通用大模型原理及训练实践
第一章：大语言模型

冯洋

中国科学院计算技术研究所
Institute of Computing Technology,Chinese Academy of Sciences

本章大纲

- 大语言模型发展历程
- 大语言模型原理介绍
  - Transformer
  - Encoder-only架构大模型
  - Decoder-only架构大模型
  - Encoder-Decoder架构大模型
- 大语言模型使用方式

大语言模型出现原因

- 自然语言处理任务
  - 对话系统
  - 机器翻译
  - 问答
  - 生成
  - 摘要
  - 阅读理解
  - ……

大语言模型出现原因

- 每个任务的解决方案涉及
  - 模型架构
    - 符号主义
    - 连接主义
      - 统计模型
      - 神经网络
  - 学习方法
    - 监督学习：输入数据全部带有标记
    - 半监督学习：少量有标记，大量无标记
    - 自监督学习：输入数据无标记，模型自己生成标记，如语言模型训练
    - 无监督学习：输入数据无标记

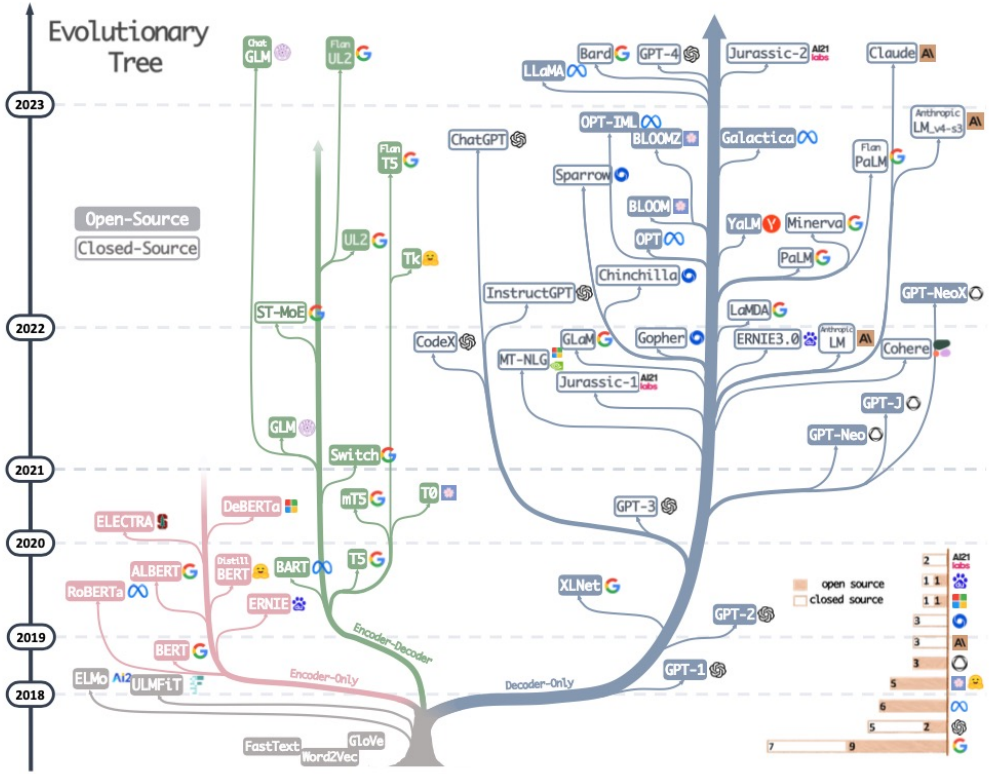
大语言模型出现原因

- 原有范式为神经网络+监督学习
- 存在问题：
  - 特定任务使用
  - 需要大量标注数据
  - 可迁移性弱
- 改进目标
  - 任务间共用
  - 无监督学习
  - 可迁移性强

大语言模型出现原因

- 改进方案：预训练-微调
  - 上游：预训练语言模型
    - 预测 next token
  - 下游：针对特定任务微调
    - 全量参数微调
    - 部分参数微调或者添加参数微调
    - 参数不变，prompt或者in-context learning
  - 与人类对齐：带人类反馈的强化学习

大语言模型进化历程



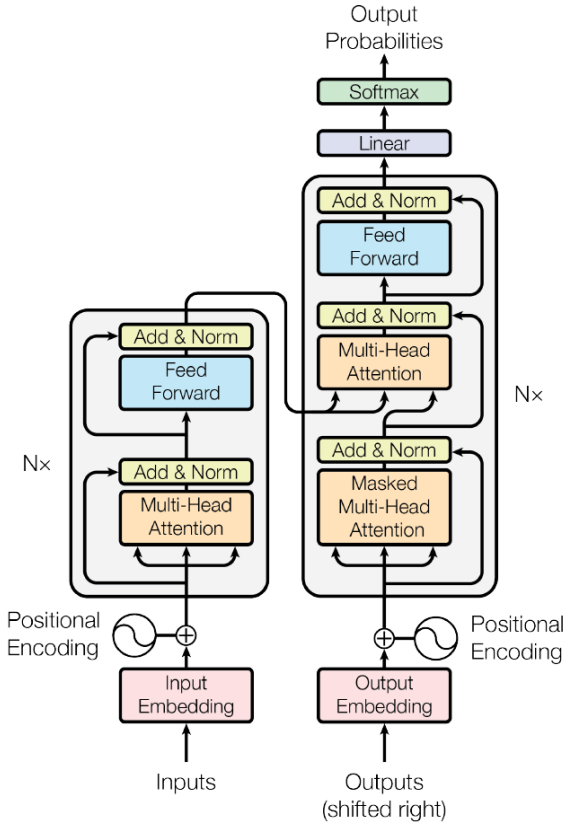
图片来源：Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond

本章大纲

- 大语言模型发展历程
- 大语言模型原理介绍
  - Transformer
  - Encoder-only架构大模型
  - Decoder-only架构大模型
  - Encoder-Decoder架构大模型
- 大语言模型使用方式

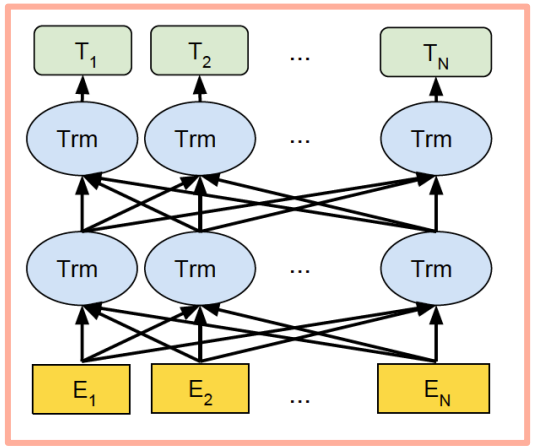
Transformer架构

- 编码器-解码器架构
  - 编码器
    - 6层，每层包括Multi-Head Self-Attention和Feed Forward
  - 解码器
    - 6层，每层包括Masked Multi-Head Self-Attention、Multi-Head Cross-Attention和Feed Forward



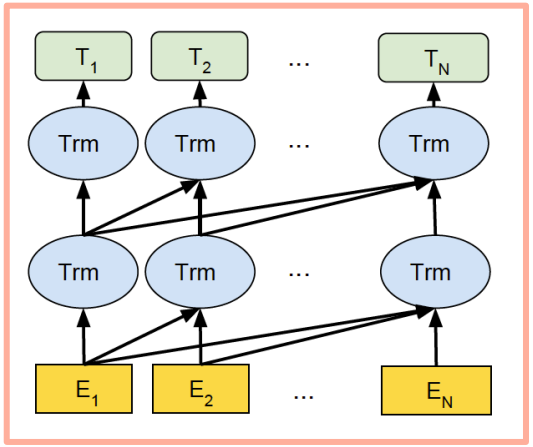
Attention is all you need

Encoder-Decoder架构



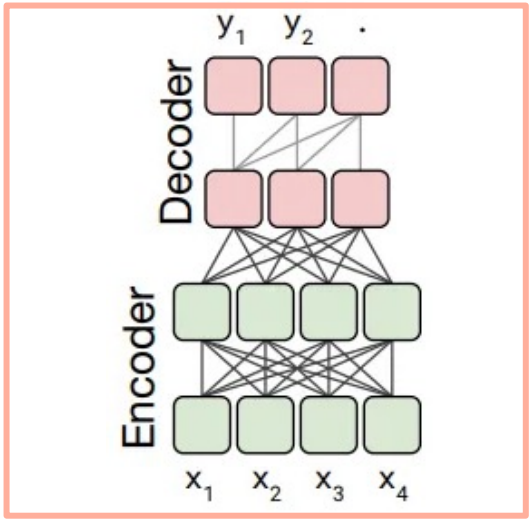
Encoder：

Bidirectional Self-Attention



Decoder:

Causal Attention



Encoder-Decoder

本章大纲

- 大语言模型发展历程
- 大语言模型原理介绍
  - Transformer
  - Encoder-only架构大模型
  - Decoder-only架构大模型
  - Encoder-Decoder架构大模型
- 大语言模型使用方式

BERT结构

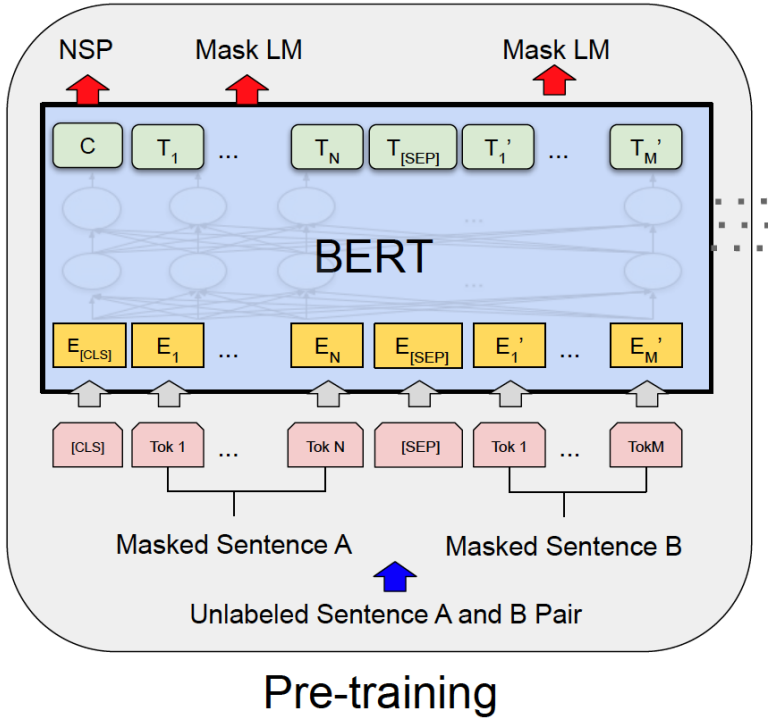
- 预训练包含两个任务
  - Next Sentence Prediction
  - Masked LM

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

- 训练损失

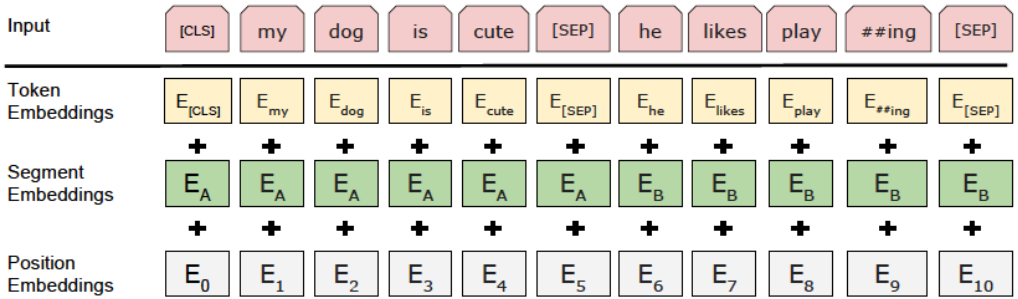
$$L = L_{nsp} + L_{mlm}$$



Pre-training

BERT结构

- 输入表示



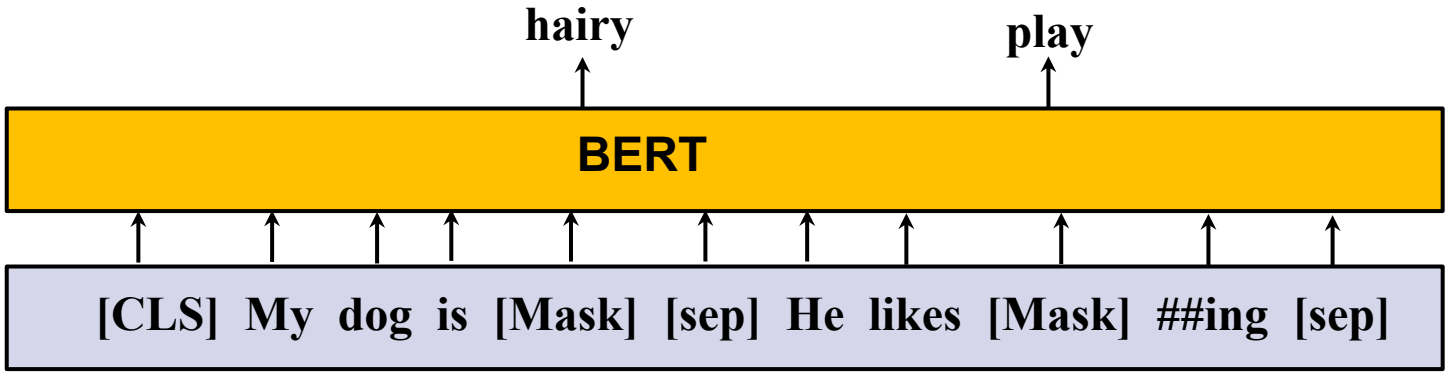
- Next Sentence Prediction
  - 判断两个片段是否连续，不一定是句子
  - 50%随机片段，50%连续片段来组成句对
  - [CLS]学到的为句子表示

BERT结构

- Masked LM
  - 随机取15%的token进行处理

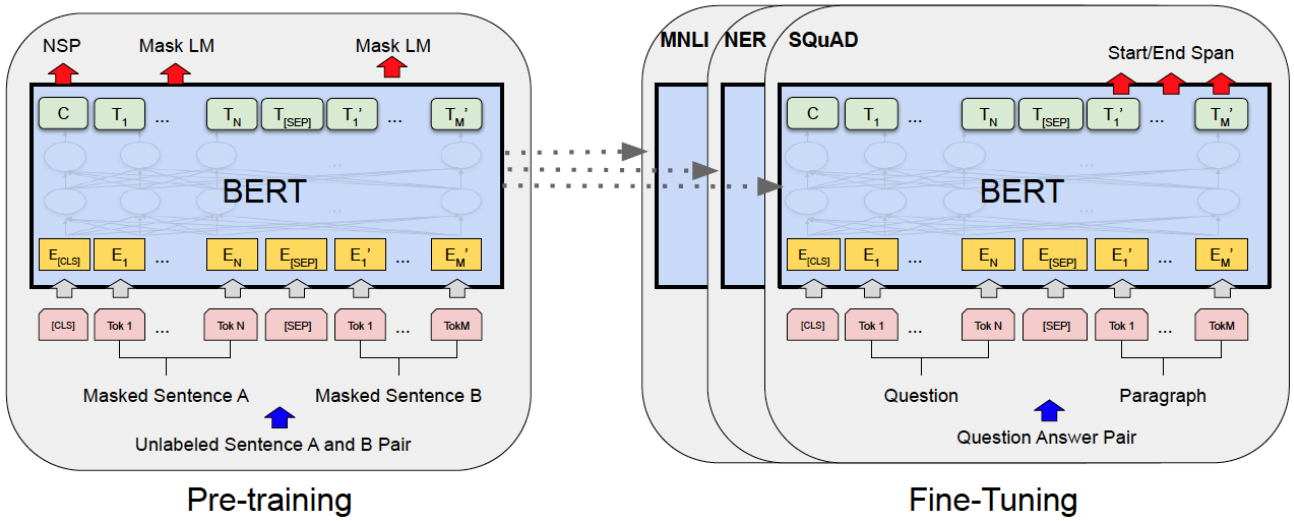
原句：my dog is hairy

- 80%被替换为[Mask] — my dog is [mask]
- 10%被替换为随机token — my dog is apple
- 10%保持原样 — my dog is hairy

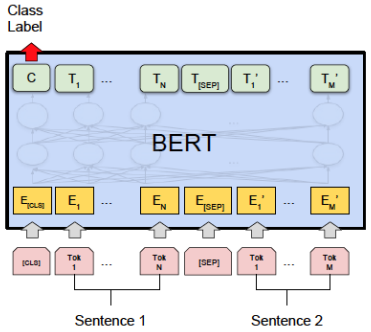


BERT使用方式

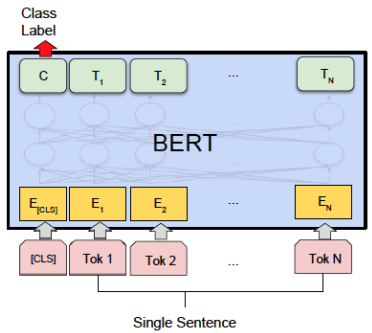
- 上游预训练，下游在特定任务上微调



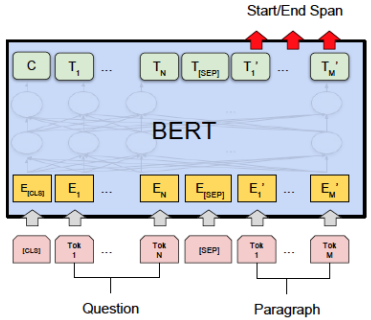
BERT使用方式



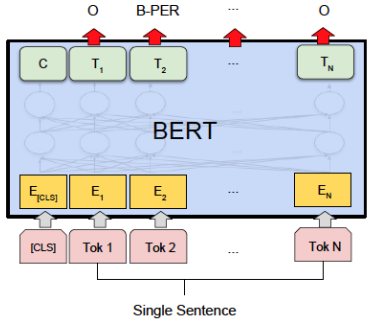
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



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

本章大纲

- 大语言模型发展历程
- 大语言模型原理介绍
  - Transformer
  - Encoder-only架构大模型
  - Decoder-only架构大模型
  - Encoder-Decoder架构大模型
- 大语言模型使用方式

OpenAI家族大模型发展历程

OpenAI 自2015年成立以来，推出了一系列具有重要影响力的人工智能模型。

- **GPT-1（2018年6月）**
  - OpenAI发布了首个生成预训练模型（GPT-1），将Transformer架构与无监督预训练相结合，开启了大型语言模型的研究之路。
- **GPT-2（2019年2月）**
  - GPT-2在GPT-1的基础上扩大了模型规模，拥有15亿参数，展现了强大的文本生成能力。由于担心滥用，最初未公开完整模型，后逐步发布。

OpenAI家族大模型发展历程

- **GPT-3（2020年6月）**
  - GPT-3拥有1750亿参数，显著提升了自然语言处理能力，能够执行多种任务，如翻译、问答和代码生成。
- **DALL·E（2021年1月）**
  - DALL·E是一种生成模型，能够根据文本描述生成图像，展示了多模态生成的潜力。
- **Codex（2021年8月）**
  - Codex是专门用于代码生成的模型，能够将自然语言描述转换为代码，成为GitHub Copilot的核心技术。

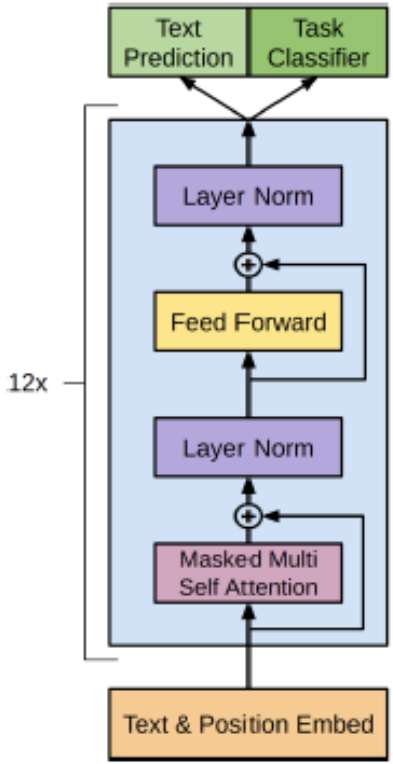
OpenAI家族大模型发展历程

- **CLIP（2021年1月）**
  - CLIP是一种多模态模型，能够理解图像和文本，并将它们映射到同一嵌入空间，实现跨模态检索和分类。
- **ChatGPT（2022年11月）**
  - 基于GPT-3.5系列微调而成的对话式AI模型，能够进行自然语言对话，展现了强大的交互能力。
- **GPT-4（2023年3月）**
  - GPT-4是大型多模态模型，能够处理文本和图像输入，进一步提升了理解和生成能力。

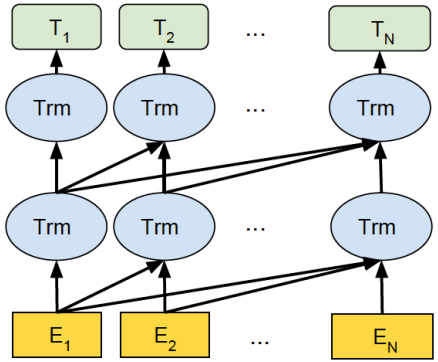
OpenAI家族大模型发展历程

- **o1（2024年9月）**
  - OpenAI推出的推理模型，专注于解决复杂问题，提升了模型的推理和逻辑能力。
- **o3（2024年12月）**
  - 最新的推理模型，性能超越o1，进一步提升了在复杂任务中的表现，特别是在编码和高级数学领域

GPT家族



模型结构



Masked Self-Attention

模型	发布时间	参数量	预训练数据
GPT	2018年6月	1.17亿	BookCorpus，约5GB
GPT-2	2019年2月	15亿	WebText， 40GB
GPT-3	2020年5月	1750亿	Common Crawl, WebText2, Book1, Book2, Wekipedia， 45TB， 3000亿token

GPT-3训练

- GPT-3的预训练为无监督预训练，指在大规模单语数据优化语言模型的任务
- 无微调的阶段
- 模型所建模的句子概率：

$$P(y_1, y_2, \ldots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \ldots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t}).$$

- 举例

P(I saw a cat on ...) =

$\underbrace{P(\text{I}) \cdot P(\text{saw}|\text{I}) \cdot P(\text{a}|\text{I saw}) \cdot P(\text{cat}|\text{I saw a}) \cdot P(\text{on}|\text{I saw a cat})}_{\text{Probability of I saw a cat on}} \cdot \underbrace{\ldots}$

情境学习（In-context learning）

微调：通过梯度更新模型

情境学习：基于任务描述或示例给出答案

Few-shot

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

One-shot

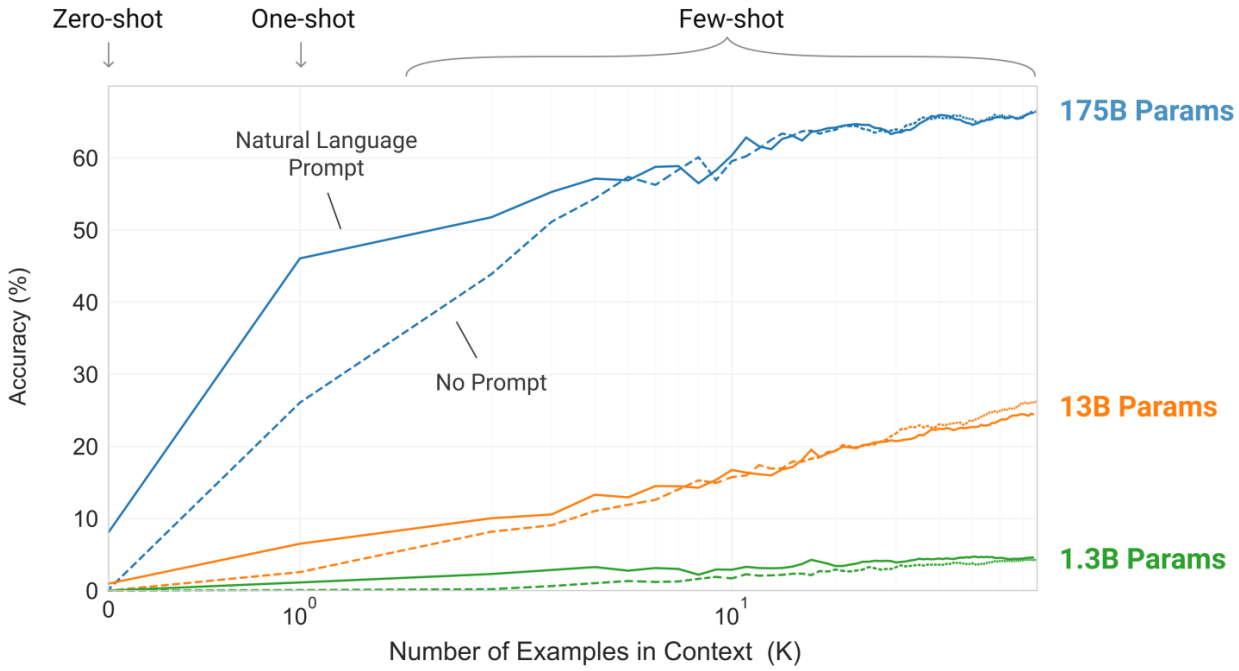
```
Translate English to French:        ← task description
sea otter => loutre de mer          ← example
cheese =>                           ← prompt
```

Zero-shot

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

情境学习与规模

- 更大的模型更可以充分利用情境信息。



GPT-3 Few-Shot性能

部分数据集上，基于GPT-3的Few-Shot学习超过SOTA系统（预训练+微调）

Cloze

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Open-domain QA

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Language Models are Few-Shot Learners

GPT-3 Few-Shot性能

部分数据集上，基于GPT-3的Few-Shot学习超过SOTA系统（预训练+微调）

Machine Translation

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0[b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

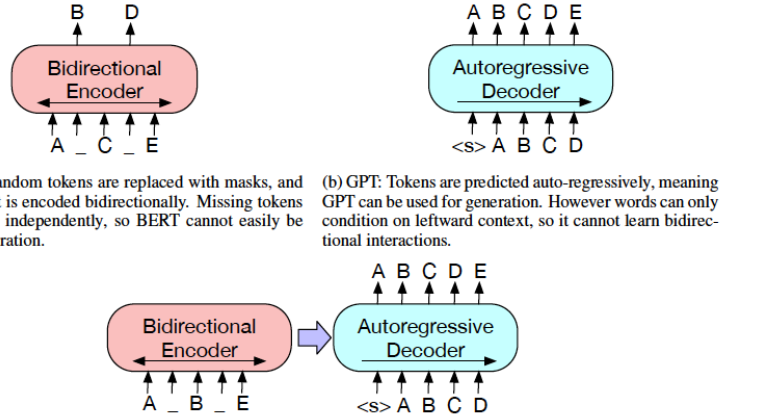
Language Models are Few-Shot Learners

本章大纲

- 大语言模型发展历程
- 大语言模型原理介绍
  - Transformer
  - Encoder-only架构大模型
  - Decoder-only架构大模型
  - Encoder-Decoder架构大模型
- 大语言模型使用方式

BART结构

- 使用方式为预训练-微调
- Base model的encoder和decoder均为6层，large model为12层



(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.

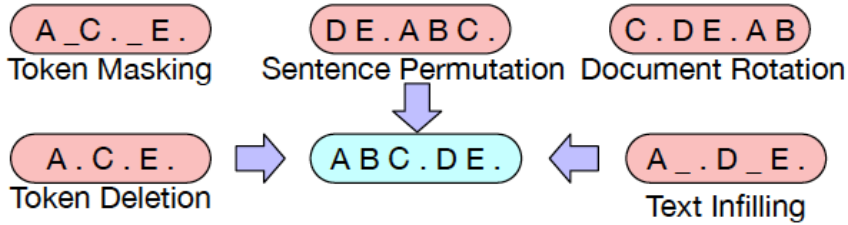
(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original 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.

BART与BERT、GPT结构对比

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

BART预训练

- Encoder添加噪声，decoder进行重构
- 训练损失为解码器输出序列与Ground Truth的词级别交叉熵
- Encoder噪声类别：
  - Token Masking：随机将词语替换为[Mask]
  - Token Deletion：随机删除词语

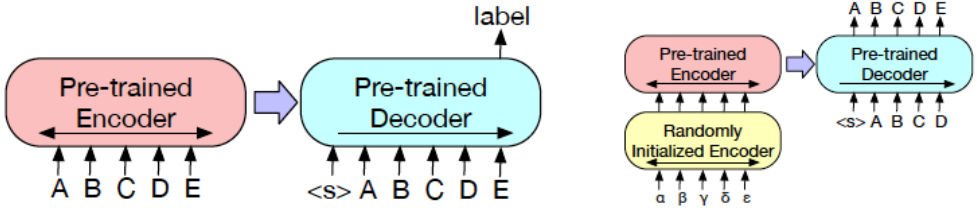


  - Token Infilling：随机将一个span替换为一个[Mask]，span长度从泊松分布采样，并修改position编号。
  - Sentence Permutation：根据句号来切分句子，随机将句子shuffle
  - Document Rotation：随机选一个token，将该token及后面部分作为开头，前面的部分放到后面。

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

BART微调

- 序列级分类任务
  - 将decoder输出的最后一个位置的hidden state后接分类器
- 序列生成
  - 源序列输入到encoder，decoder自回归式生成目标序列
- 机器翻译
  - 将BART的embedding替换为随机初始化的encoder
- 词级别分类任务
  - 将decoder输出的所有位置的hidden state均后接分类器



(a) To use BART for classification problems, the same input is fed into the encoder and decoder, and the representation from the final output is used.

(b) For machine translation, we learn a small additional encoder that replaces the word embeddings in BART. The new encoder can use a disjoint vocabulary.

- 端到端采用交叉熵进行训练：
  - 第一步：固定住其他部分，只训练添加的encoder、BART的position embedding以及BART encoder第一层的self-attention input projection matrix。
  - 第二步：训练所有参数

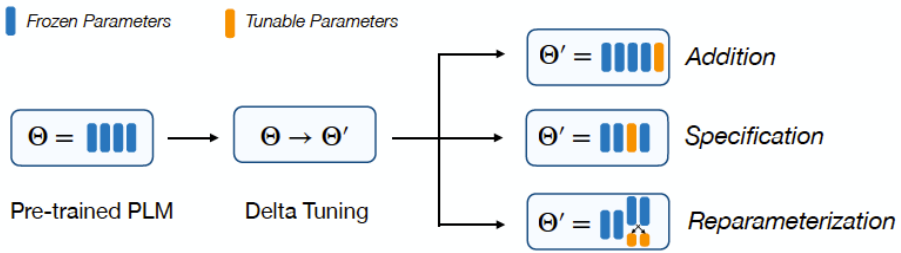
BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

本章大纲

- 大语言模型发展历程
- 大语言模型原理介绍
  - Transformer
  - Encoder-only架构大模型
  - Decoder-only架构大模型
  - Encoder-Decoder架构大模型
- 大语言模型使用方式

大语言模型下游使用方式

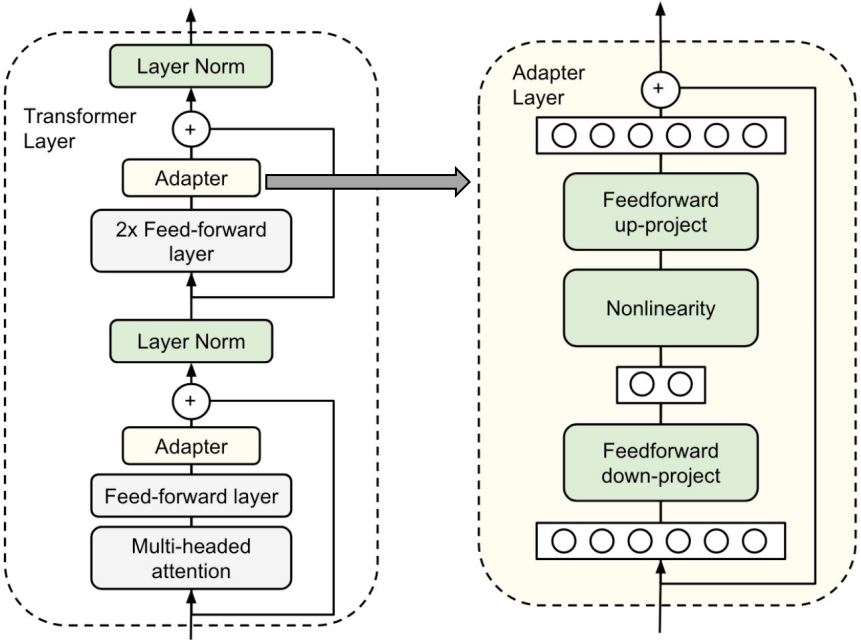
- Fine Tuning (FT)
- Delta Tuning/Parameter-efficient Fine Tuning
  - Addition-based Fine Tuning
    - Adapter-based FT
    - Prompt-based FT
  - Specification-based Fine Tuning
  - Reparameterization-based Fine Tuning
- Discrete Prompt (In-context Learning)



Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning
Delta Tuning : A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

Addition-based FT -- Adapters

- 添加adapters，大模型参数固定，只微调adapter参数

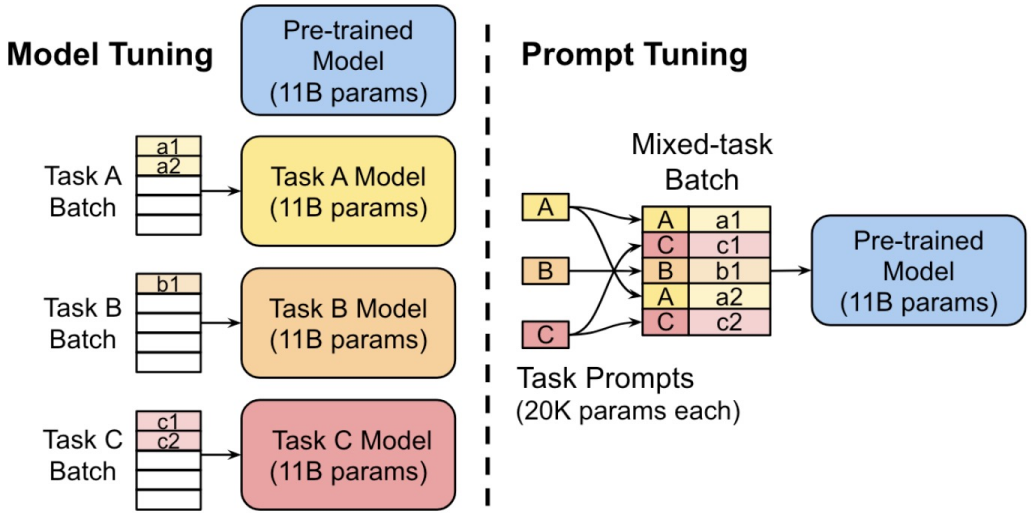


Parameter-Efficient Transfer Learning for NLP

Prompt-based FT -- Prompt Tuning

- 为每个任务添加soft prompt，拼接到输入序列，只训练prompt参数

- 初始化有三种方式：
  - 随机初始化
  - 初始化为某些词语的embedding
  - 分类问题，初始化为所有类别的embedding的拼接



The Power of Scale for Parameter-Efficient Prompt Tuning

Prompt-based FT -- Prefix-Tuning

- 为每个任务添加prompt，只优化prompt
- Prompt后接MLP，只保存MLP后表示
- 每一层均添加prompt

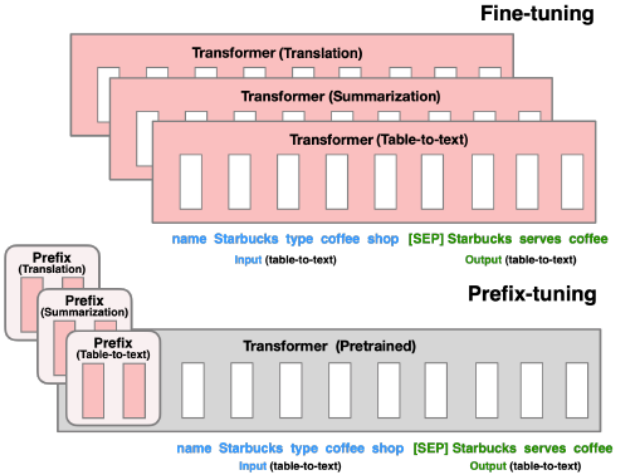


Figure 1: Fine-tuning (top) updates all Transformer parameters (the red Transformer box) and requires storing a full model copy for each task. We propose prefix-tuning (bottom), which freezes the Transformer parameters and only optimizes the prefix (the red prefix blocks). Consequently, we only need to store the prefix for each task, making prefix-tuning modular and space-efficient. Note that each vertical block denote transformer activations at one time step.

Prefix-Tuning: Optimizing Continuous Prompts for Generation

Prompt-based FT -- Prefix-Tuning

- Prefix-Tuning添加prefix方式
  - Encoder-Decoder结构，为[P, X, P', Y]
  - 自回归式结构，为[P, X, Y]

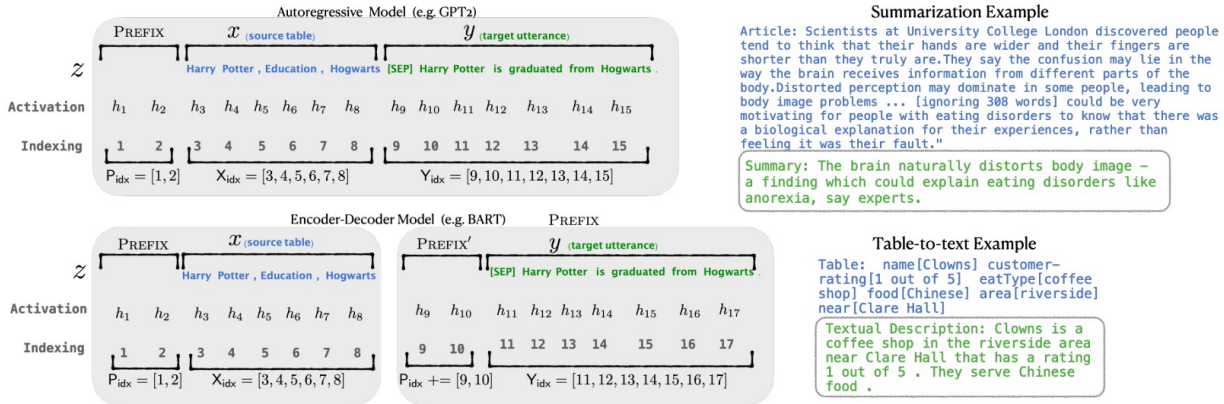


Figure 2: An annotated example of prefix-tuning using an autoregressive LM (top) and an encoder-decoder model (bottom). The prefix activations $\forall i \in P_{idx}, h_i$ are drawn from a trainable matrix P_θ. The remaining activations are computed by the Transformer.

Prefix-Tuning: Optimizing Continuous Prompts for Generation

Specification-based FT -- BitFit

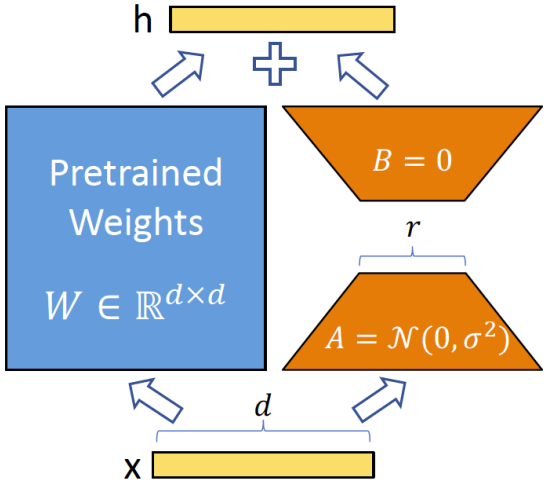
- Bias-terms Fine-tuning (BitFit)
  - 只训练bias

$$\mathbf{Q}^{m,\ell}(\mathbf{x}) = \mathbf{W}_q^{m,\ell}\mathbf{x} + \mathbf{b}_q^{m,\ell}$$
$$\mathbf{K}^{m,\ell}(\mathbf{x}) = \mathbf{W}_k^{m,\ell}\mathbf{x} + \mathbf{b}_k^{m,\ell}$$
$$\mathbf{V}^{m,\ell}(\mathbf{x}) = \mathbf{W}_v^{m,\ell}\mathbf{x} + \mathbf{b}_v^{m,\ell}$$

BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models

Reparameterization-based FT -- LoRA

- 只需训练两个低秩矩阵A和B



$$h = W_0 x + \Delta W x = W_0 x + BAx$$

LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

谢谢！