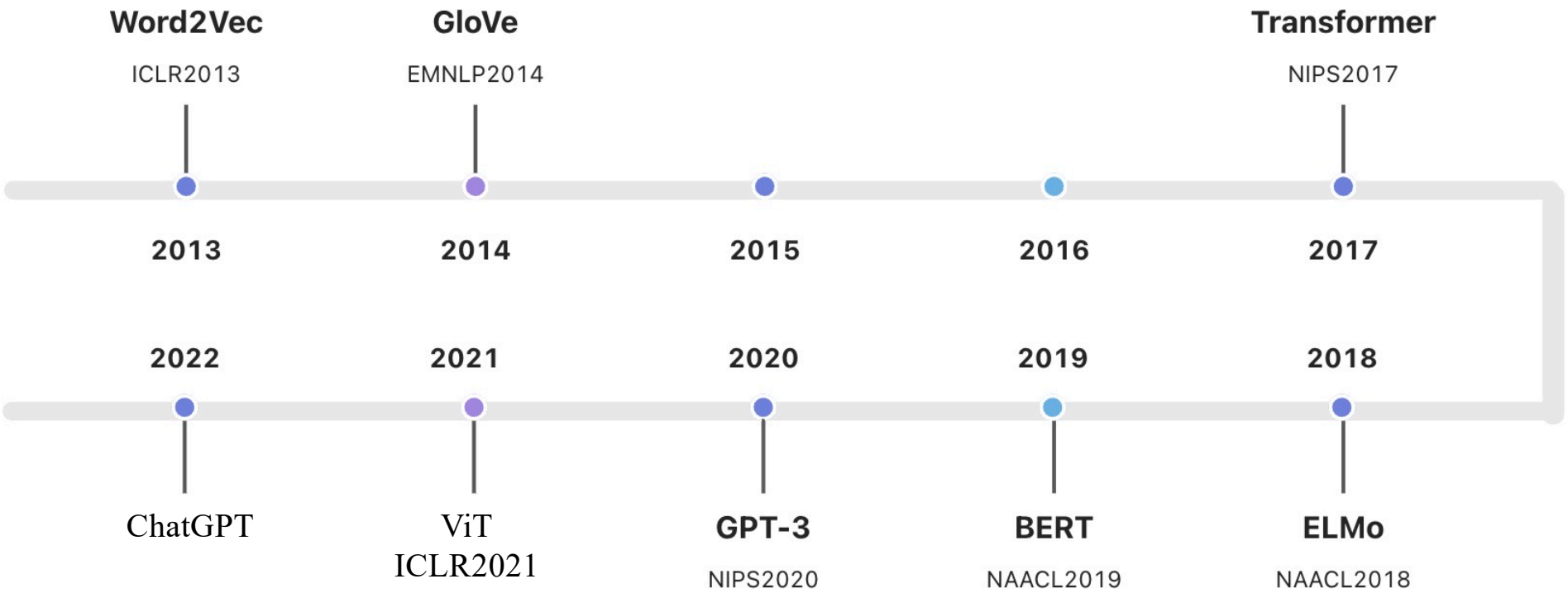
预训练语言模型（下）
基础模型与训练优化

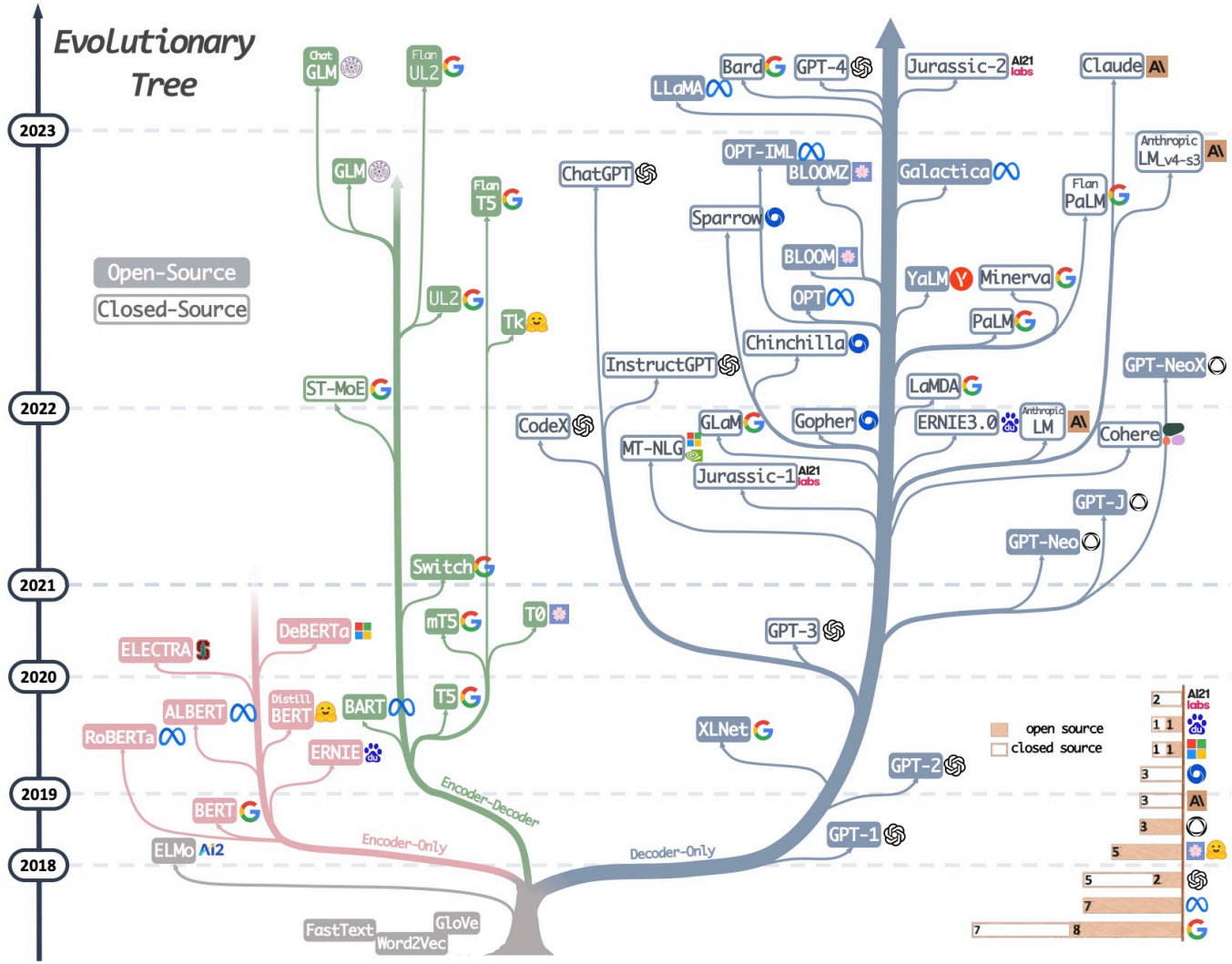
王嘉宁

深度学习课程

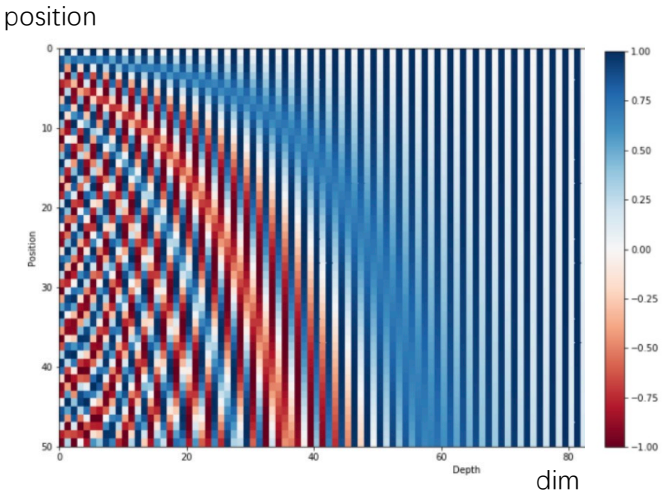
预训练语言模型发展史


Word2Vec
ICLR2013
GloVe
EMNLP2014
Transformer
NIPS2017
2013
2014
2015
2016
2017
2022
2021
2020
2019
2018
ChatGPT
ViT
ICLR2021
GPT-3
NIPS2020
BERT
NAACL2019
ELMo
NAACL2018

预训练语言模型发展史

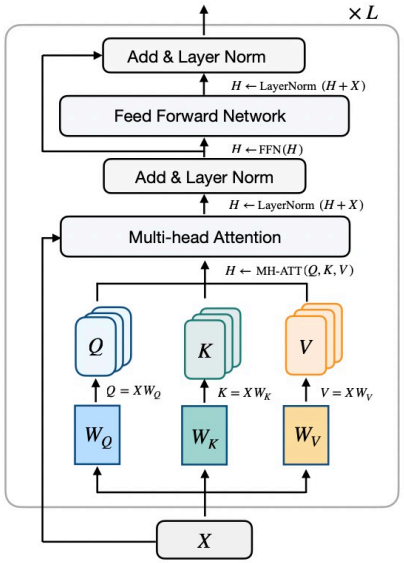

Evolutionary Tree
2023
2022
2021
2020
2019
2018
Open-Source
Closed-Source
Chat GLM
Flan UL2
GLM
Flan T5
UL2
Tk
ST-MoE
Switch
mT5
T0
DeBERTa
ELECTRA
ALBERT
Distill BERT
BART
T5
RoBERTa
ERNIE
BERT
ELMo AI2
FastText
GloVe
Word2Vec
Encoder-Only
Encoder-Decoder
Decoder-Only
LLaMA
Bard
GPT-4
Jurassic-2 AI21 labs
Claude
ChatGPT
OPT-IML
BLOOMZ
Galactica
Anthropic LM_v4-s3
Sparrow
Flan PaLM
BLOOM
YaLM
Minerva
OPT
PaLM
Chinchilla
InstructGPT
LaMDA
GPT-NeoX
CodeX
GLaM
Gopher
ERNIE3.0
Anthropic LM
Cohere
MT-NLG
Jurassic-1 AI21 labs
GPT-J
GPT-Neo
GPT-3
XLNet
GPT-2
GPT-1
open source
closed source
2
1 1
1 1
3
3
3
5
5 2
7
7 8

预训练语言模型基础要素



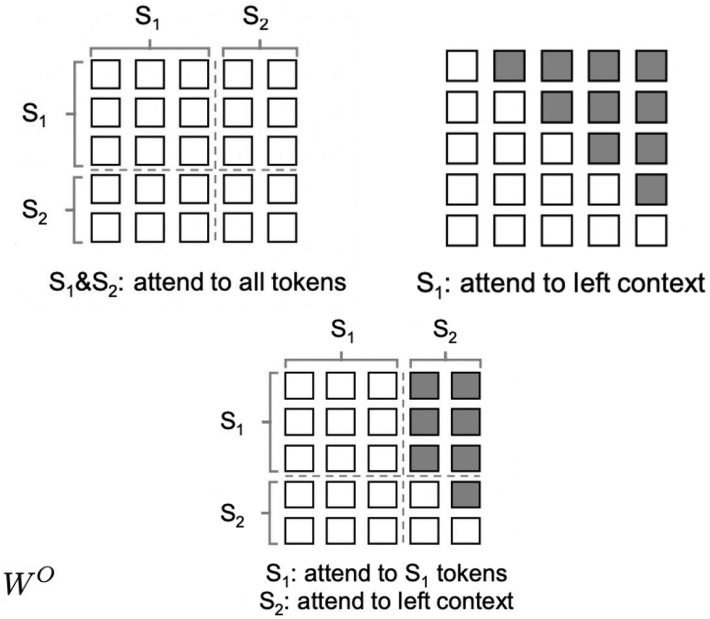
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Transformer分词与位置表征



$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Transformer基础结构

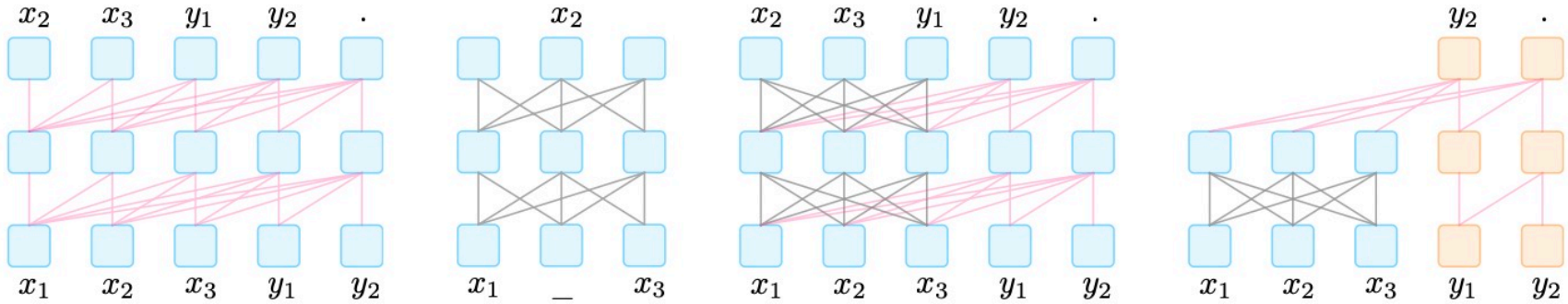


$$\text{Att}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \texttt{Softmax}(\frac{\mathbf{QK}^\mathsf{T}}{\sqrt{d}} + \mathbf{M})\mathbf{V},$$

Attention Mask设计

预训练语言模型的范式：Pre-training + Fine-tuning

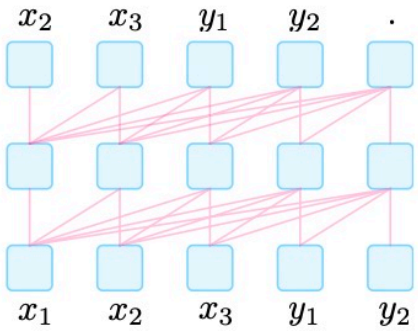
预训练语言模型主要类型



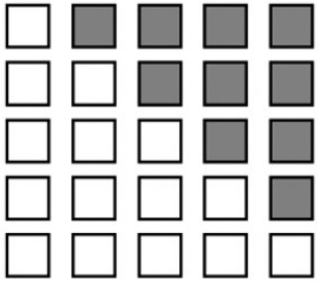
(a) Left-to-right LM. (b) Masked LM. (c) Prefix LM. (d) Encoder-Decoder.

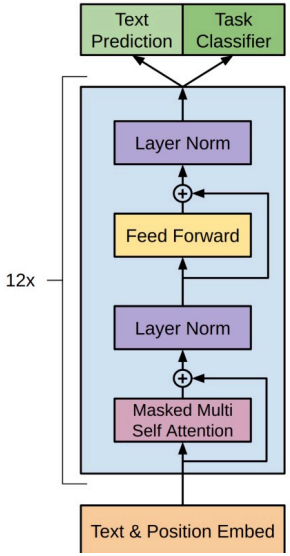
单向（因果）语言模型	双向（掩码）语言模型	混合自回归	端到端语言模型
代表： **ELMO、SiATL GPT、GPT-2、GPT-3、GPT-4、GPT-J、OPT、ChatGPT、LLaMA、Alpaca**	代表： **BERT、MT-DNN、ELECTRA、SpanBERT、RoBERTa、ERNIE-THU、ERNIE-Baidu-1.0/2.0/3.0 ALBERT、XLNet、DeBERTa、SpellBERT、MacBERT、CoLAKE、StructBERT、K-BERT、KEPLER、DKPLM、LayoutLM-V1/2/3、Longformer、Roformer**	代表： **UniLM、GLM、M6、BiET、VLMo**	代表： **BART、T5、MASS**

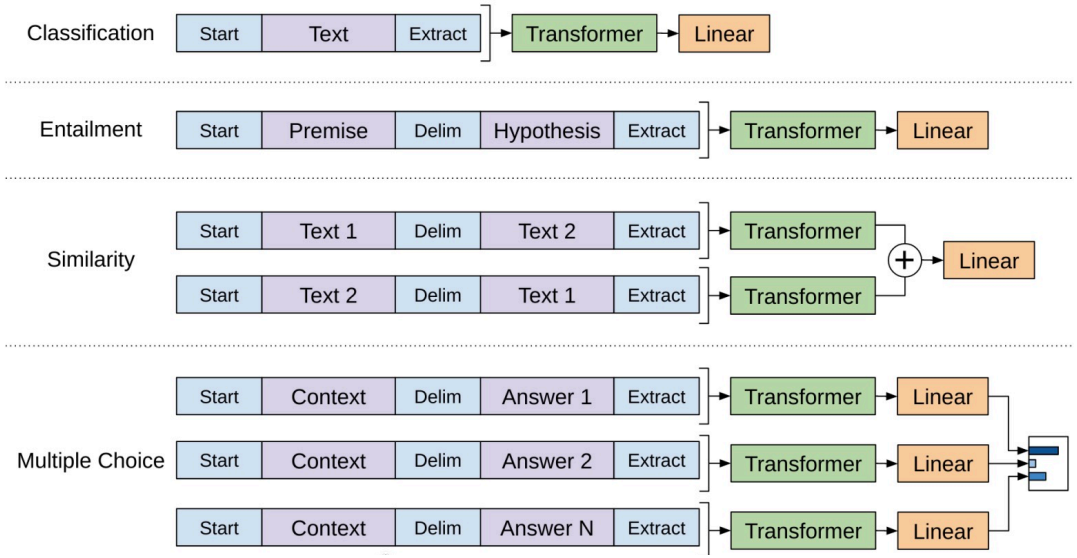
单向语言模型——模型架构



(a) Left-to-right LM.







GPT模型

单向语言模型特点：

- 只使用Transformer Decoder作为模型架构；
- Attention矩阵为Masked Attention；
- 具备很强的生成和推理能力；
- 广泛应用于对话系统、文本问答等；

单向语言模型——语料与训练样例

预训练语料

GPT-2：Reddit上的WebText，数据集共有约800万篇文章，累计体积约40G。

GPT-3：共训练了5个不同的语料，分别是低质量的Common Crawl，高质量的WebText2，Books1，Books2和Wikipedia，GPT-3根据数据集的不同的质量赋予了不同的权值，权值越高的在训练的时候越容易抽样到。

预训练样例

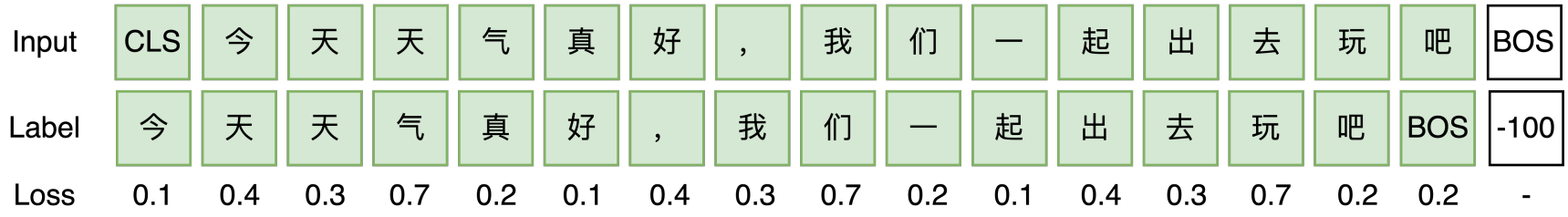
Input: Google News releases that Apple founder Steve Jobs will speak about the new iPhone 4 product

Generation: Google News releases that Apple founder Steve Jobs will speak about the new iPhone 4 product at a press conference in 2014.

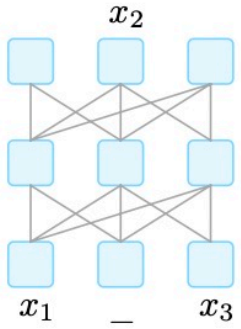
$$p(y_{i+1}, \cdots, y_n | X = \{x_1, x_2, \cdots, x_i\})$$

单向语言模型——Causal Language Modeling预训练

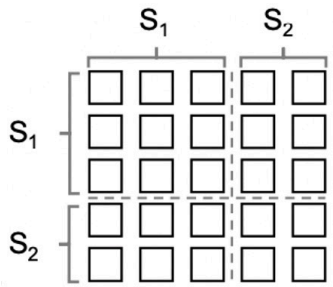
```
transformer_outputs = self.transformer(
     input_ids,
     past_key_values=past_key_values,
     attention_mask=attention_mask,
     token_type_ids=token_type_ids)

hidden_states = transformer_outputs[0]

lm_logits = self.lm_head(hidden_states)

loss = None
if labels is not None:
     # Shift so that tokens < n predict n
     shift_logits = lm_logits[..., :-1, :].contiguous()
     shift_labels = labels[..., 1:].contiguous()
     # Flatten the tokens
     loss_fct = CrossEntropyLoss()
     loss = loss_fct(shift_logits.view(-1, shift_logits.size(-1)), shift_labels.view(-1))

if not return_dict:
          output = (lm_logits,) + transformer_outputs[1:]
return ((loss,) + output) if loss is not None else output
```



双向语言模型——模型架构

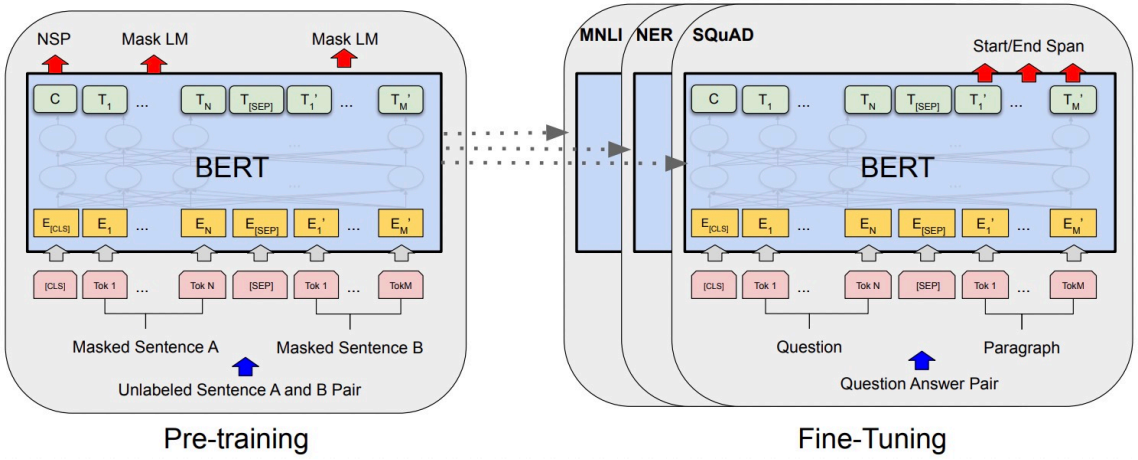


(b) Masked LM.

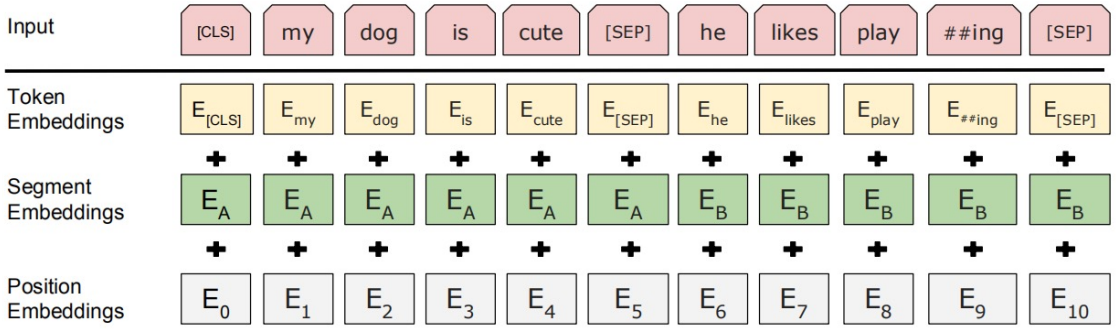


双向语言模型特点：

- 只使用Transformer Encoder作为模型架构；
- Attention矩阵中不存在Masking；
- 具备很强的语义理解能力；
- 广泛应用于文本分类、文本释义、信息抽取、阅读理解等；



BERT模型



BERT Embedding

双向语言模型——语料与训练样例

预训练语料

英文领域：
Wikipedia、BookCorpus、CC-News、OpenWebText、Stories

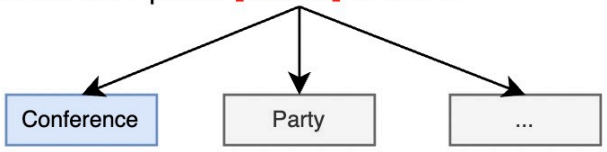
- 中文领域：

百度百科、维基百科（中文）、千言、wudao2.0

预训练样例：Masked Language Modeling（MLM）

Input: Google News releases that Apple founder Steve Jobs will speak about the new iPhone 4 product at a press conference in 2014.

MLM: Google News [MASK] that Apple [MASK] Steve Jobs will speak about the new iPhone 4 product at a press [MASK] in 2014.



$$p(y_j|X = \{x_1, x_2, \cdots, x_n\}, x_j = [\text{MASK}])$$

语料构建：

- 对于每个文本，随机挑选15%的token；
- 对于被选中的token，其中：
  - 80%的token改为[MASK]；
  - 10%的token随机替换词表中其他token；
  - 10%的token保持不变；

拓展：

- 默认情况下，语料中有13.5%的token被训练。
- 实践中，被选中的token数量也可以更多，例如40%。

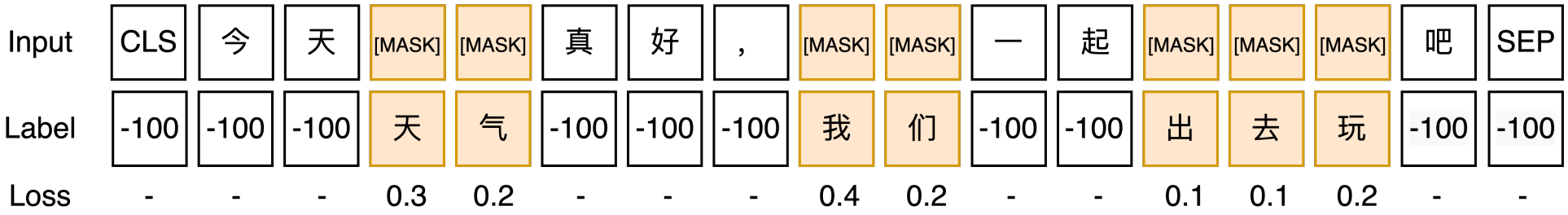
双向语言模型——Masked Language Modeling预训练

```
transformer_outputs = self.transformer(
    input_ids,
    attention_mask=attention_mask,
    token_type_ids=token_type_ids,
    position_ids=position_ids)

sequence_output = transformer_outputs[0]

prediction_scores = self.cls(sequence_output)

masked_lm_loss = None
if labels is not None:
    loss_fct = CrossEntropyLoss() # -100 index = padding token
    masked_lm_loss = loss_fct(prediction_scores.view(-1, self.config.vocab_size), labels.view(-1))

if not return_dict:
        output = (prediction_scores,) + outputs[2:]
return ((masked_lm_loss,) + output) if masked_lm_loss is not None else output
```

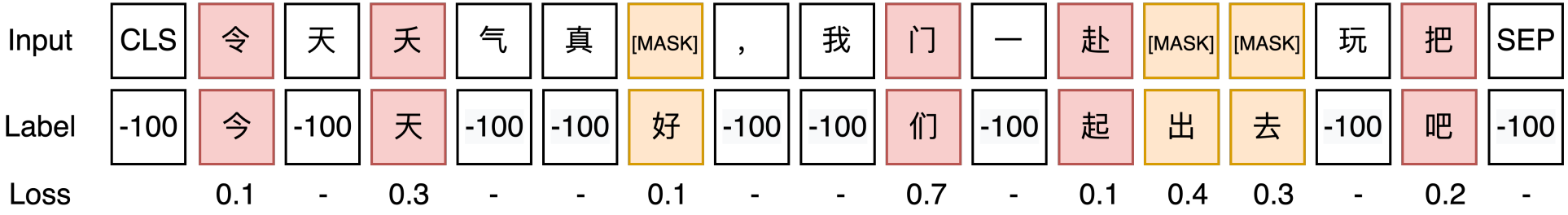
Input	CLS	今	[MASK]	天	[MASK]	真	好	,	我	[MASK]	一	起	出	去	[MASK]	吧	SEP
Label	-100	-100	天	-100	气	-100	-100	-100	-100	们	-100	-100	-100	-100	玩	-100	-100
Loss	-	-	0.3	-	0.2	-	-	-	-	0.2	-	-	-	-	0.2	-	-

双向语言模型——其他典型的预训练

Whole Word Masking MLM

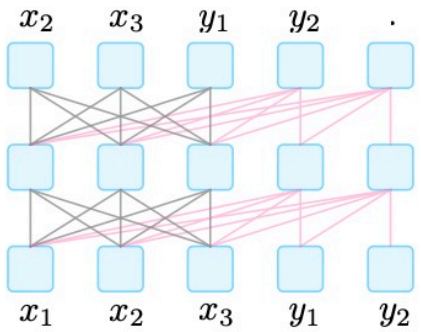


Spell Correction



预训练任务本质是一种自监督训练、自我寻找、构建标签实现在大量语料中学习和理解自然语义知识

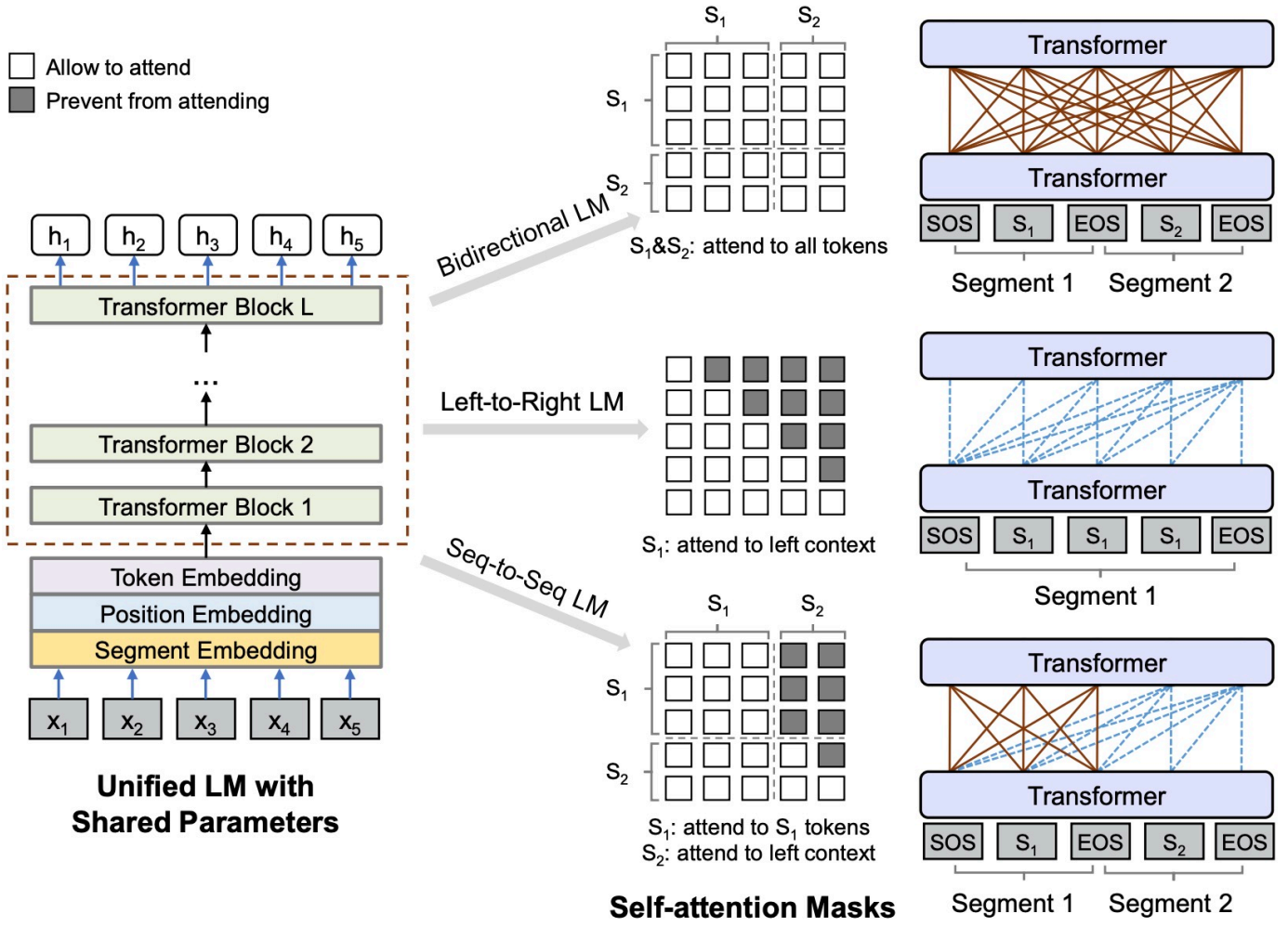
混合语言模型——模型架构



(c) Prefix LM.

双向语言模型特点：

- 只使用Transformer Encoder作为模型架构；
- 根据Attention矩阵的Masking设计，划分为双向、单向和单双混合三种情况；
- 既具备语言理解能力，又拥有文本生成能力；
- 广泛应用于文本摘要、机器翻译等；

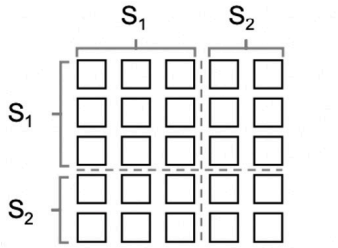


UniLM模型

混合语言模型——预训练

仅双向Mask预训练

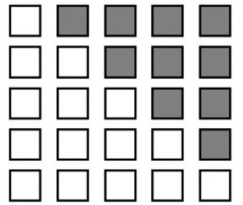
Input	SOS	今	[MASK]	天	[MASK]	真	好	,	我	[MASK]	一	起	出	去	[MASK]	吧	BOS
Label	-100	-100	天	-100	气	-100	-100	-100	-100	们	-100	-100	-100	-100	玩	-100	-100
Loss	-	-	0.3	-	0.2	-	-	-	-	0.2	-	-	-	-	0.2	-	-



S_1&S_2: attend to all tokens

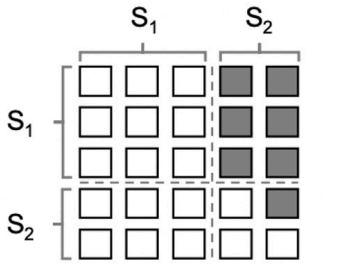
仅单向Causal预训练

Input	SOS	今	天	天	气	真	好	,	我	们	一	起	出	去	玩	吧	BOS
Label	今	天	天	气	真	好	,	我	们	一	起	出	去	玩	吧	BOS	-100
Loss	0.1	0.4	0.3	0.7	0.2	0.1	0.4	0.3	0.7	0.2	0.1	0.4	0.3	0.7	0.2	0.2	-


S_1: attend to left context

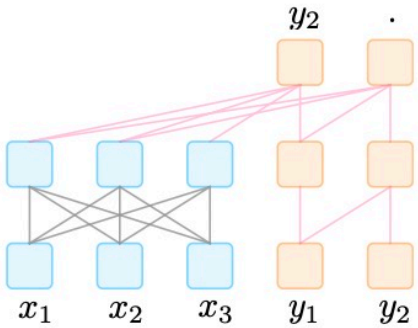
混合Mask & Causal预训练

Input	SOS	今	天	[MASK]	气	真	[MASK]	EOS	我	们	一	起	出	去	玩	吧	BOS
Label	-100	-100	-100	天	-100	-100	好	我	们	一	起	出	去	玩	吧	BOS	-100
Loss	-	-	-	0.3	-	-	0.1	0.4	0.3	0.7	0.2	0.1	0.4	0.3	0.7	0.2	-

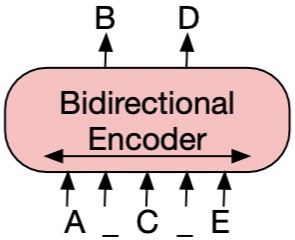


S_1: attend to S_1 tokens
S_2: attend to left context

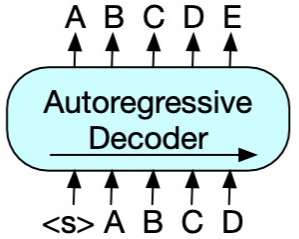
端到端语言模型——模型架构



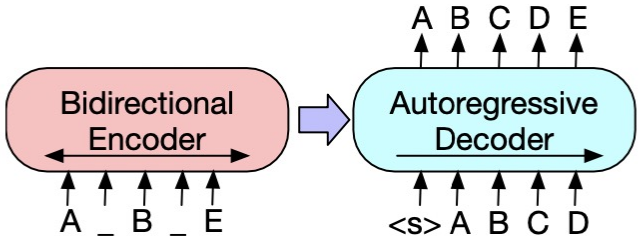
(d) Encoder-Decoder.



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



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



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

端到端语言模型特点：

- 同时使用Transformer Encoder和Decoder作为模型架构；
- Encoder只负责对输入文本进行编码；Decoder则负责自回归生成；
- 既具备语言理解能力，又拥有文本生成能力；
- 广泛应用于文本摘要、机器翻译等；

BART模型、T5模型

端到端语言模型——Conditional Generation预训练

```
# Convert encoder inputs in embeddings if needed
encoder_outputs = self.encoder(
    input_ids=input_ids,
    attention_mask=attention_mask,
)

hidden_states = encoder_outputs[0]

if labels is not None and decoder_input_ids is None
and decoder_inputs_embeds is None:
# get decoder inputs from shifting lm labels to the
right
decoder_input_ids = self._shift_right(labels)

# Decode
decoder_outputs = self.decoder(
    input_ids=decoder_input_ids,
    attention_mask=decoder_attention_mask
```

```
sequence_output = decoder_outputs[0]

lm_logits = self.lm_head(sequence_output)

loss = None
if labels is not None:
    loss_fct = CrossEntropyLoss(ignore_index=-100)
    loss = loss_fct(lm_logits.view(-1,
    lm_logits.size(-1)), labels.view(-1))

if not return_dict:
        output = (lm_logits,) + decoder_outputs[1:] +
encoder_outputs
return ((loss,) + output) if loss is not None else
output
```

Input	SOS	今	天	天	气	真	好	,	我	们	一	起	出	去	玩	吧	BOS
Label	今	天	天	气	真	好	,	我	们	一	起	出	去	玩	吧	BOS	-100
Loss	0.1	0.4	0.3	0.7	0.2	0.1	0.4	0.3	0.7	0.2	0.1	0.4	0.3	0.7	0.2	0.2	-

预训练语言模型的优化——数据优化

超大规模数据

大规模高质量语料

- 英文领域：

Wikipedia、BookCorpus、CC-News、OpenWebText、Stories

- 中文领域：

百度百科、维基百科（中文）、千言、wudao2.0

超大规模参数

模型架构加深加宽

- BERT/RoBERTa-large：770M
- Erlangshen：1.3B
- Nezha：4B
- GPT-3：175B
- M6：1000B
- Switch-Transformer：1600B

超大规模算力

大规模分布式深度学习训练

- 混合精度FP16
- MOE训练
- ZeRO-Offload
- Deepspeed分布式
- Megatron

预训练语言模型的优化——数据优化

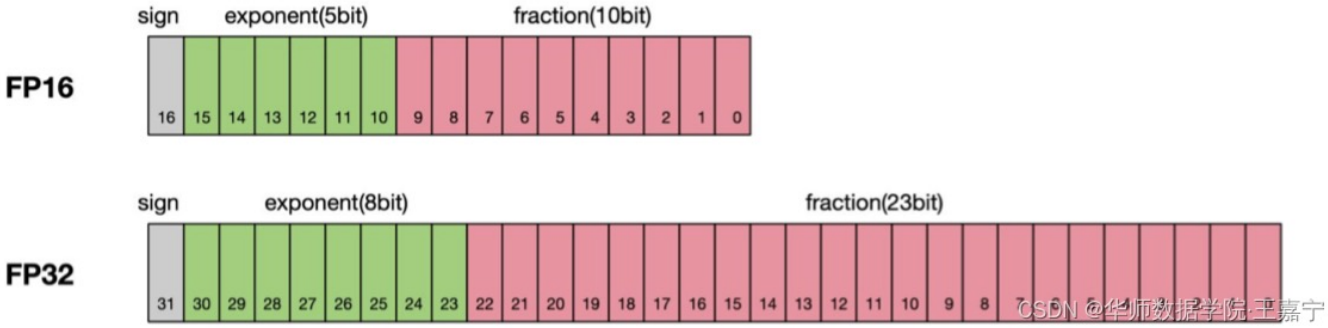
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$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

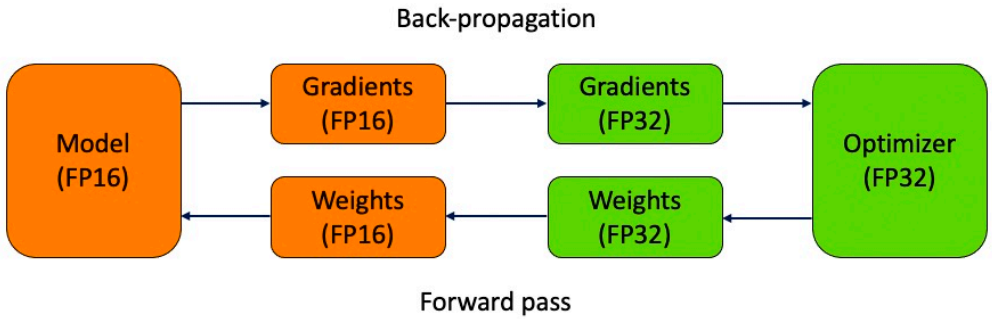
Hyperparam	$\text{RoBERTa}_{\text{LARGE}}$	$\text{RoBERTa}_{\text{BASE}}$
Number of Layers	24	12
Hidden size	1024	768
FFN inner hidden size	4096	3072
Attention heads	16	12
Attention head size	64	64
Dropout	0.1	0.1
Attention Dropout	0.1	0.1
Warmup Steps	30k	24k
Peak Learning Rate	4e-4	6e-4
Batch Size	8k	8k
Weight Decay	0.01	0.01
Max Steps	500k	500k
Learning Rate Decay	Linear	Linear
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Gradient Clipping	0.0	0.0

- 增加数据量、batch size、迭代次数
- 在pre-training预fine-tuning之间增加continual pre-training
- Trick：使用梯度累积来增大batch size
- 缺陷：难以获得高质量语料，对低资源语言不友好

预训练语言模型的优化——算力优化

- 混合精度（Fp16、BF16）
- Deepspeed（3D并行）
- int8量化
- 梯度累积
- Gradient checkpointing
- Torch FSDP
- CPU offloading

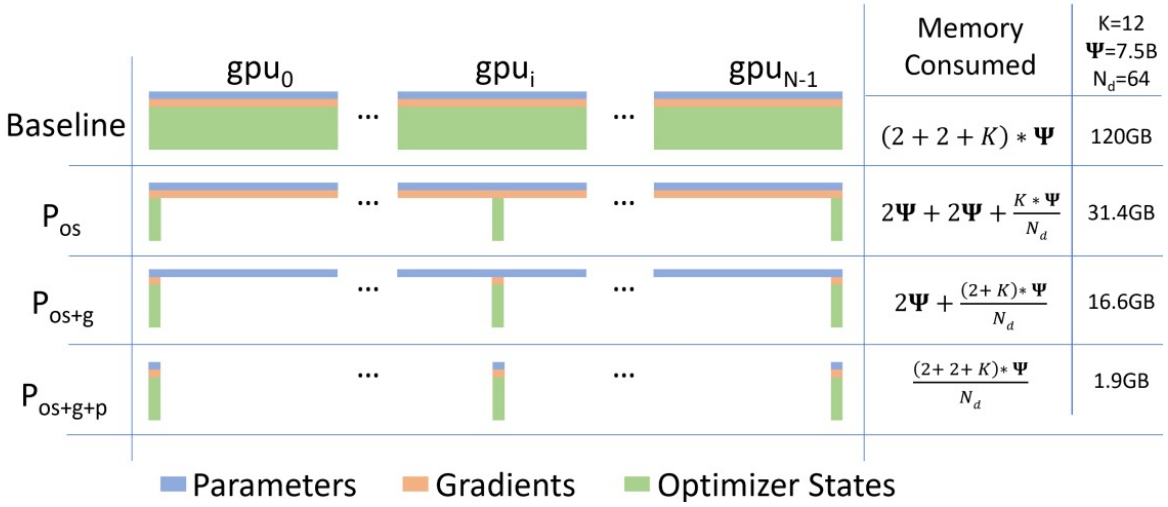


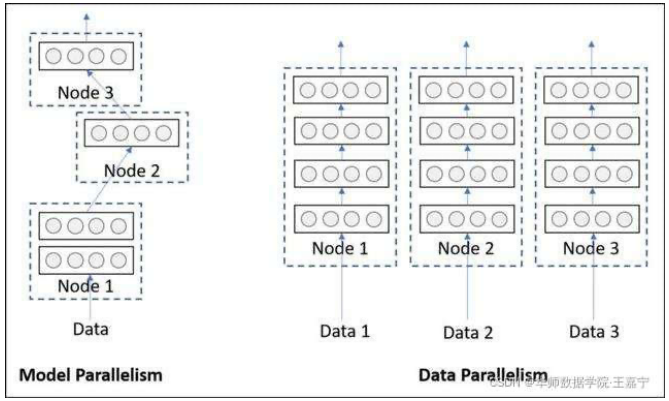


参考资料：https://wjn1996.blog.csdn.net/article/details/130764843

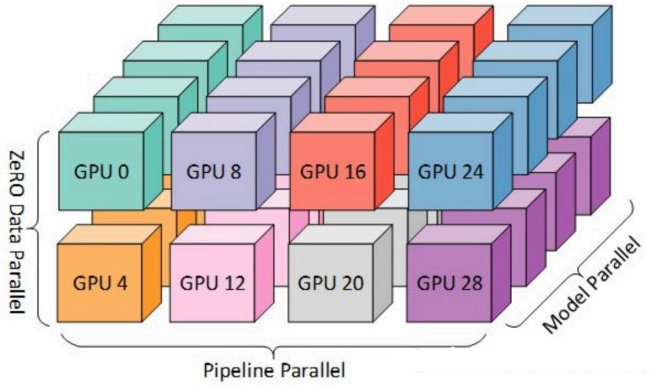
预训练语言模型的优化——算力优化

- 混合精度（Fp16、BF16）
- Deepspeed（3D并行）
- int8量化
- 梯度累积
- Gradient checkpointing
- Torch FSDP
- CPU offloading





模型并行与数据并行

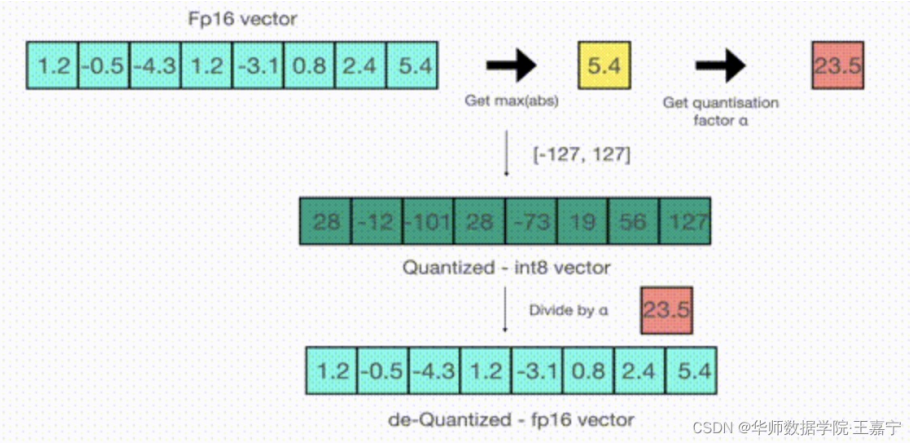


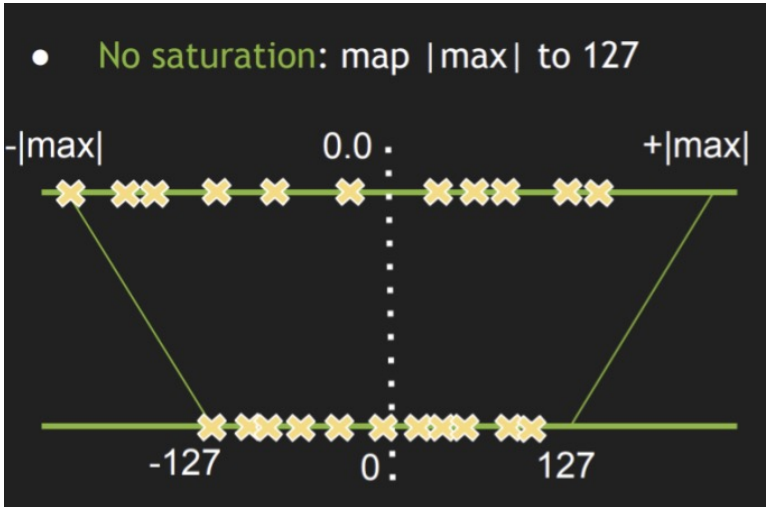
DeepSpeed+ZeRO——3D并行

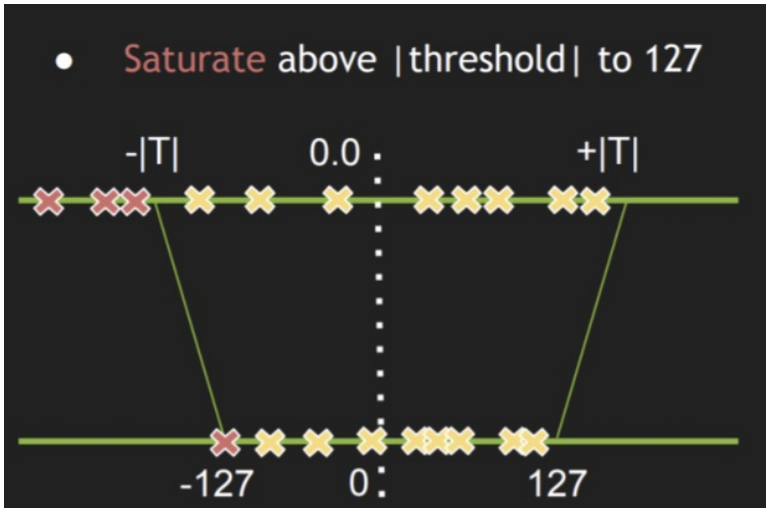
参考资料：https://wjn1996.blog.csdn.net/article/details/130764843

预训练语言模型的优化——算力优化

- 混合精度（Fp16、BF16）
- Deepspeed（3D并行）
- int8量化
- 梯度累积
- Gradient checkpointing
- Torch FSDP
- CPU offloading







参考资料：https://wjn1996.blog.csdn.net/article/details/130764843

预训练语言模型的优化——算力优化

- 混合精度（Fp16、BF16）
- Deepspeed（3D并行）
- int8量化
- 梯度累积
- Gradient checkpointing
- Torch FSDP
- CPU offloading

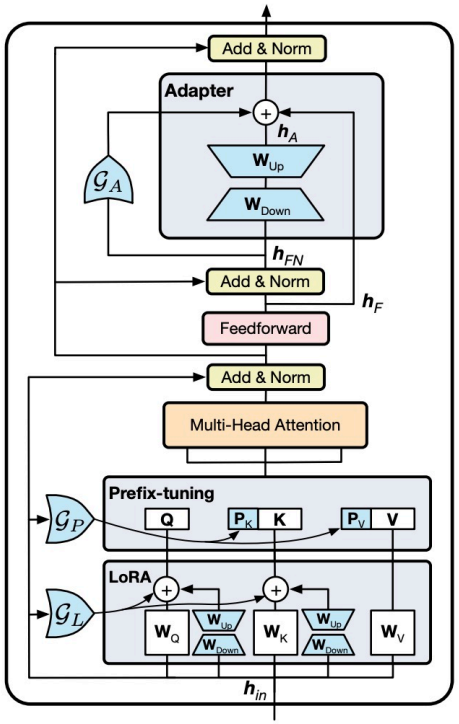
梯度累积：
一轮Batch样本计算梯度后，并不立刻进行参数更新，而是等待多轮Batch的梯度之后再进行参数更新。本质上等价于扩大了Batch size。

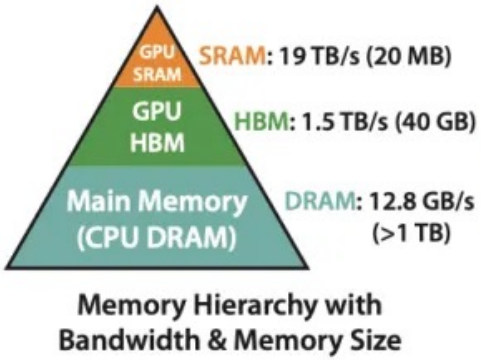
Gradient Checkpointing：
在进行反向传播计算梯度时，并不存储每个张量（算子）的梯度，而是重新计算梯度，消除了显式存储梯度所占用的显存。

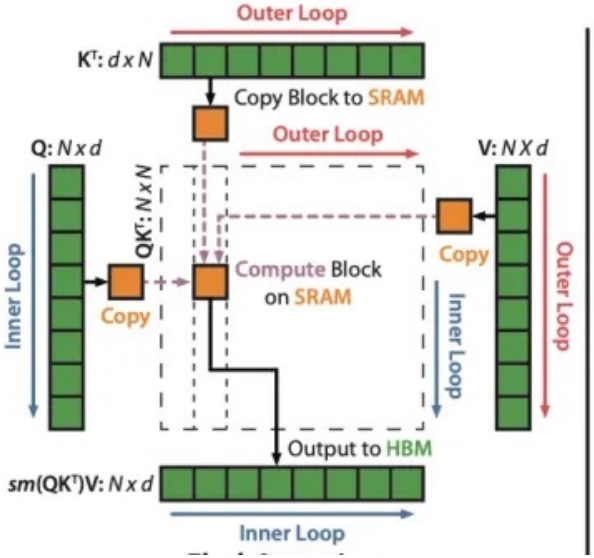
Torch FSDP+CPU Offloading：
Pytorch实现的分布式训练框架，同时借助CPU内存实现存储。需要频繁地执行内存与显存之间的数据通信。

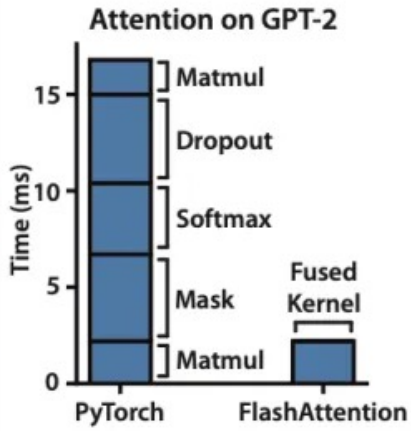
参考资料：https://wjn1996.blog.csdn.net/article/details/130764843

预训练语言模型的优化——模型优化









参数有效性训练

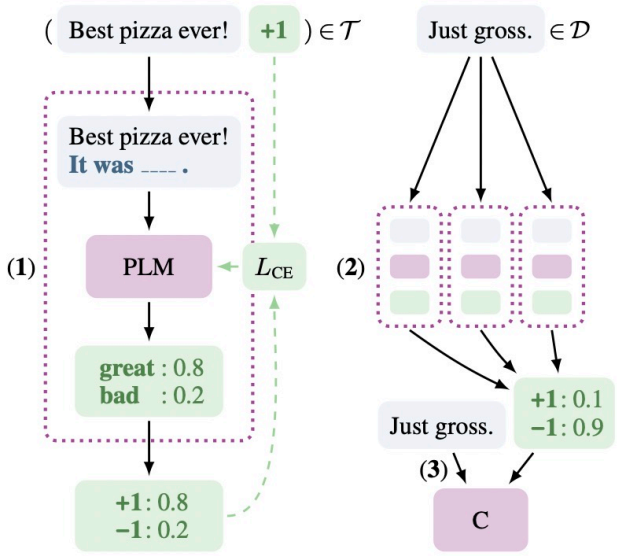
Adapter & Prefix & LoRA

Self-Attention优化：Flash-Attention

参考资料：https://wjn1996.blog.csdn.net/article/details/130764843

预训练语言模型的优化——范式优化

- Prompt-tuning（提示学习）
- In-Context Learning（上下文学习）
- Instruction-tuning（指令微调）
- Chain-of-Thought（思维链）

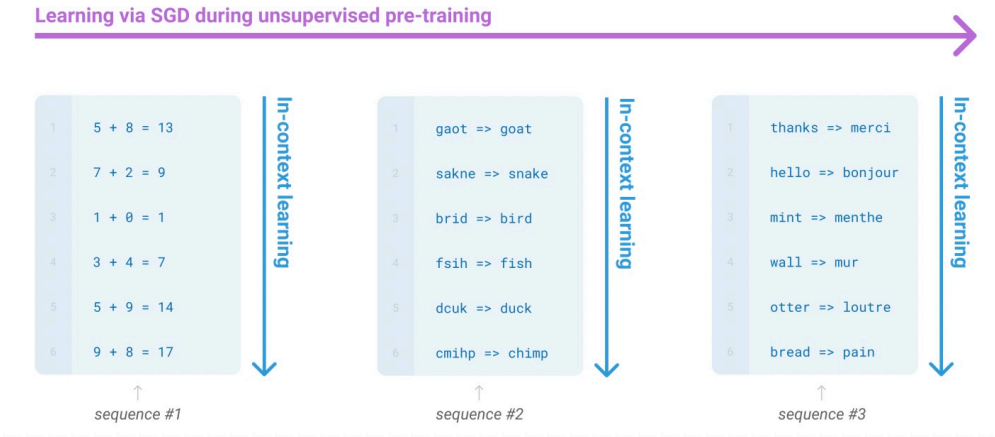


Pattern	Verbalizer
$P_1(a) =$ It was ____. a $P_2(a) =$ Just ____! $\|\|$ a $P_3(a) =$ a. All in all, it was ____. $P_4(a) =$ $a \|\|$ In summary, the restaurant is ____.	$v(1) =$ terrible $v(2) =$ bad $v(3) =$ okay $v(4) =$ good $v(5) =$ great
$P_1(\mathbf{x}) =$ "a"? $\|\|$ ____, "b" $P_2(\mathbf{x}) =$ a? $\|\|$ ____, b	$v_1(0) =$ Wrong $v_1(1) =$ Right $v_1(2) =$ Maybe $v_2(0) =$ No $v_2(1) =$ Yes $v_2(2) =$ Maybe
p. Question: q? Answer: ___. p. Based on the previous passage, q? ___. Based on the following passage, q? ___. p	**Yes / No**

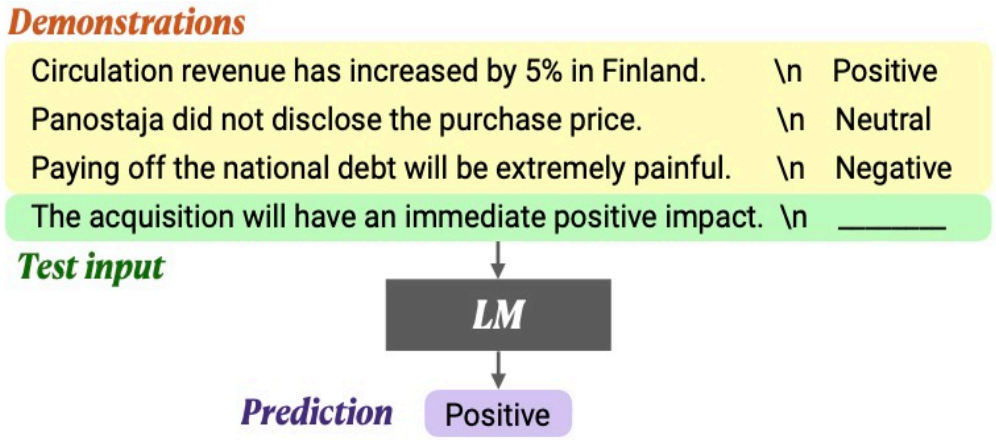
参考资料：https://wjn1996.blog.csdn.net/article/details/120607050

预训练语言模型的优化——范式优化

- Prompt-tuning（提示学习）
- In-Context Learning（上下文学习）
- Instruction-tuning（指令微调）
- Chain-of-Thought（思维链）



GPT-3

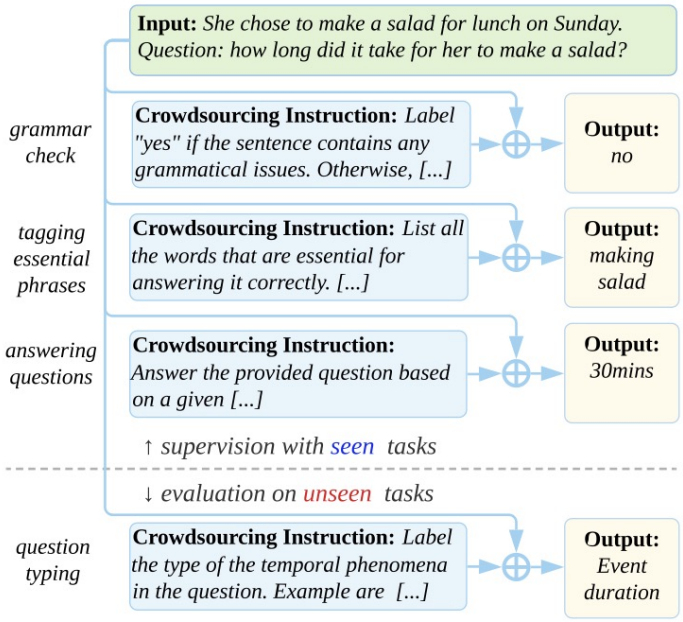


In-Context Learning

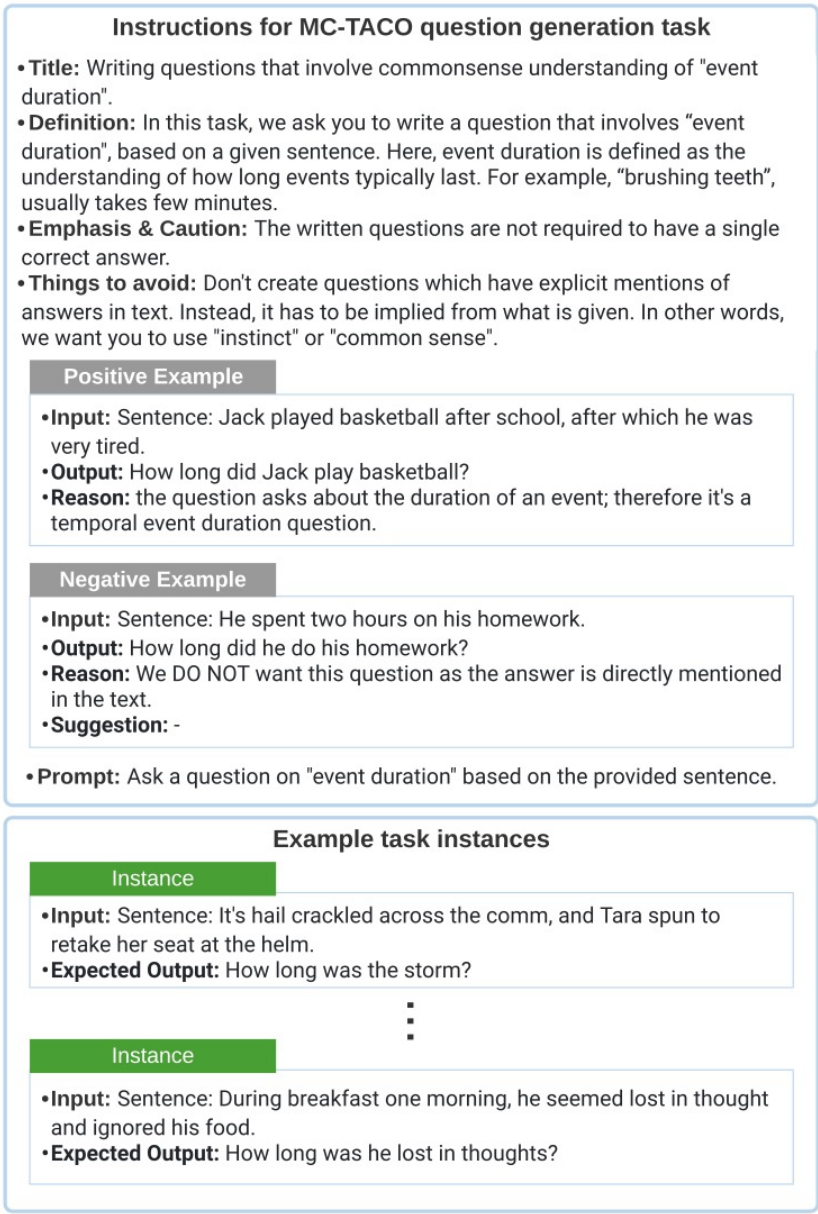
参考资料：https://wjn1996.blog.csdn.net/article/details/120607050

预训练语言模型的优化——范式优化

- Prompt-tuning（提示学习）
- In-Context Learning（上下文学习）
- Instruction-tuning（指令微调）
- Chain-of-Thought（思维链）



ChatGPT（GPT-3.5-turbo）主要使用指令微调+人类反馈强化学习实现的



参考资料：https://wjn1996.blog.csdn.net/article/details/120607050

预训练语言模型的优化——范式优化

- Prompt-tuning（提示学习）
- In-Context Learning（上下文学习）
- Instruction-tuning（指令微调）
- Chain-of-Thought（思维链）

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

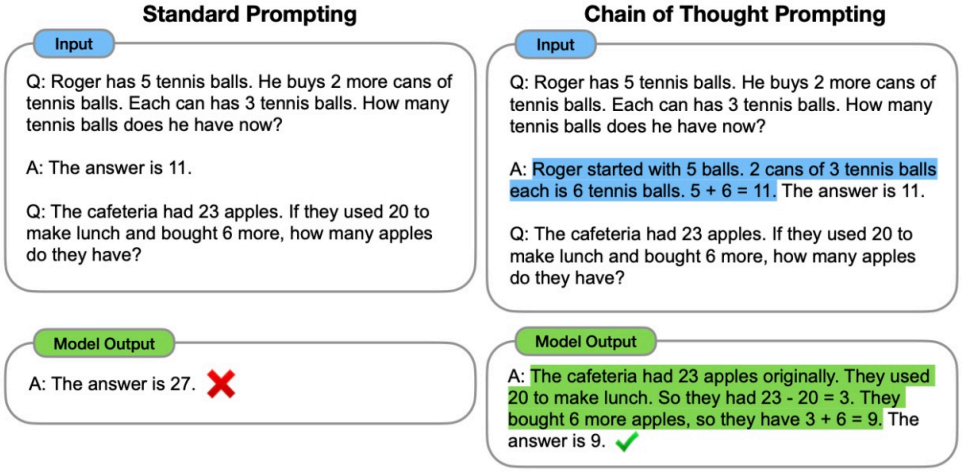
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

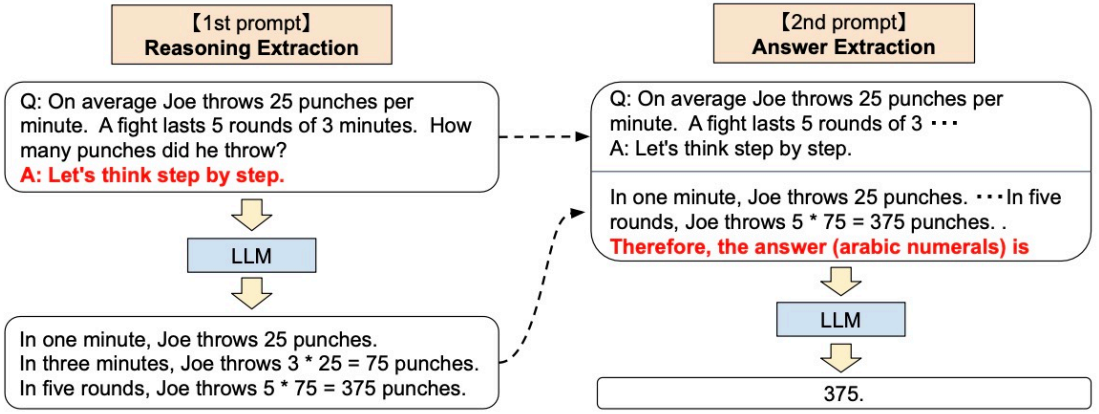
A: There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?

A: Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39. The answer is 39.



人工标注推理路径

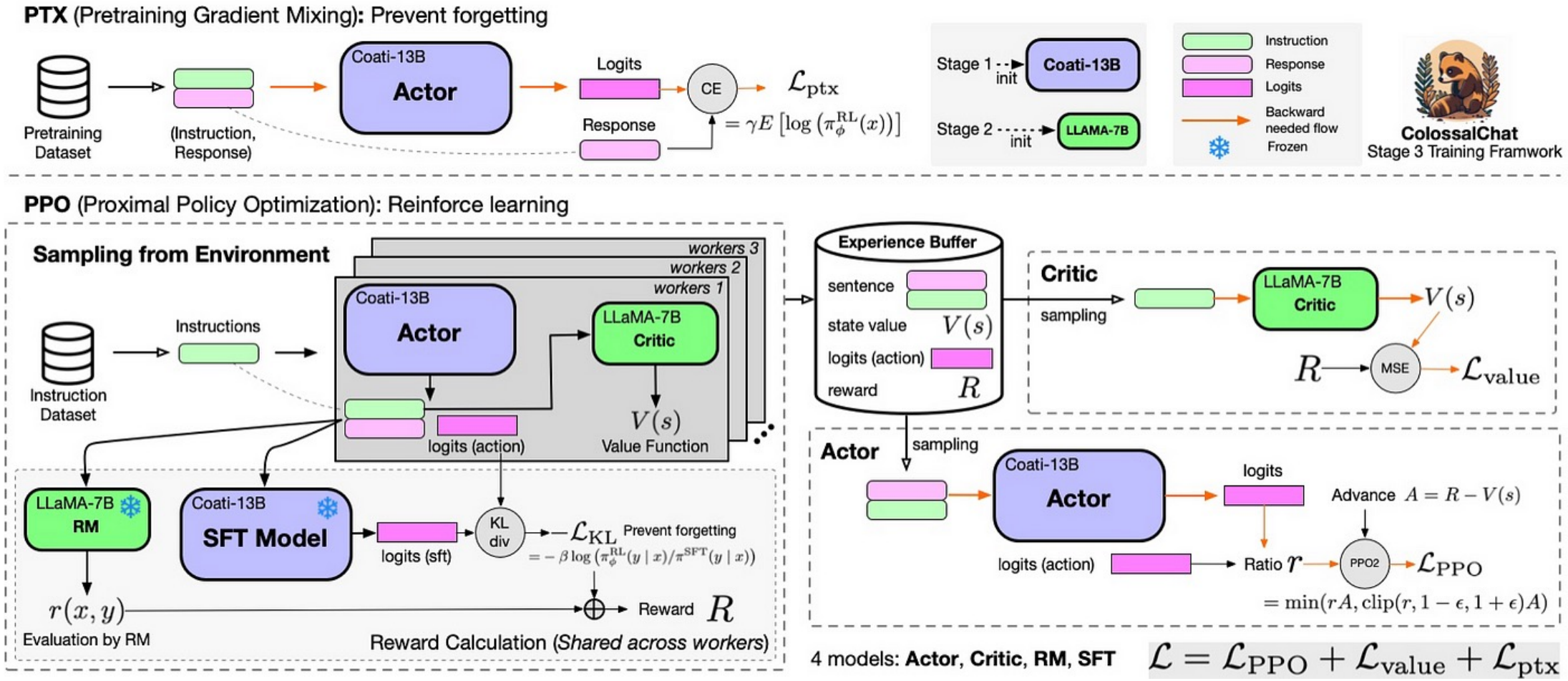


零样本推理路径

参考资料：https://wjn1996.blog.csdn.net/article/details/120607050

ChatGPT简要介绍

第一步：指令微调　　第二步：训练奖励函数　　第三步：人类反馈的强化学习对齐+PPO算法



类ChatGPT热门开源项目

名称	项目地址	基础模型	训练方法/数据集	备注
Alpaca	https://github.com/tatsu-lab/stanford_alpaca	LLaMA	Alpaca	finetune 参考 https://github.com/tloen/alpaca-lora
ChatGLM	https://github.com/THUDM/ChatGLM-6B	GLM	自定义数据集(1T)	finetune 参考 https://github.com/mymusise/ChatGLM-Tuning
Dolly	https://github.com/databrickslabs/dolly	GPT-J 6B	Alpaca	
BELLE	https://github.com/LianjiaTech/BELLE	BLOOM	Alpaca转中文+自定义数据集(0.5 ~ 2M)	
OpenChatKit	https://github.com/togethercomputer/OpenChatKit	GPT-NEOX/Pythia	OIG-43M	第一次测的时候问题比较大就没有跟进，现在更新了可以再看下
FastChat/Vicuna	https://github.com/lm-sys/FastChat	LLaMA	shareGPT(70k)	搞ShareGPT的数据集用来训练用户真的知道吗？使用了GPT-4作质量判断
gpt4all	https://github.com/nomic-ai/gpt4all	LLaMA	自定义数据集(800k)	数据集由GPT-3.5生成
lit-llama	https://github.com/Lightning-AI/lit-llama	Lit-LLaMA	Alpaca	
antropic	https://huggingface.co/datasets/Anthropic/hh-rlhf		无害性数据集（160k）	
stack- exchange-paired	https://huggingface.co/datasets/lvwerra/stack-exchange-paired		用于训练偏好模型	
InstructionWild	https://github.com/XueFuzhao/InstructionWild		高质量的user instruct prompt	
HugNLP	https://github.com/HugAILab/HugNLP	GPT-2、OPT等	搜集的大量指令微调数据	

课后思考

- 预训练语言模型有哪些？它们的预训练任务是什么？
- 不同类型的预训练语言模型的Transformer结构有什么不同？
- 预训练语言模型如何进行Fine-tuning？
- XLNet、RoBERTa和ALBERT相比BERT改进了什么？
- 你知道有哪些预训练模型的优化技术？
- 什么是Prompt？有哪些Prompt技术？
- 你知道现有的类ChatGPT模型还有哪些？
- 为什么现在热门的大模型ChatGPT是Decoder-only模型？
- 走向通用人工智能的决定因素有哪些？