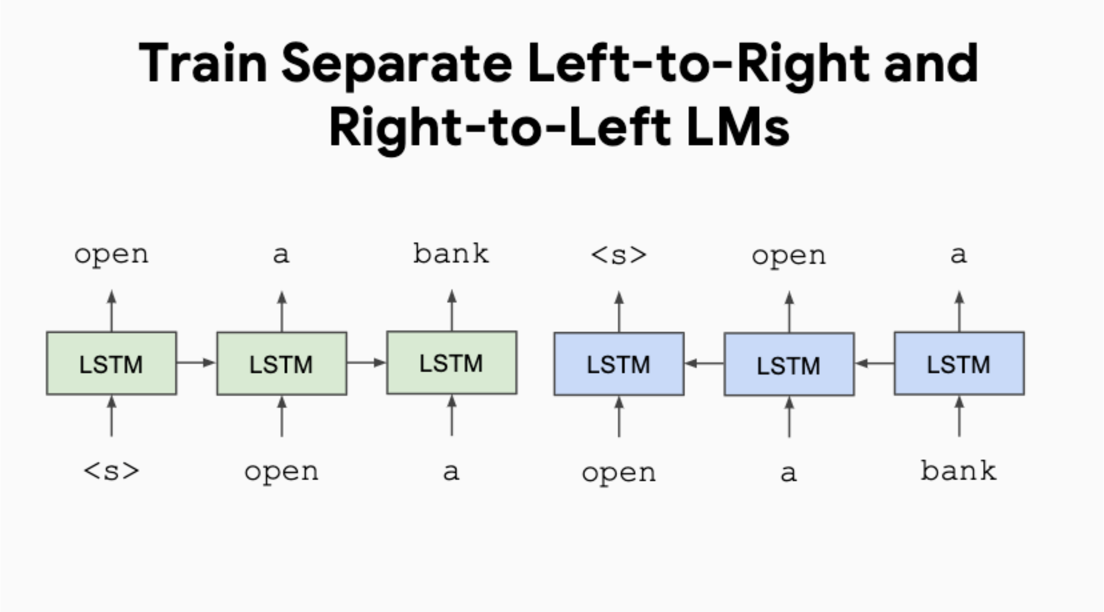
Self-supervised learning

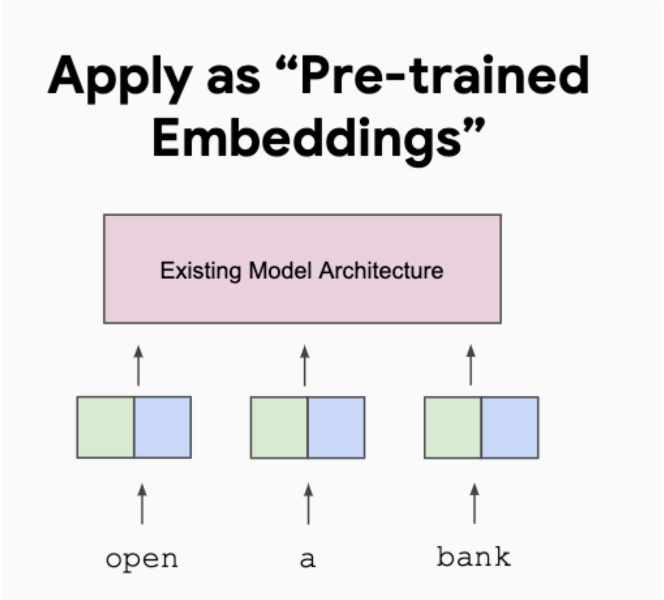


Prior Work: ELMo

ELMo (Peters et al., 2018; NAACL 2018 best paper)

- Train two separate **unidirectional** LMs (left-to-right and right-to-left) based on **LSTMs**
- **Feature-based** approach: pre-trained representations used as input to task-specific models
- Trained on **single sentences** from 1B word benchmark (Chelba et al., 2014)

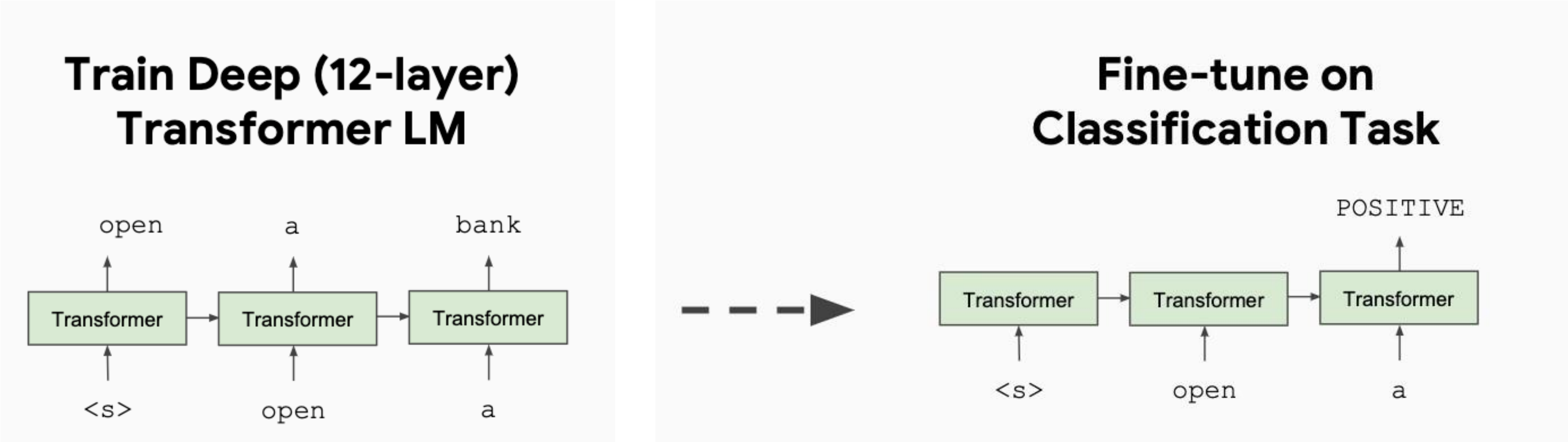




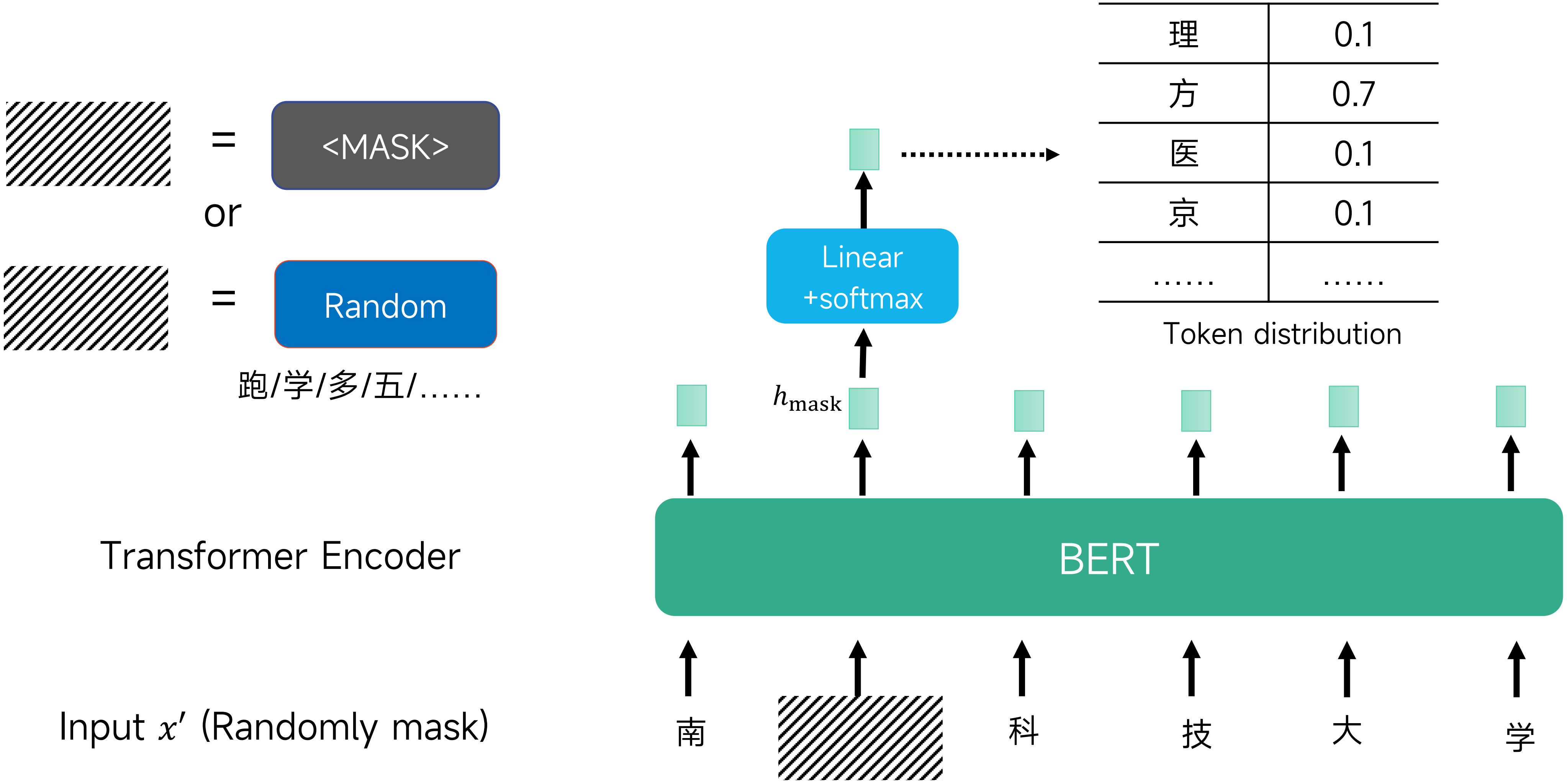
Prior Work: OpenAI GPT

OpenAI GPT (Radford et al., 2018; released in 2018/6)

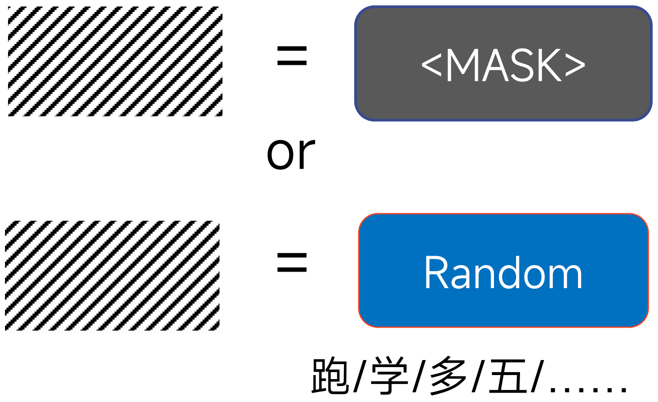
- Train one unidirectional LM (left-to-right) based on a deep **Transformer decoder**
- **Fine-tuning** approach: all pre-trained parameters are re-used & updated on downstream tasks
- Trained on 512-token segments on BooksCorpus — much **longer** context!

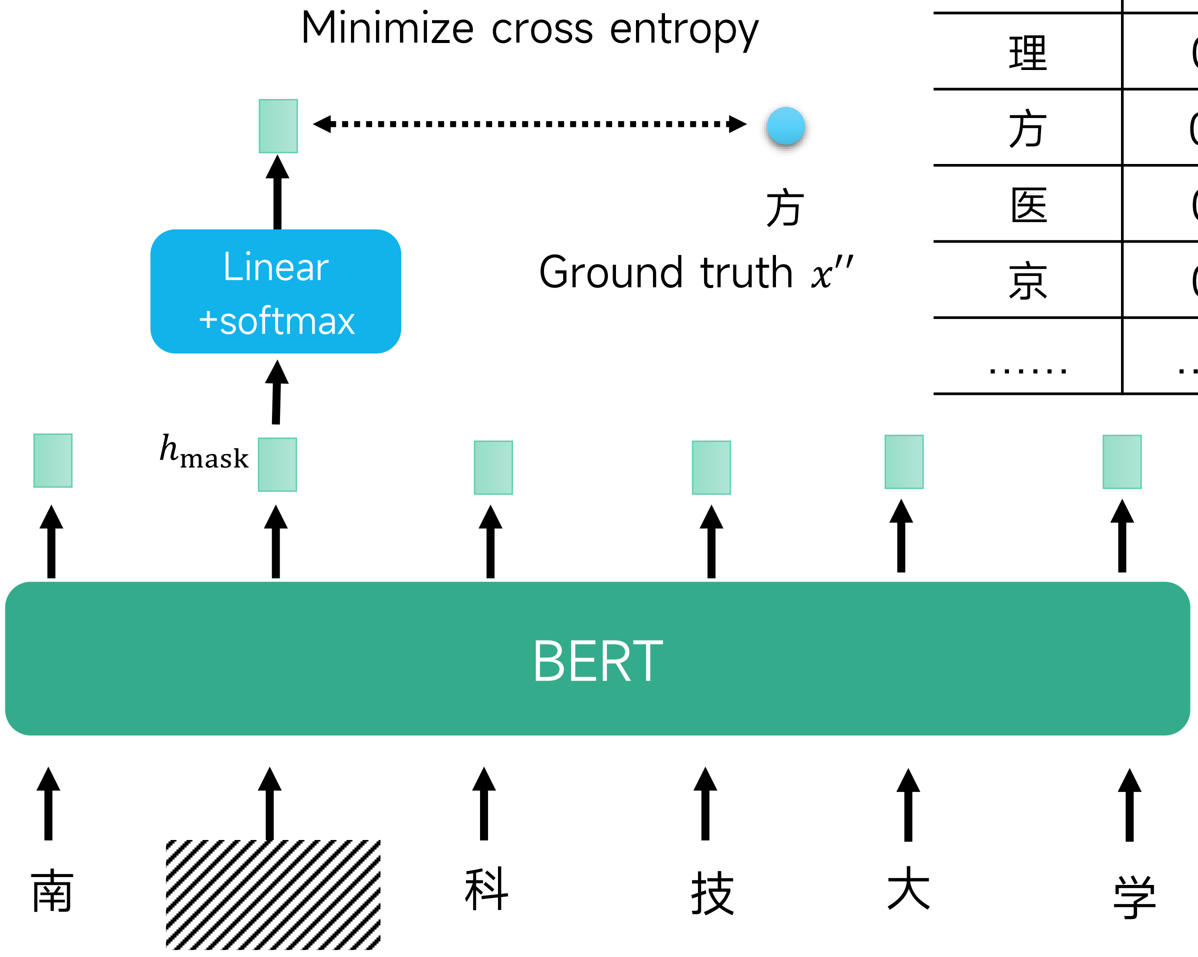


Masked Token Prediction



Masked Token Prediction





Token	Pred	Label
理	0.1	0
方	0.7	1
医	0.1	0
京	0.1	0
……	……	……

Transformer Encoder

Input x' (Randomly mask)

MLM: 80-10-10 Corruption

For the 15% predicted words,

- 80% of the time, they replace it with [MASK] token

  went to the store ⇨ went to the [MASK]

- 10% of the time, they replace it with a random word in the vocabulary

  went to the store ⇨ went to the running

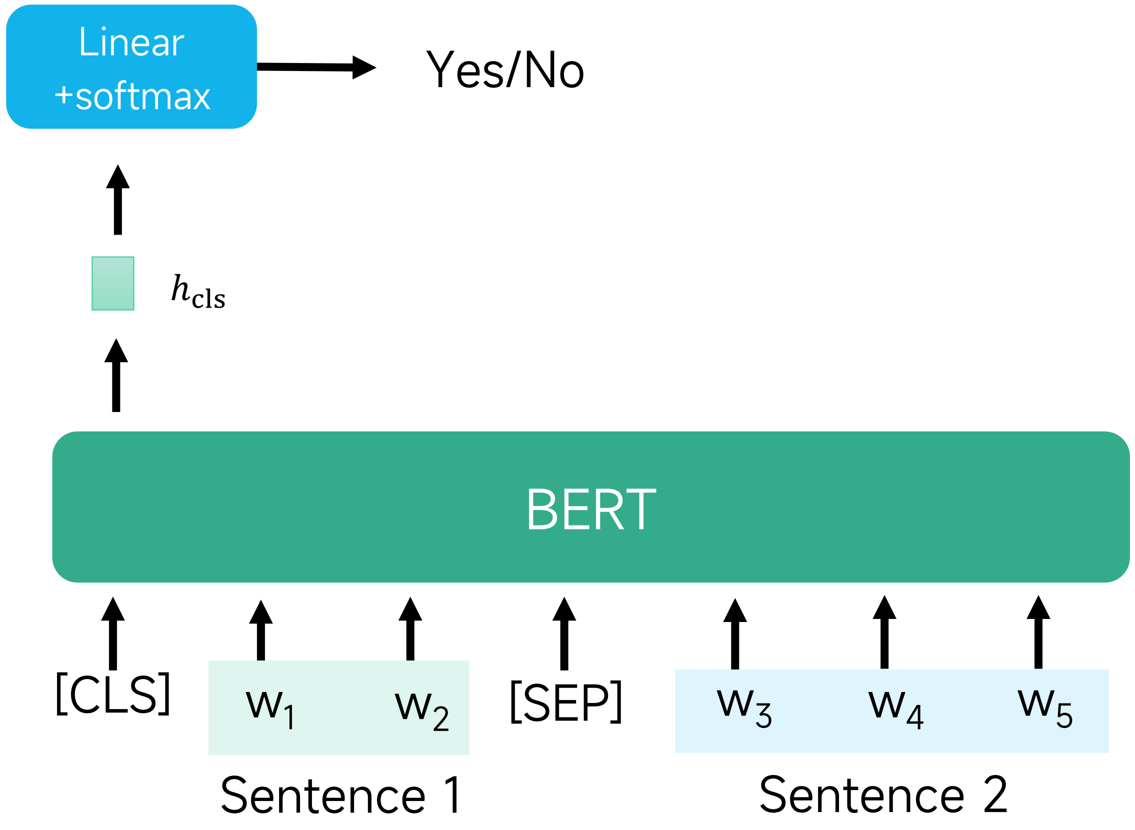
- 10% of the time, they keep it unchanged

  went to the store ⇨ went to the store

Why?

Because [MASK] tokens are never seen during fine-tuning

Next Sentence Prediction



- This approach is not helpful.
  Observed in Roberta

- **SOP**: Sentence order prediction
  Used in ALBERT

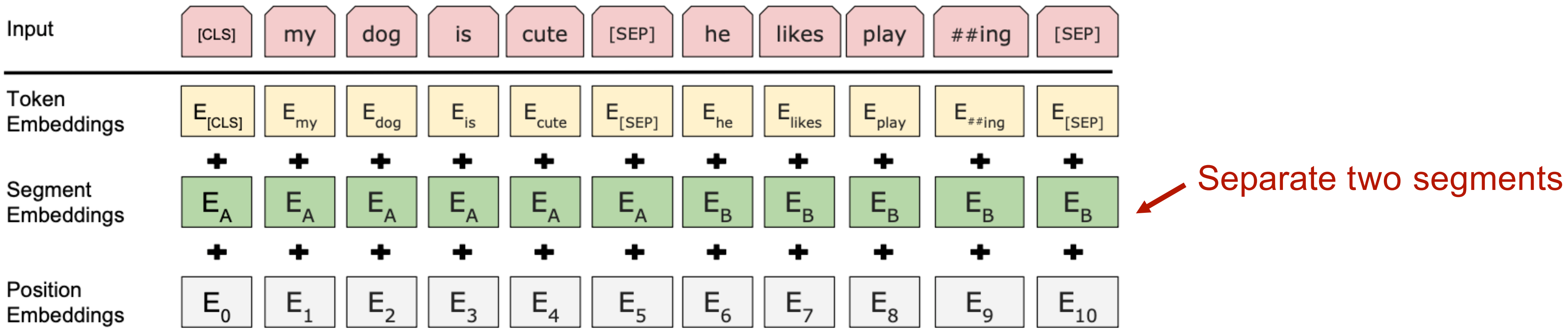
https://arxiv.org/abs/1907.11692

https://arxiv.org/abs/1909.11942

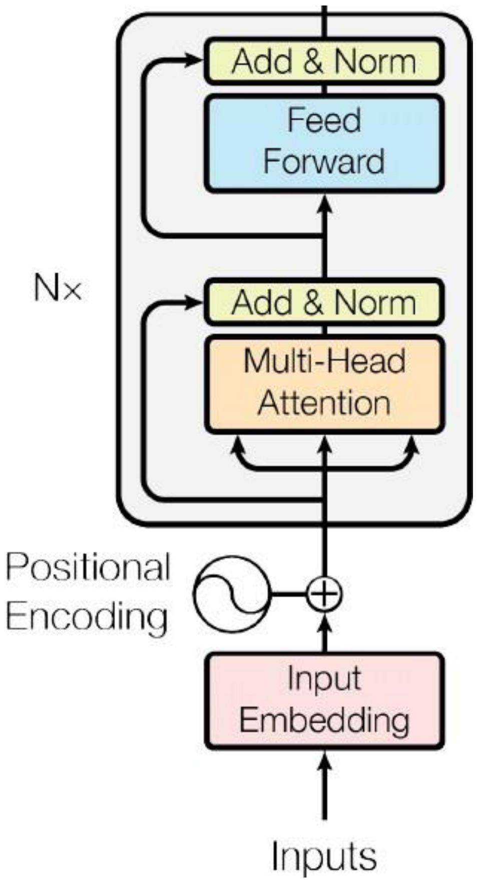
BERT Embedding

- Vocabulary size: 30,000 wordpieces (common sub-word units) (Wu et al., 2016)

- Input embeddings:



BERT Pretraining



- BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters

Same as OpenAI GPT

- BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters

OpenAI GPT was trained on BooksCorpus only!

- Training corpus: Wikipedia (2.5B) + BooksCorpus (0.8B)
- Max sequence size: 512 wordpieces (roughly 256 and 256 for two non-contiguous sequences)
- Trained for 1M steps, batch size 128k

Pretrain-and-Finetune Paradigm

Pretrain once, finetune many times

Pretrain

Self-supervised Learning

BERT

- Masked token prediction
- Next sentence prediction

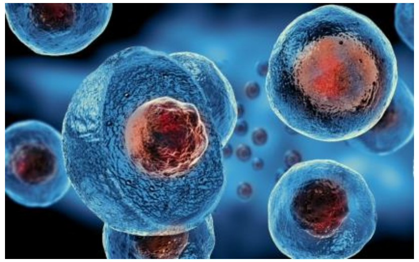
Supervised Learning

Finetune

Model for Task 1

Model for Task 2

Model for Task 3



Embryonic stem cells

Downstream Tasks

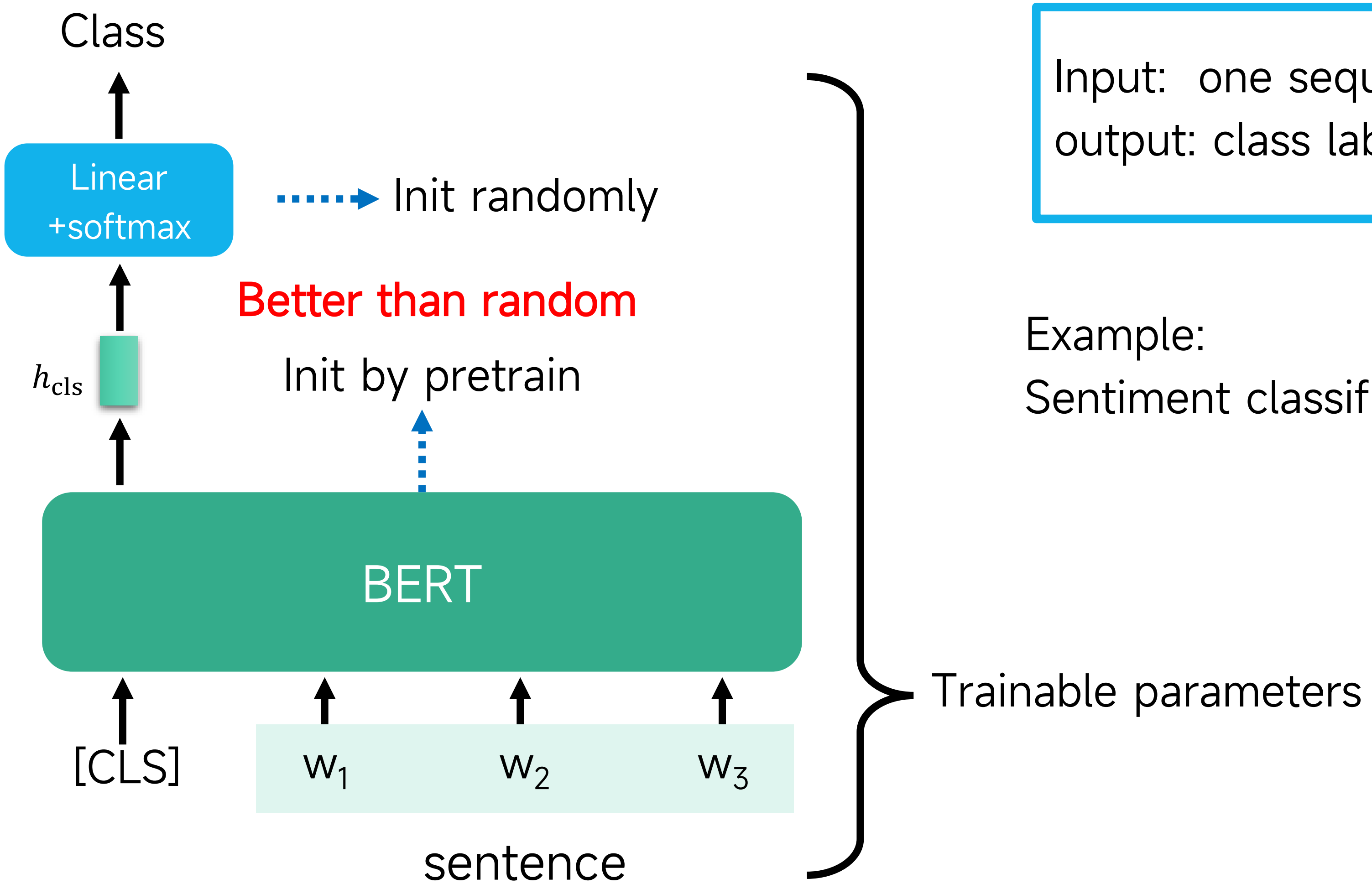
- The tasks we care
- We have a little bit of labeled data

Different Downstream Tasks

- Text classification
- Sequence labeling
- Matching
- Extractive question answering

Pretrained Encoder

Case 1: Text Classification



Input: one sequence
output: class label

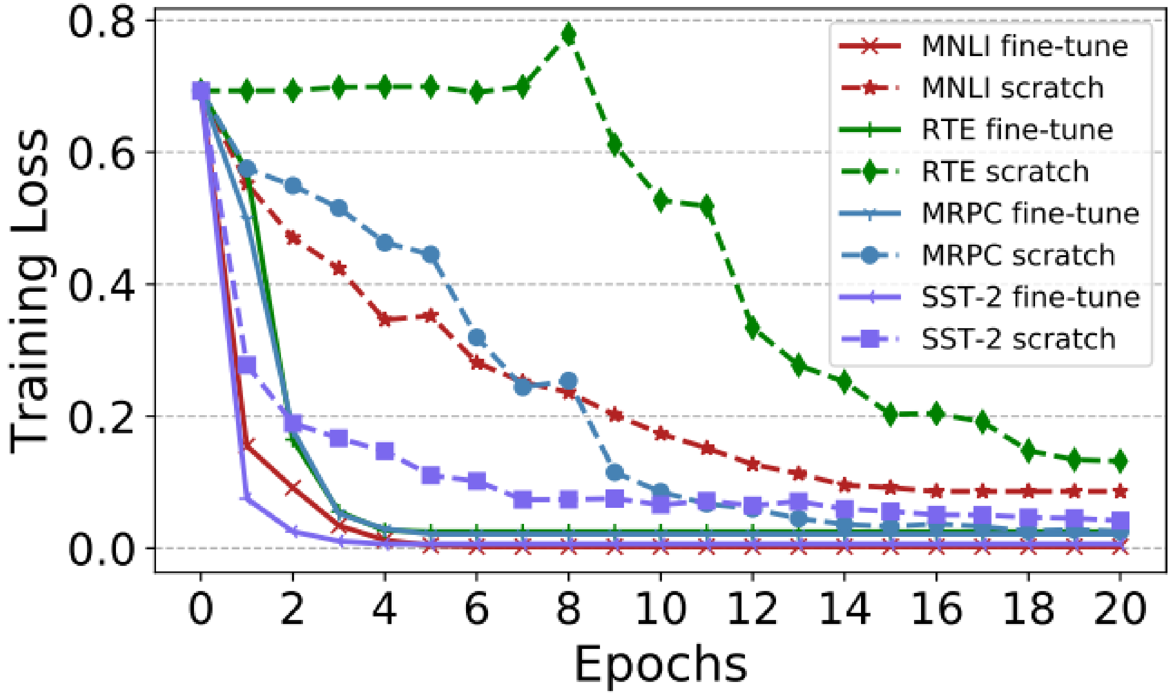
Example:
Sentiment classification

The food is good
↓
Positive

Pretrain vs. Random Initialization

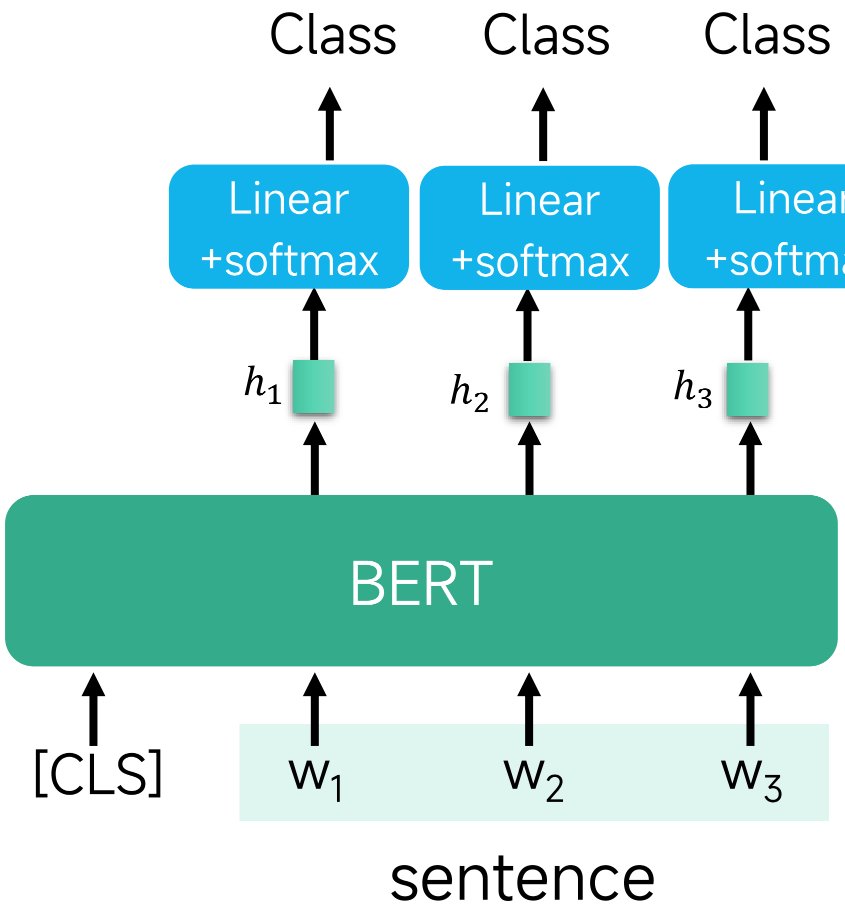
(finetune) (scratch)

Which one is better? Why?



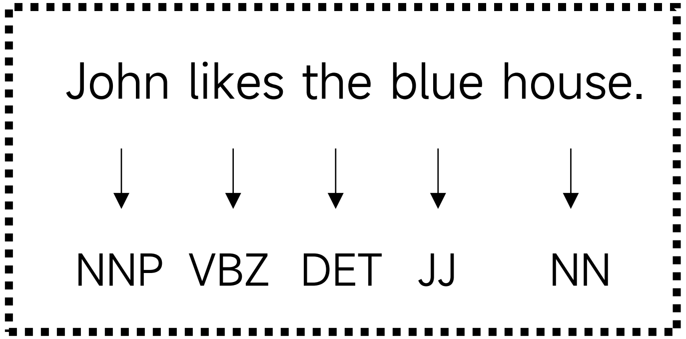
Source of image: https://arxiv.org/abs/1908.05620

Case 2: Sequence Labeling

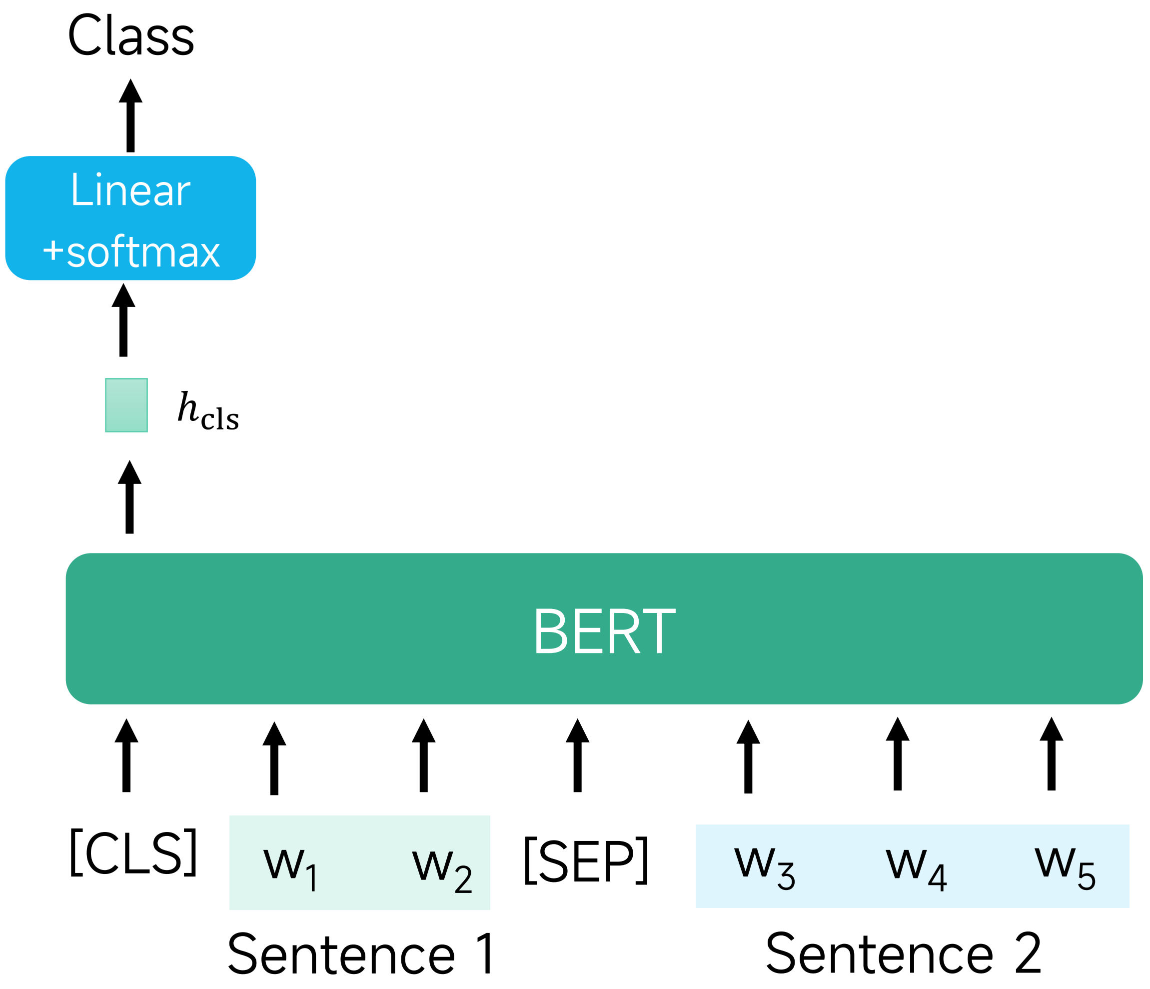


Input: one sequence
Output: one label sequence

Example:
POS tagging



Case 3: Matching



Input: two sequences
output: class label

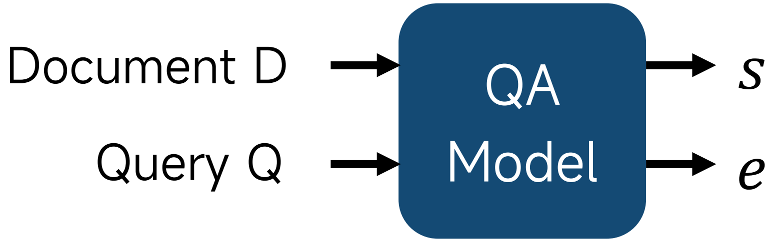
Example:

Natural Language Inference (NLI)

Premise: A person on a horse jumps over a stone.
Hypothesis: A person is at a diner.
Output: Contradiction

Case 4: Extractive Question Answering

Input: two sequences
Output: answer



Output: two integers (s, e)

Answer: $A = \{d_s, \cdots, d_e\}$

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. 17 main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail… Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice cry 77 **within a cloud**. 79, intense periods of … in scattered lo… are called "showers".

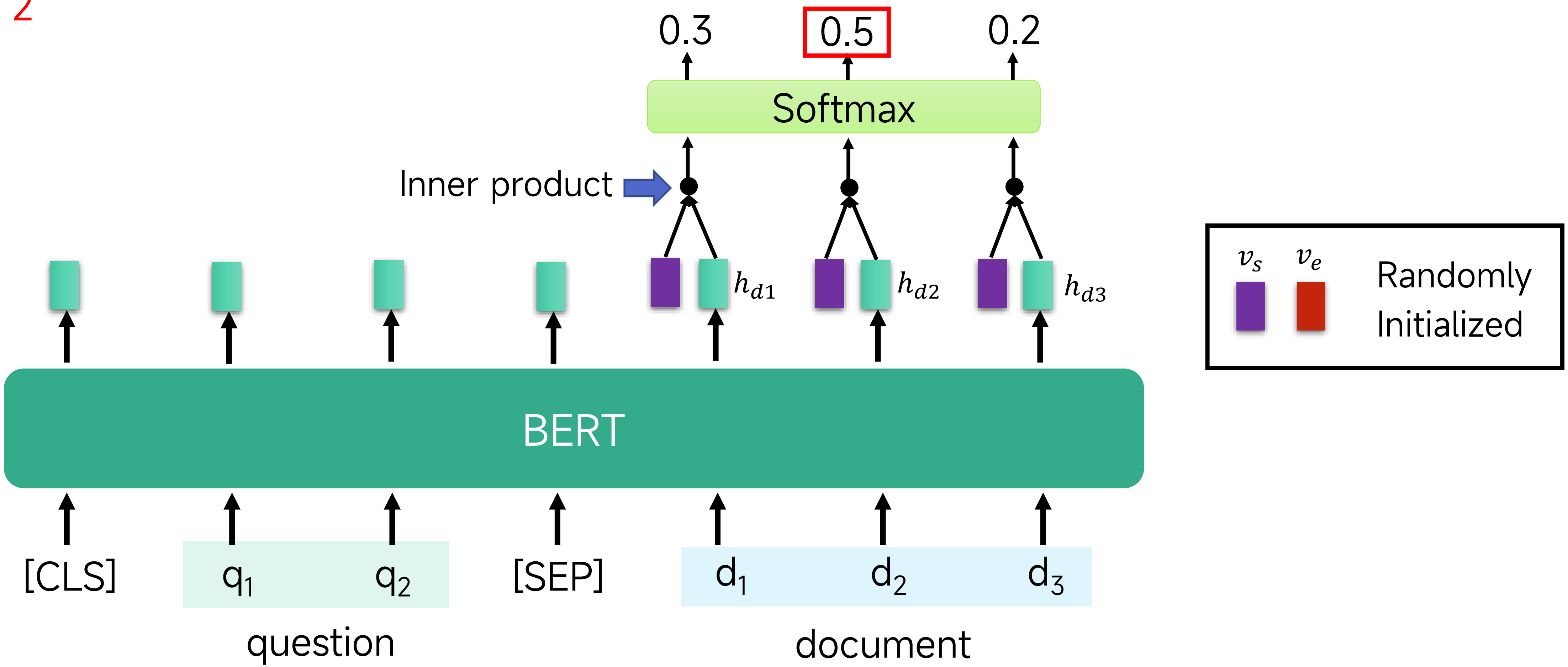
What causes precipitation to fall?
gravity $s = 17, e = 17$

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud $s = 77, e = 79$

Case 4: Extractive Question Answering

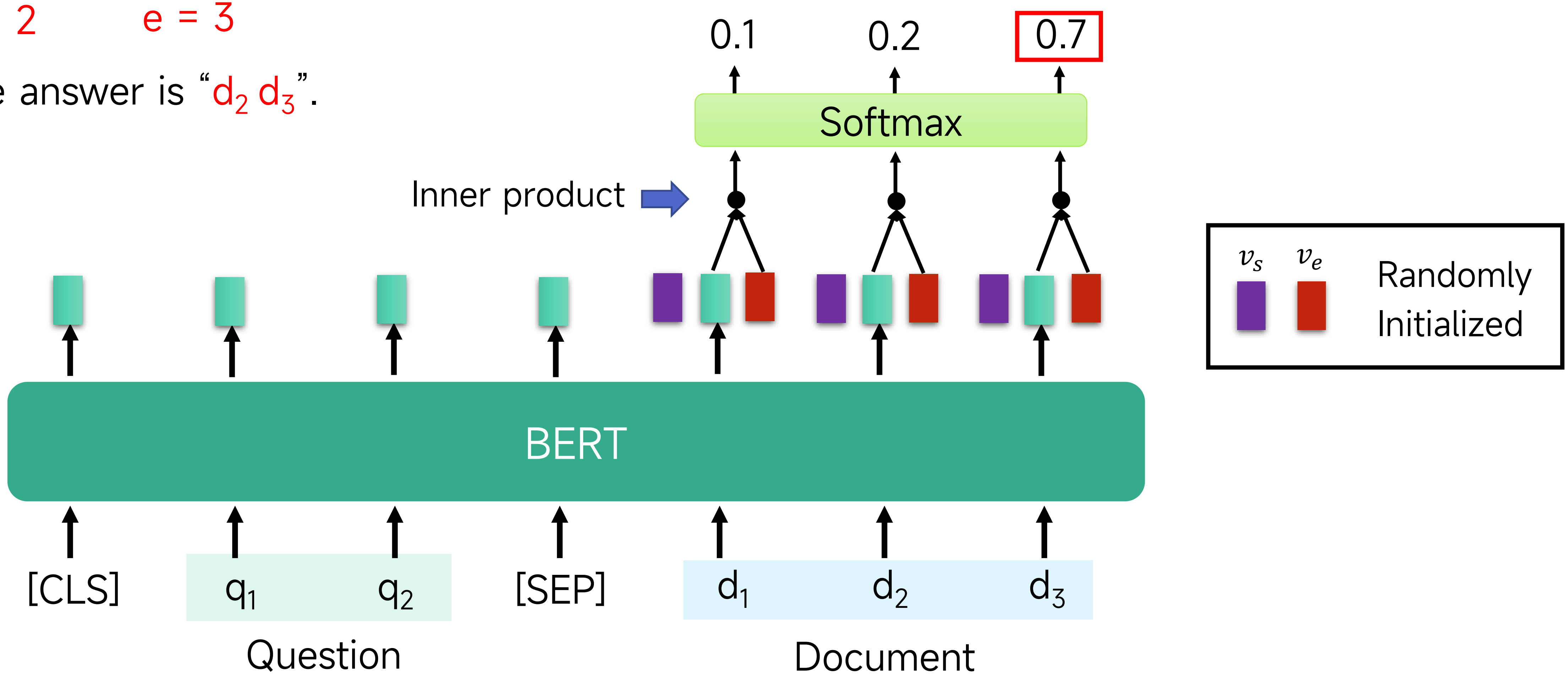
s = 2



Case 4: Extractive Question Answering

$s = 2$ $e = 3$

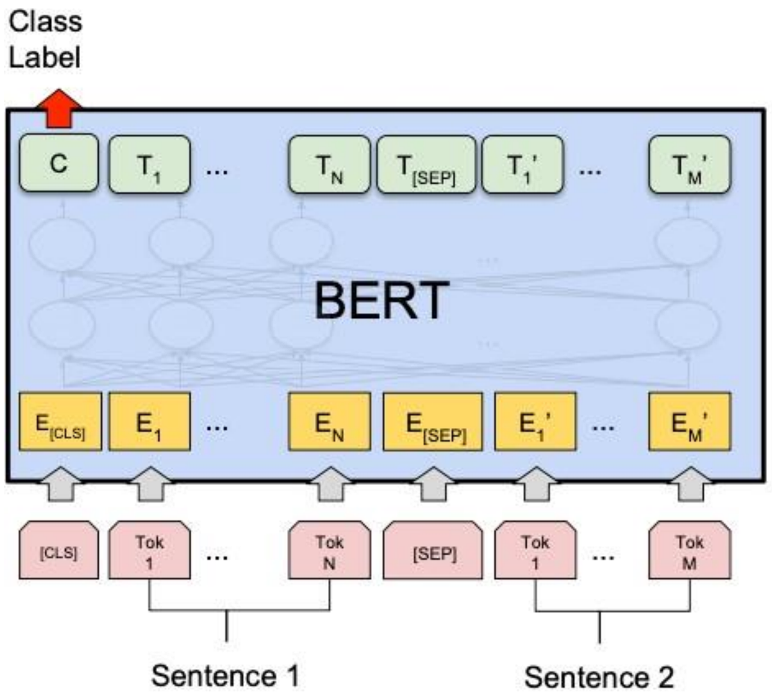
The answer is "d_2 d_3".



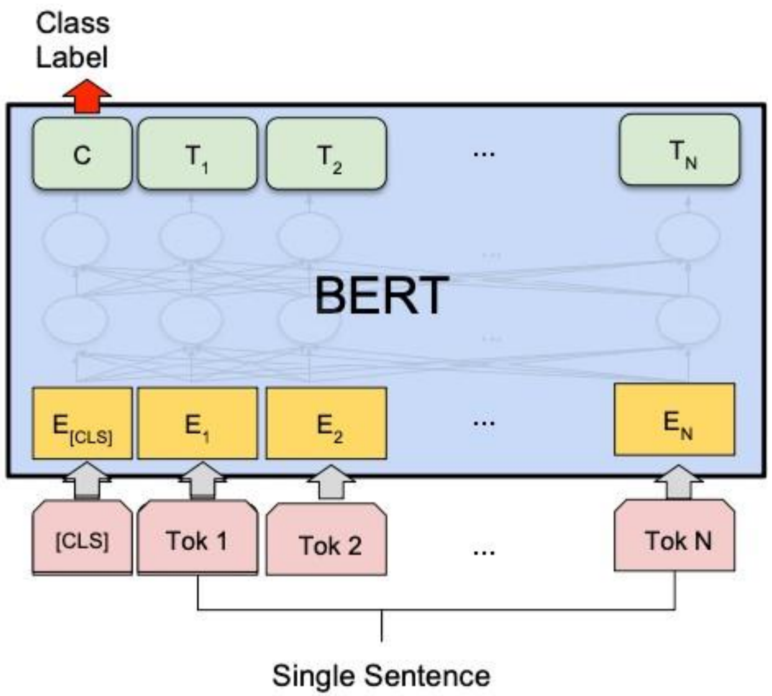
Fine-tuning BERT

"Pretrain once, finetune many times."

sentence-level tasks



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

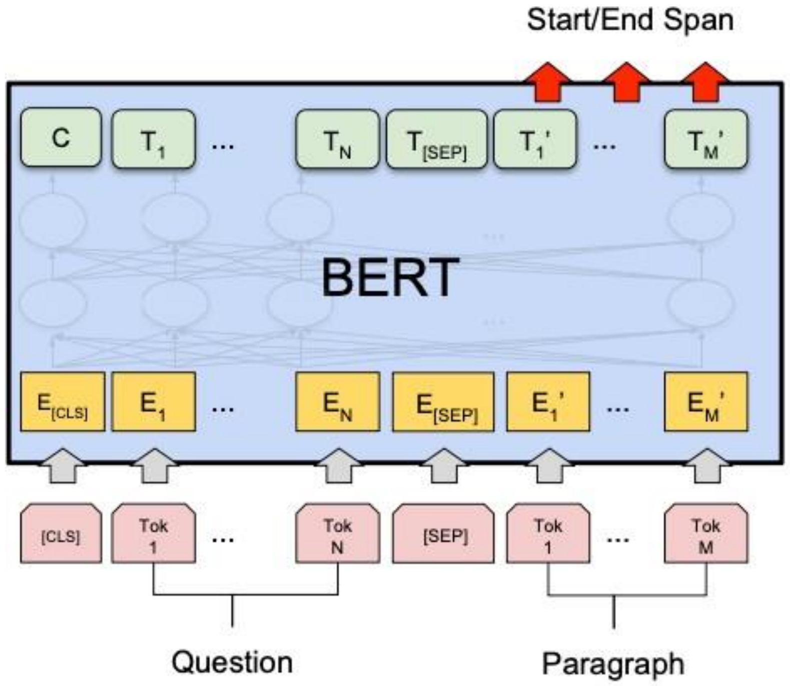


(b) Single Sentence Classification Tasks: SST-2, CoLA

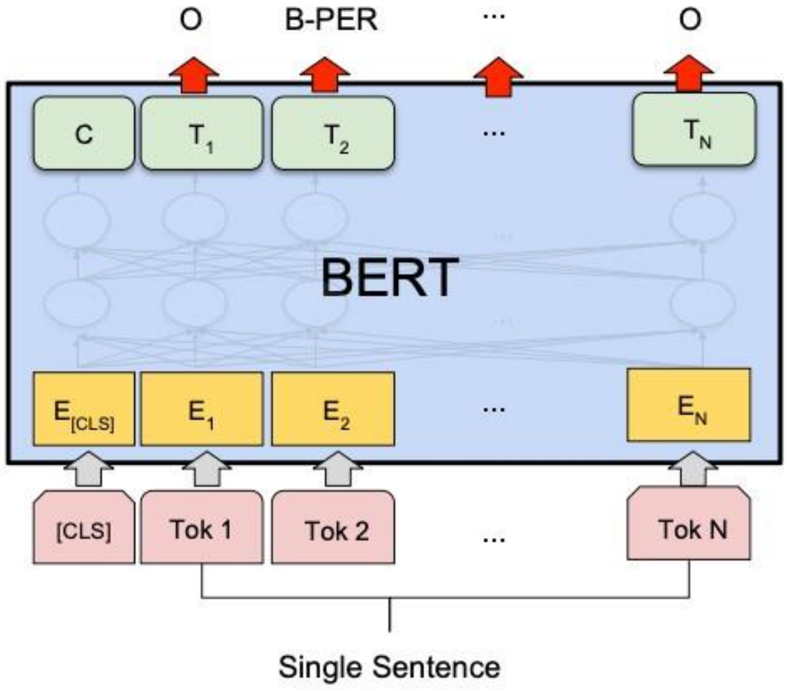
Fine-tuning BERT

“Pretrain once, finetune many times.”

token-level tasks



(c) Question Answering Tasks:
SQuAD v1.1



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

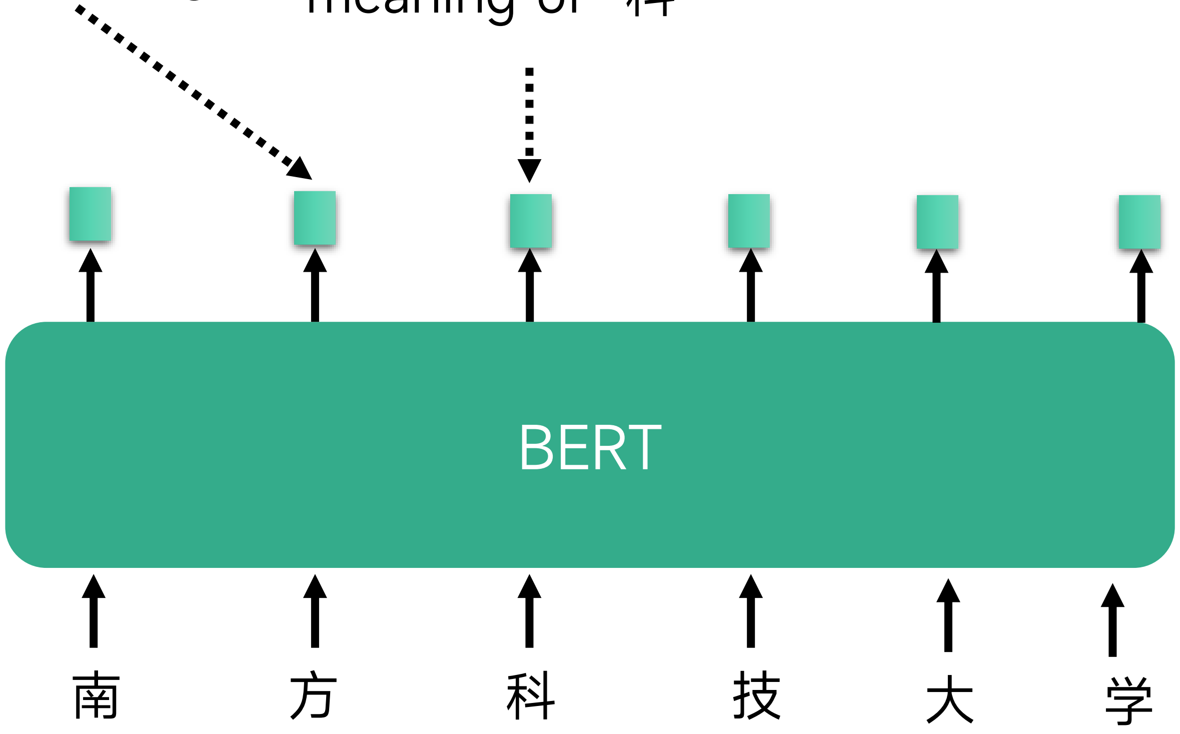
Experimental Results: GLUE

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**

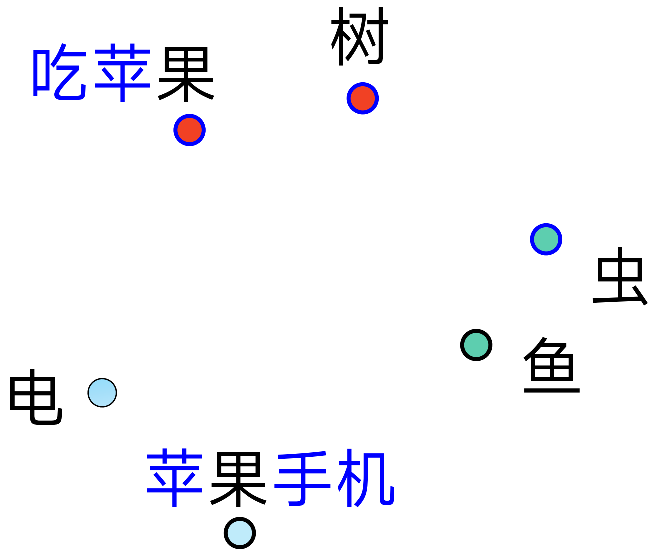
Why does BERT Work?

Embedding

Represent the meaning of “科”

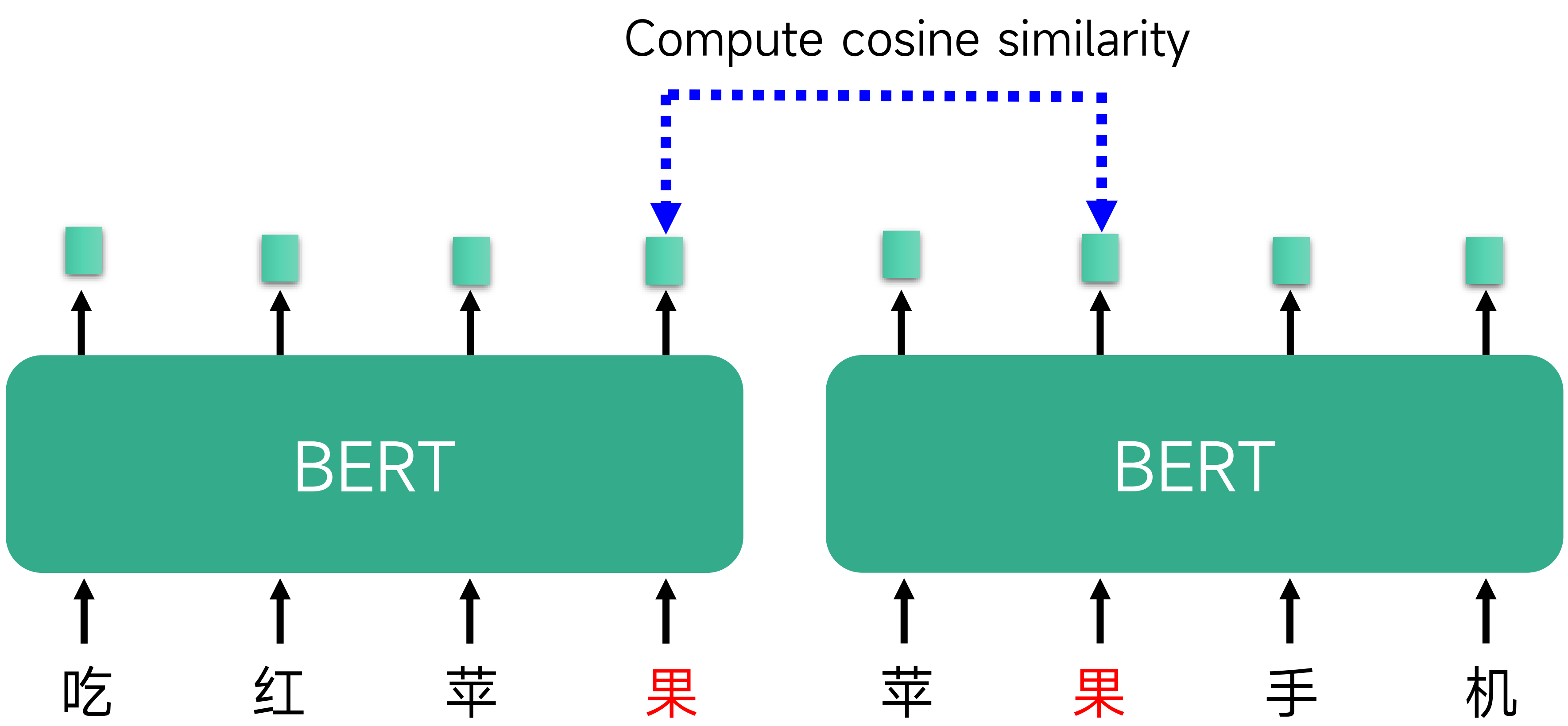


The tokens with similar meaning have similar embedding



Context is considered

Why does BERT Work?



Why does BERT Work?

今天买了苹果吃。

进口苹果每公斤平均下跌10.2%。

她不喜欢喝苹果奶茶。

新鲜苹果马上就要上市了。

漫山遍野都是苹果树。

苹果下个月月初发布新款iphone
。

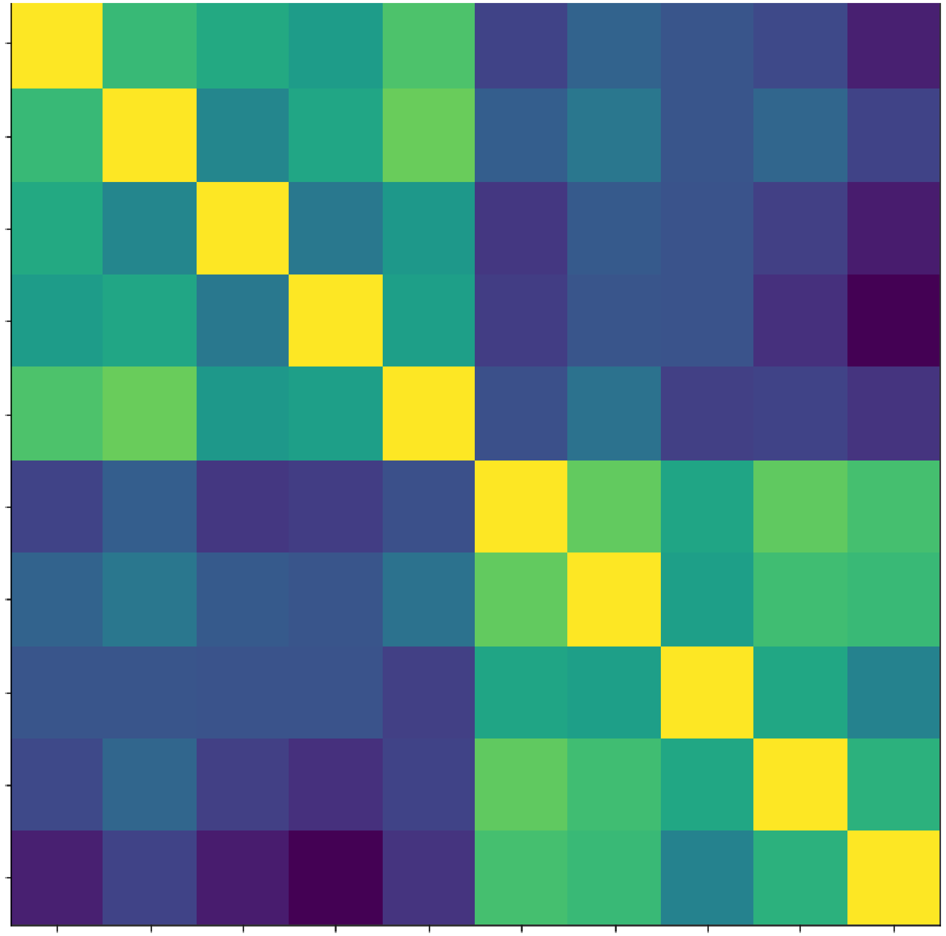
苹果新建团队研发元宇宙技术。

苹果获得了面部识别的新专利。

今天我去商场买了苹果手机。

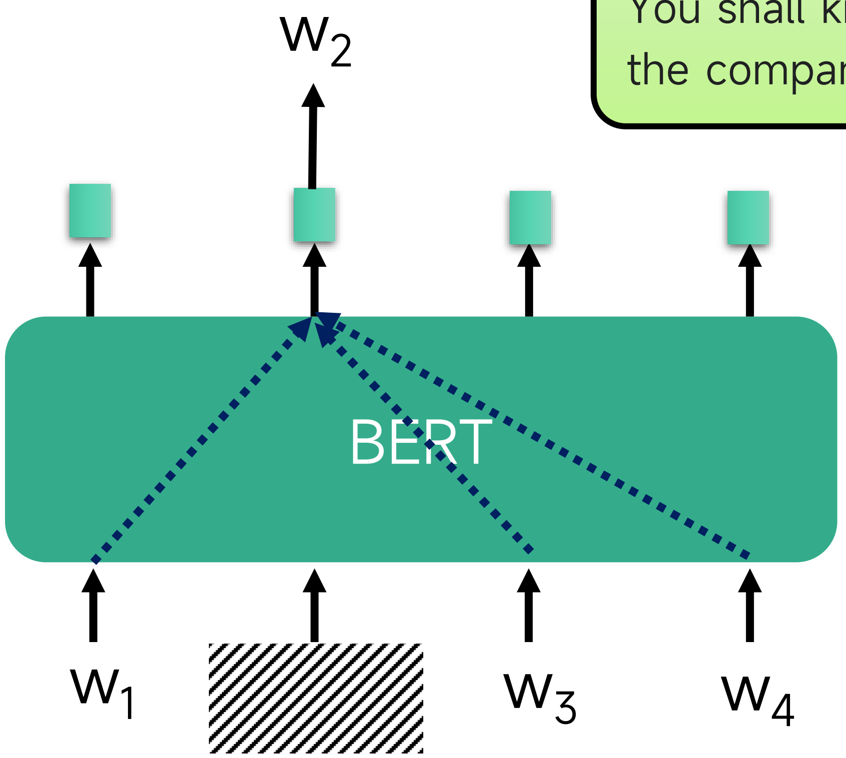
苹果的股价又涨了。

Cosine similarities of BERT embeddings



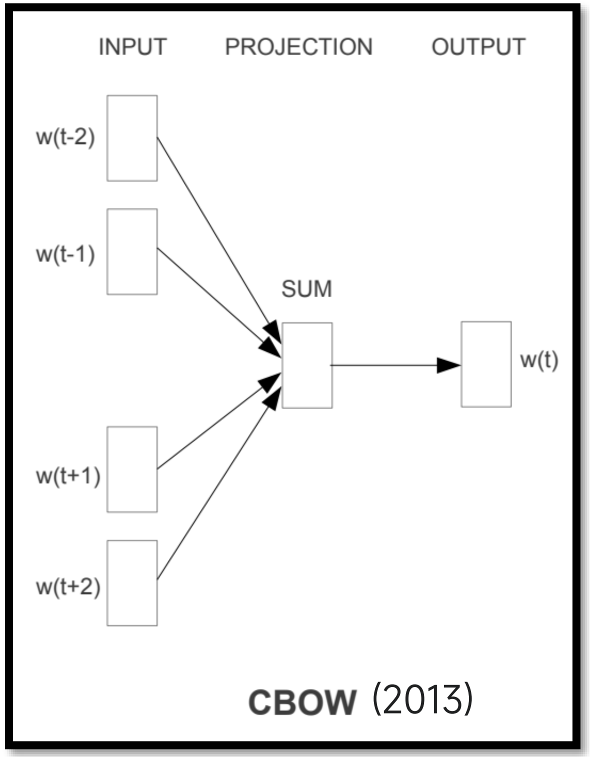
Why does BERT work?

Contextualized word embedding



You shall know a word by the company it keeps

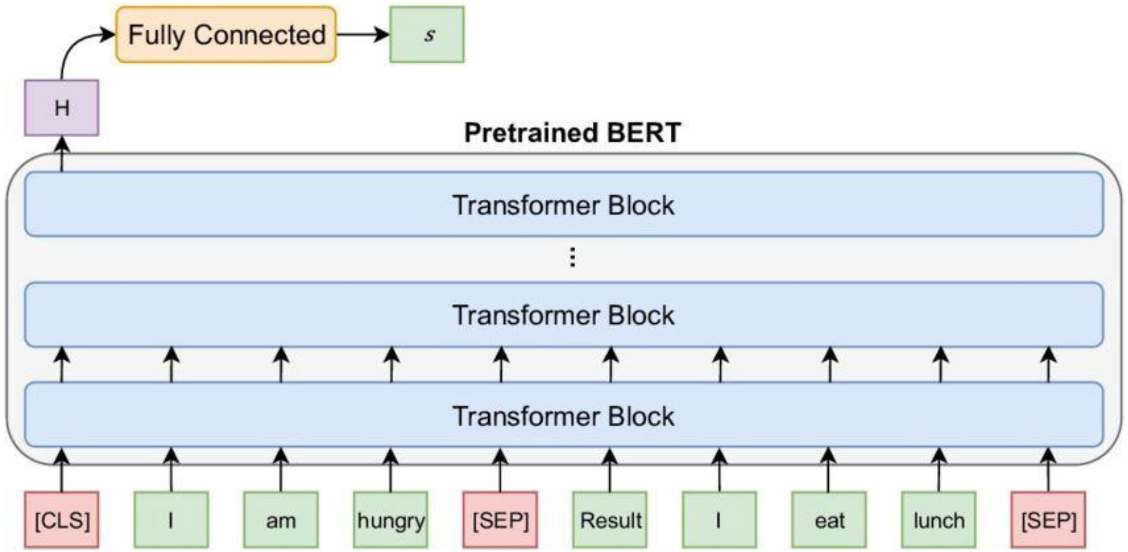
John Rupert Firth (1960s)



Word2vec

Fully Finetuning

- Pretrain a language model on task
- Attach a small task specific layer
- Fine-tune the weights of full NN by propagating gradients on a downstream task



[Devlin et al. 2019]

Parameter-Efficient Finetuning

- With standard fine-tuning, we need to make a new copy of the model for each task
- In the extreme case of a different model per user, we could never store 1000 different full models
- If we fine tuned a subset of the parameters for each task, we could alleviate storage costs
- This is parameter-efficiency

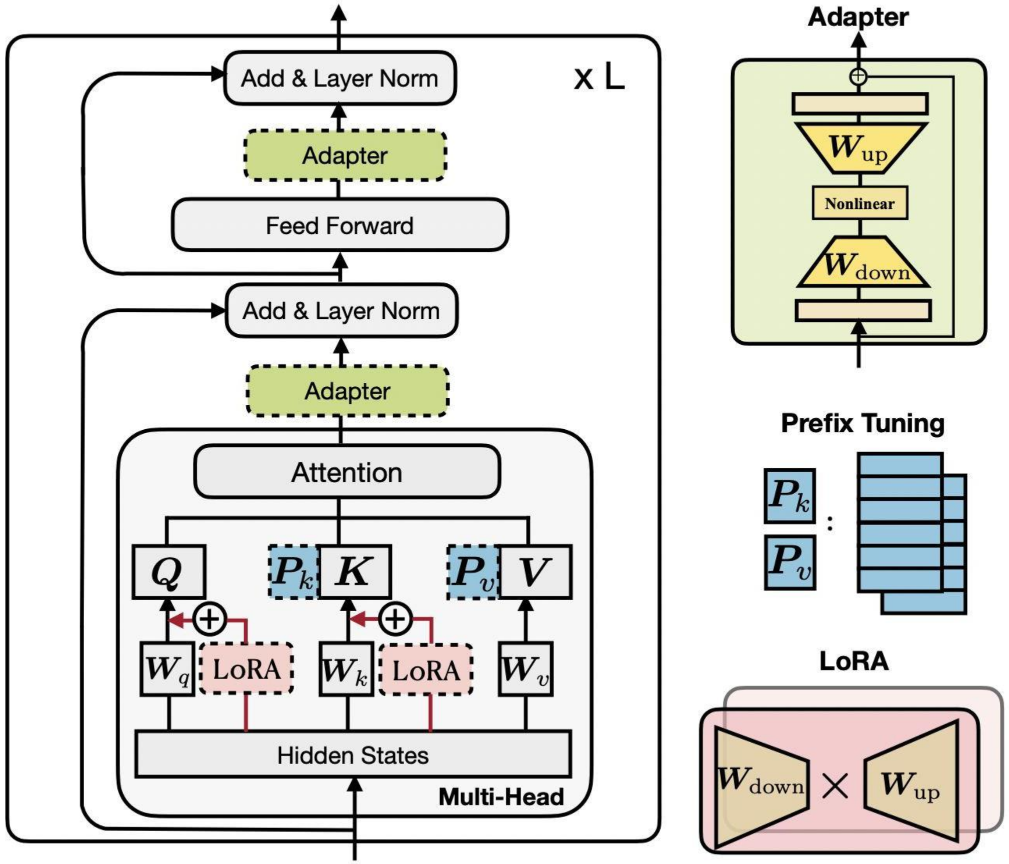


Image: (He et al. 2022)

Tuning the Model

Whether the pretrained models are trained

- Fully finetune (trained)
- Parameter-efficient finetune (fixed)
  - Additional parameters are trained

Adapter-based	LoRA-based	Prefix-tuning
Prompt-tuning	P-Tuning	BitFit

Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing

Pengfei Liu
Carnegie Mellon University
pliu3@cs.cmu.edu

Weizhe Yuan
Carnegie Mellon University
weizhey@cs.cmu.edu

Jinlan Fu
National University of Singapore
jinlanjonna@gmail.com

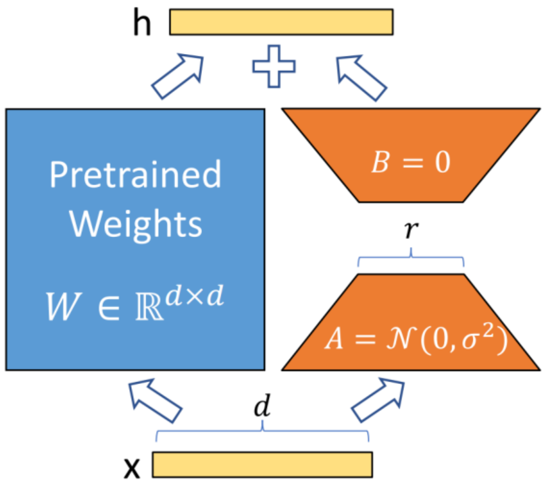
Zhengbao Jiang
Carnegie Mellon University
zhengbaj@cs.cmu.edu

Hiroaki Hayashi
Carnegie Mellon University
hiroakih@cs.cmu.edu

Graham Neubig
Carnegie Mellon University
gneubig@cs.cmu.edu

LoRA Finetuning

- Low-rank adaptation
- Hypothesizes that the change of weights during model tuning has a low intrinsic rank
- Optimize the low-rank decomposition for the change of original weight matrices in the self-attention modules.



LoRA $(W = W_0 + \gamma BA)$

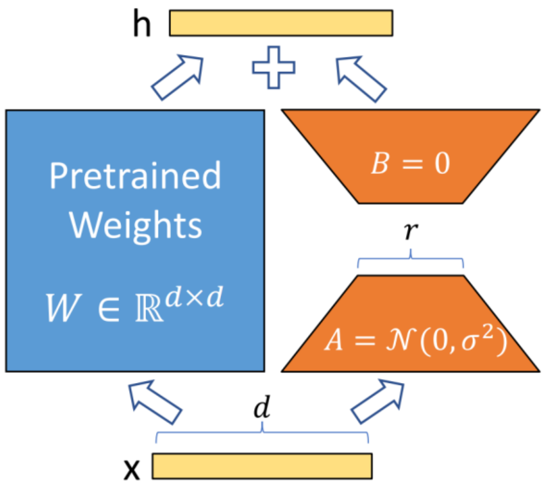
$B \in R^{d \times r}, A \in R^{r \times d}$

[Parameter-Efficient LLM Finetuning With Low-Rank Adaptation (LoRA) - Lightning AI]

LoRA Finetuning

- Code illustration

```
input_dim = 768  # e.g., the hidden size of the pre-trained model
output_dim = 768  # e.g., the output size of the layer
rank = 8  # The rank 'r' for the low-rank adaptation

W = ... # from pretrained network with shape input_dim x output_dim


W_A = nn.Parameter(torch.empty(input_dim, rank)) # LoRA weight A
W_B = nn.Parameter(torch.empty(rank, output_dim)) # LoRA weight B


# Initialization of LoRA weights
nn.init.kaiming_uniform_(W_A, a=math.sqrt(5))
nn.init.zeros_(W_B)


def regular_forward_matmul(x, W):
    h = x @ W
return h


def lora_forward_matmul(x, W, W_A, W_B):
    h = x @ W  # regular matrix multiplication
    h += x @ (W_A @ W_B)*alpha # use scaled LoRA weights
return h
```



LoRA Finetuning

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Code example

```
import peft
from transformers import AutoModelForCausalLM, AutoTokenizer
from peft import LoraConfig, get_peft_model, PeftModel


lora_config = LoraConfig(
    r=4, #As bigger the R bigger the parameters to train.
    lora_alpha=1, # a scaling factor that adjusts the magnitude of the weight
matrix. Usually set to 1
    target_modules=["query_key_value"], #You can obtain a list of target modules
in the URL above.
    lora_dropout=0.05, #Helps to avoid Overfitting.
    bias="lora_only", # this specifies if the bias parameter should be trained.
    task_type="CAUSAL_LM"
)


model_name = "Qwen/Qwen2.5-1.5B-Instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
foundation_model = AutoModelForCausalLM.from_pretrained(model_name)


peft_model = get_peft_model(foundation_model, lora_config)
print(peft_model.print_trainable_parameters())
```

LoRA Finetuning

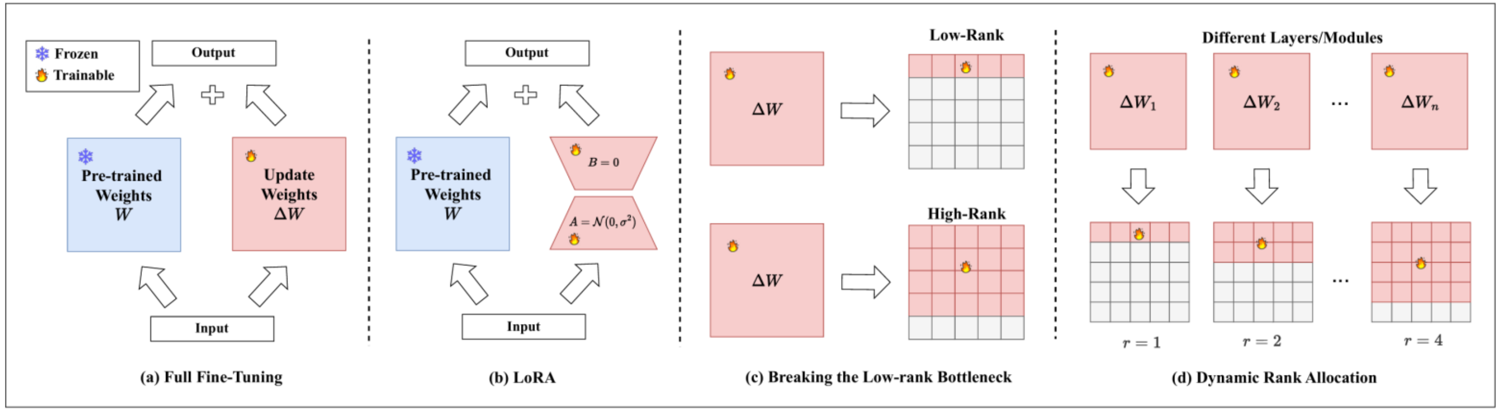


Fig. 2 An illustration of full fine-tuning (a), LoRA (b) and its variants for improving downstream adaptation, which includes breaking the low-rank bottleneck (c) and dynamic rank allocation (d).

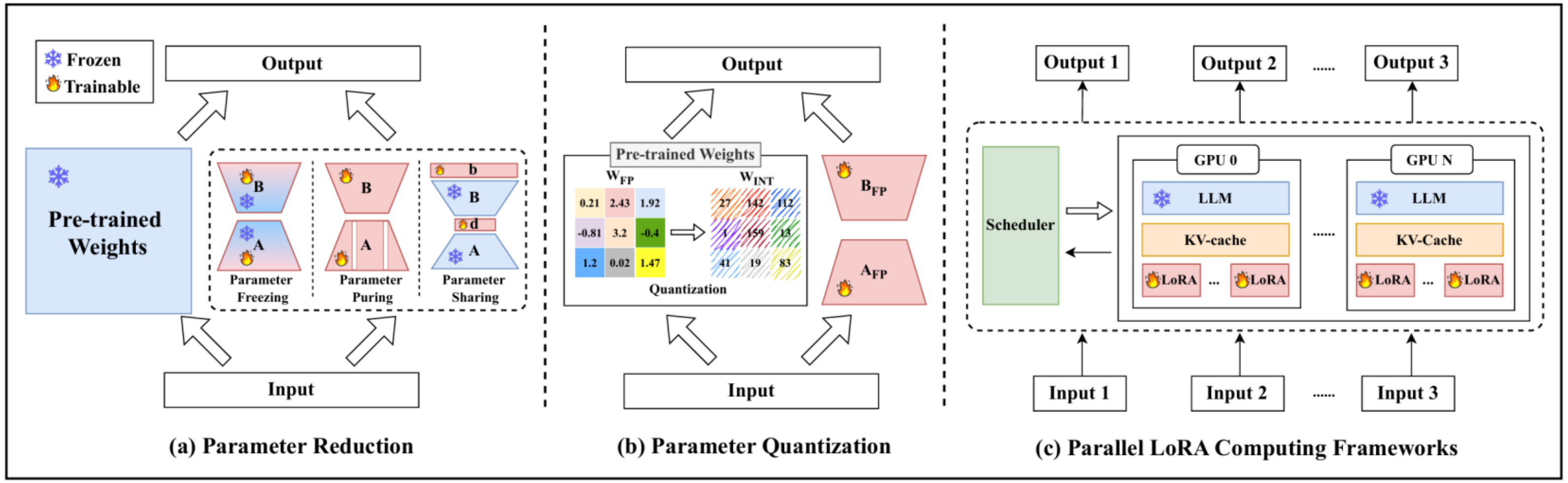


Fig. 4 An illustration of efficiency improving methods.

[2407.11046] A Survey on LoRA of Large Language Models

What Happened after BERT?

Lots of people are trying to understand what BERT has learned and how it works

A Primer in BERTology: What We Know About How BERT Works

Anna Rogers
Center for Social Data Science
University of Copenhagen
arogers@sodas.ku.dk

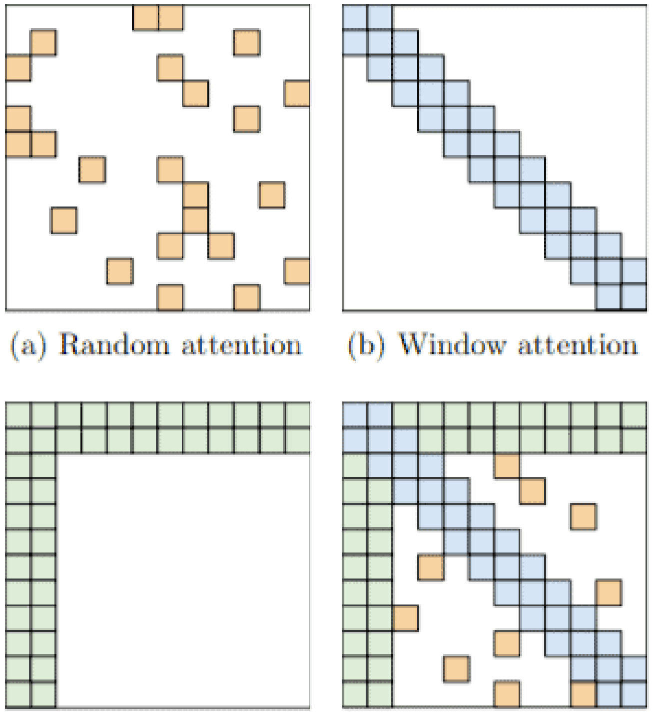
Olga Kovaleva
Dept. of Computer Science
University of Massachusetts Lowell
okovalev@cs.uml.edu

Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

- Syntactic knowledge, semantic knowledge, world knowledge…
- How to mask, what to mask, where to mask, alternatives to masking..

What Happened after BERT?

- Models that handle long contexts (much more than 512 tokens)
  - Longformer, Big Bird, …
- Multilingual BERT
  - mBERT: Trained single model on 104 languages from Wikipedia. Shared 110k WordPiece vocabulary
  - XLM-R: 100 languages with 250k sentencepiece vocabulary.
- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT, …
- Making BERT smaller to use
  - DistillBERT, TinyBERT, …



(a) Random attention (b) Window attention (c) Global Attention (d) BigBird

Image from the original paper

What Happened after BERT?

- RoBERTa (Liu et al., 2019)
  - Trained on 10x data & longer, no NSP
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
  - Still one of the most popular models to date
- ALBERT (Lan et al., 2020)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower..
- ModernBERT (Warner et al., 2024)
  - Train on 2 trillion tokens with a native 8192 sequence length
  - RoPE, Pre-Norm, GeGLU

AnswerDotAI/ModernBERT: Bringing BERT into modernity via both architecture changes and scaling

T5 Model

- T5 refers to "Text-to-Text Transfer Transformer"

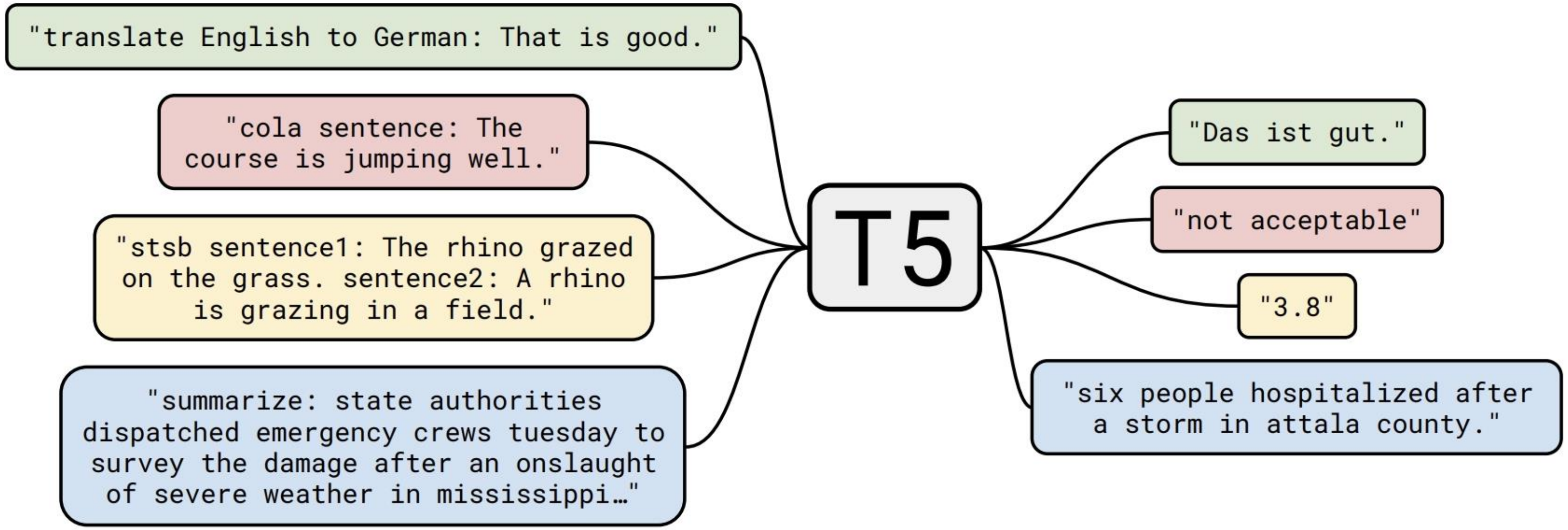


Figure 1: A diagram of our text-to-text framework. Every task we consider—including

C4 Dataset

- Colossal Clean Crawled Corpus (750 GB)
- Clean the data from Common Crawl (20TB data each month)

- We only retained lines that ended in a terminal punctuation mark (i.e. a period, exclamation mark, question mark, or end quotation mark).
- We discarded any page with fewer than 3 sentences and only retained lines that contained at least 5 words.
- We removed any page that contained any word on the "List of Dirty, Naughty, Obscene or Otherwise Bad Words".[6]
- 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.
- Since some of the scraped pages were sourced from Wikipedia and had citation markers (e.g. [1], [citation needed], etc.), we removed any such markers.
- Many pages had boilerplate policy notices, so we removed any lines containing the strings "terms of use", "privacy policy", "cookie policy", "uses cookies", "use of cookies", or "use cookies".
- To deduplicate the data set, we discarded all but one of any three-sentence span occurring more than once in the data set.

T5 Input and Output Format

- Train a single model on the diverse set of tasks
- Cast all of the tasks we consider into a "text-to-text" format
  - "[Task-specific prefix]: [Input text]" -> "[output text]"
  - Add a task-specific (text) prefix to the original input sequence
  - For translation, add the sequence "translate English to German: That is good."
  - For classification, simply predicts a single word corresponding to the target label
- The choice of text prefix used for a given task is essentially a hyperparameter
- Provides a consistent training objective both for pre-training and fine-tuning

T5 Pretraining

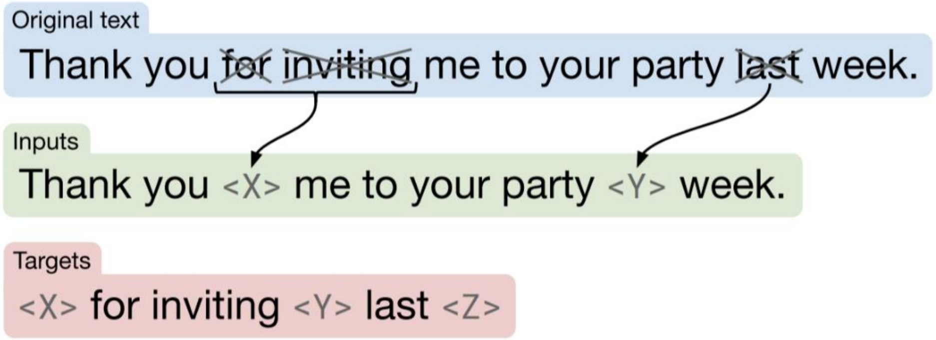
- Pretrain a standard Transformer using a simple denoising objective and then separately fine-tune on each of our downstream tasks
- A model with about 220 million parameters. Roughly twice the number of parameters of Bert-base
- Use an "inverse square root" learning rate schedule:

$$\frac{L_r}{\sqrt{\max(n,k)}}$$

  - Where n is the current training iteration and k is the number of warm-up steps (set to 10^4)
  - Sets a constant learning rate of 0.01 for the first 10^4 steps, then exponentially decays the learning rate until pre-training is over

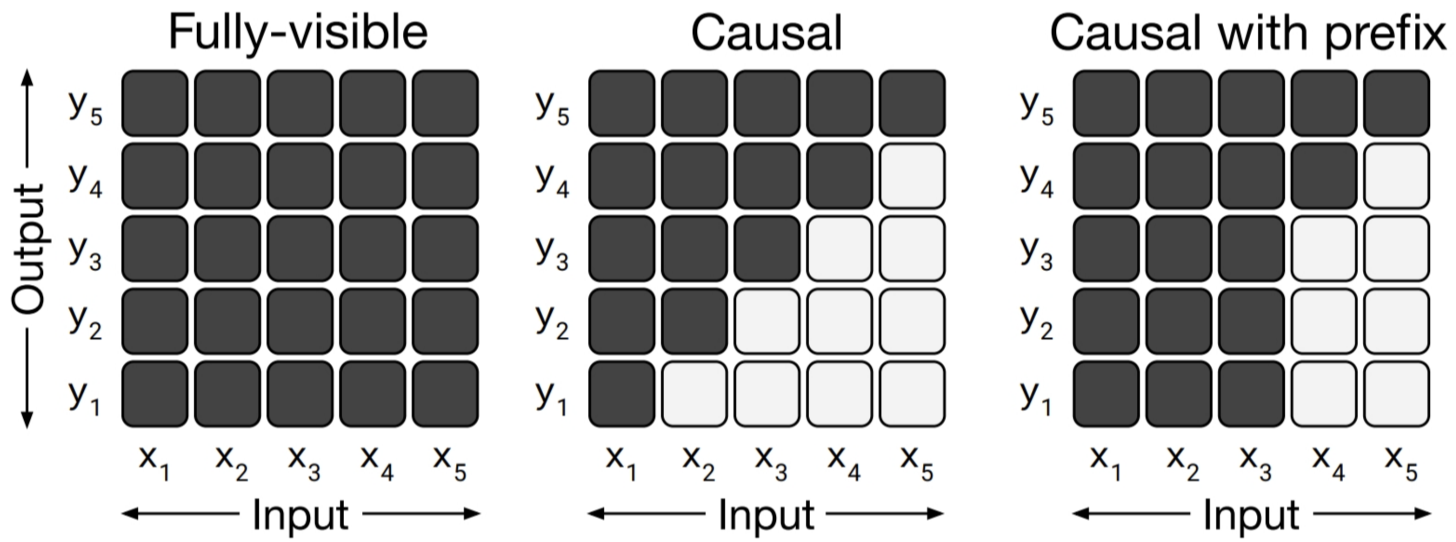
T5 Pretraining

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style	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)*
I.i.d. noise, mask tokens	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>



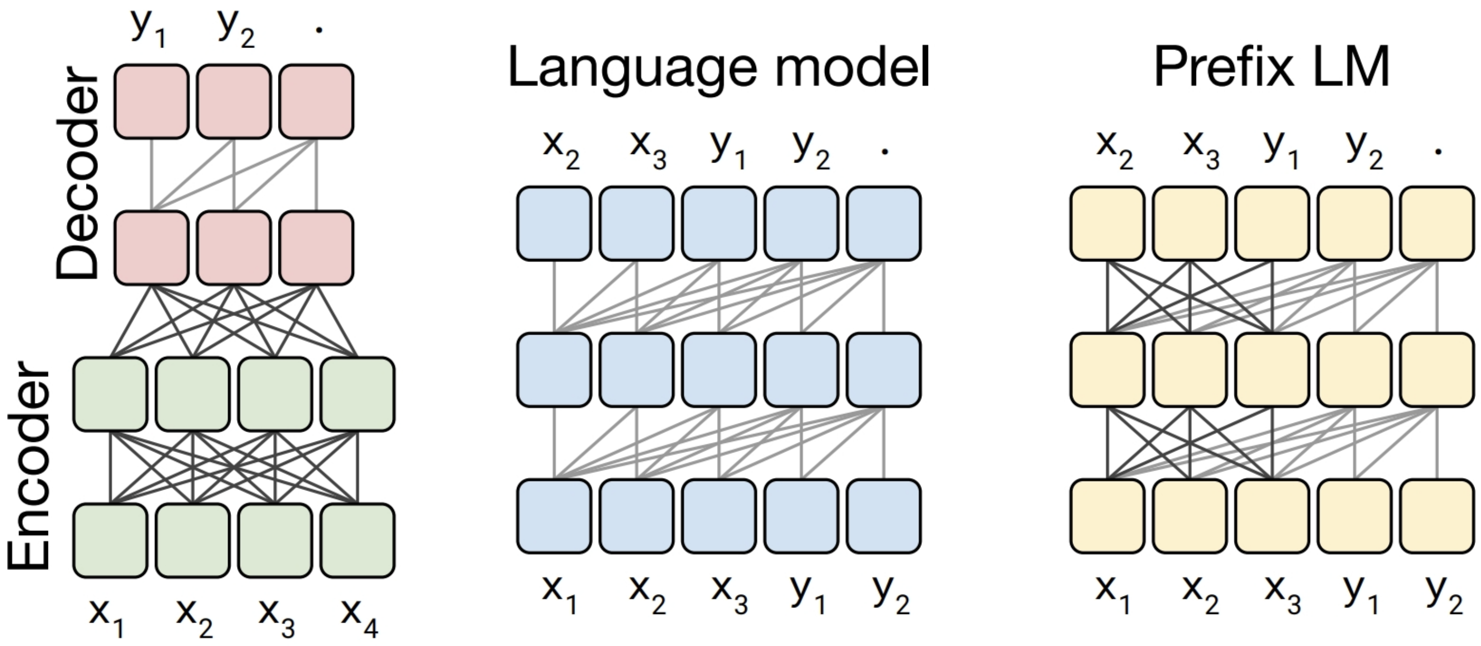
T5 Pretraining

- Different Attention Mask Patterns



3: Matrices representing different attention mask patterns. The input and output of the self-attention mechanism are denoted x and y respectively. A dark cell at row i and column j indicates that the self-attention mechanism is allowed to attend to input element j at output timestep i. A light cell indicates that the

T5 Pretraining



Performance of Different Variants

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

1. Sharing parameters in encoder and decoder models perform nearly as well as the baseline.
2. Halving the number of layers in encoder and decoder hurts the performance.
3. Performance of Encoder and Decoder with shared parameters is better than decoder only LM and prefix LM.

Further Reading

- The Transformer Family Version 2.0
- 乘风破浪的PTM：两年来预训练模型的技术进展
- 通向AGI之路：大型语言模型（LLM）技术精要

Thank you