GPT

Decoder-Only 架构

特点：Decoder-Only 架构仅包含解码器部分。它设计用于生成任务，通过自回归方式逐个生成输出序列的元素。每个解码器层通常包含掩蔽的自注意力层，确保预测当前元素时只使用之前的元素，从而保持生成过程的因果关系。

应用场景：**文本生成**：如语言模型、机器翻译、文本摘要。**代码生成**：自动编写程序代码。**对话系统**：自动生成用户交互响应。

典型模型：

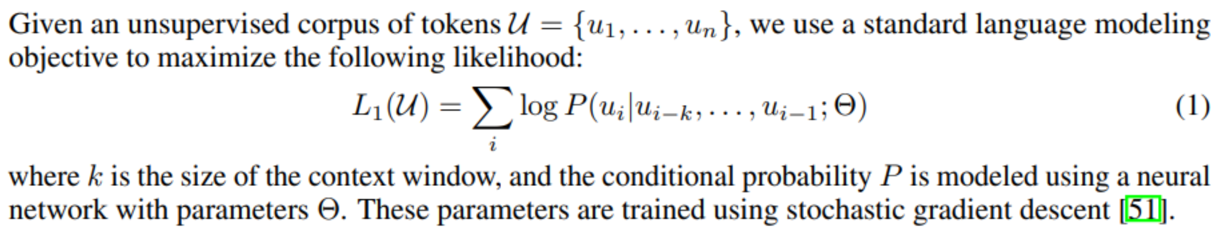
- GPT（Generative Pre-trained Transformer）系列是 Decoder-Only 架构的代表，广泛用于各种生成任务。
- Llama,llama2,llama3
- Qwen,Qwen2
- ....

GPT1

技术特点

- 采用大量unlabelled数据进行预训练，然后根据任务进行微调
- 微调时，调整输入不需要对于模型架构的改变，使用少量的标记数据
- **难点**
- 1.对于无标签数据损失函数的设定
- 2.由文本的输出到其他子任务的转化

模型架构

- 

  Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

  $$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \quad (1)$$

  where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].
- 目标函数为使得长为K 的序列用最大的概率与训练给出的文本尽可能的相同
- **编码器与解码器**
- 编码器无mask部分，目的是语义的融合->BERT->完形填空

- 解码器由mask部分，目的是预测输出->GPT->读后续写
- 预训练部分

  In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

  $$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \tag{2}$$

  where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

-
- 微调部分

  After training the model with the objective in Eq. 1, we adapt the parameters to the supervised target task. We assume a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label y. The inputs are passed through our pre-trained model to obtain the final transformer block's activation h_l^m, which is then fed into an added linear output layer with parameters W_y to predict y:

  $$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \tag{3}$$

  This gives us the following objective to maximize:

  $$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m). \tag{4}$$

  We additionally found that including language modeling as an auxiliary objective to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence. This is in line with prior work [50, 43], who also observed improved performance with such an auxiliary objective. Specifically, we optimize the following objective (with weight λ):

  $$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \tag{5}$$

  Overall, the only extra parameters we require during fine-tuning are W_y, and embeddings for delimiter tokens (described below in Section 3.3).

-
- L$_1$目标函数的目的是预测后面序列的概率最大化
- L$_2$目标函数的目的是预测句子标号概率的最大化
- L$_3$目标函数的目的是前两者的结合
- **微调时根据下游子任务的处理方式**

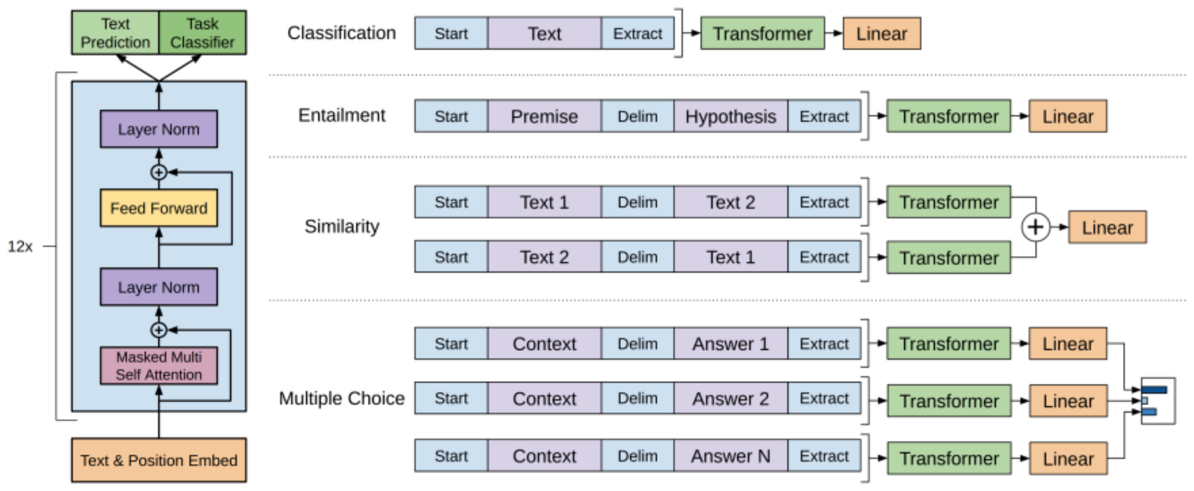


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

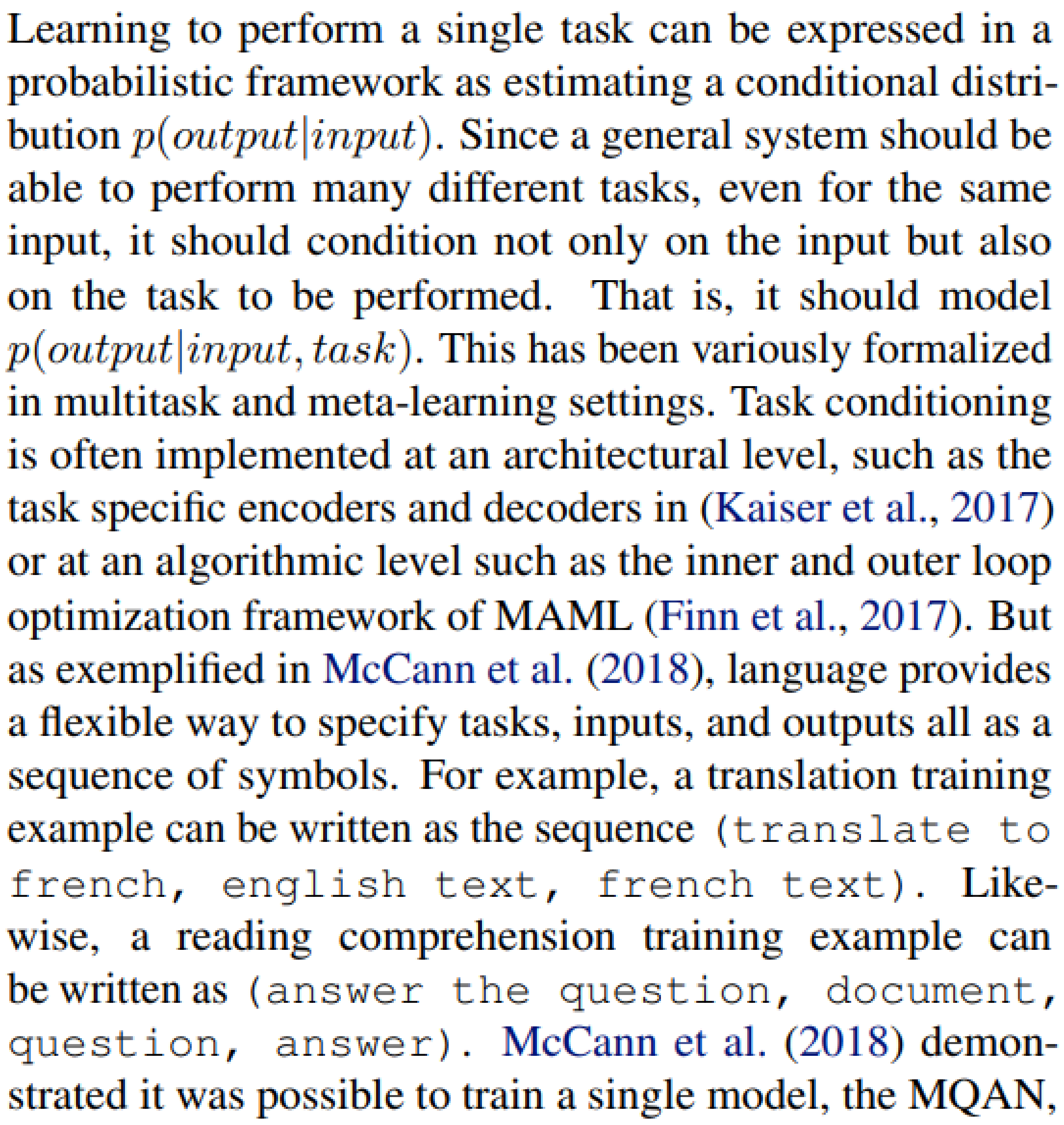
-

- [start],[delim],[extract]分别为开始，分割，截止符。

GPT2

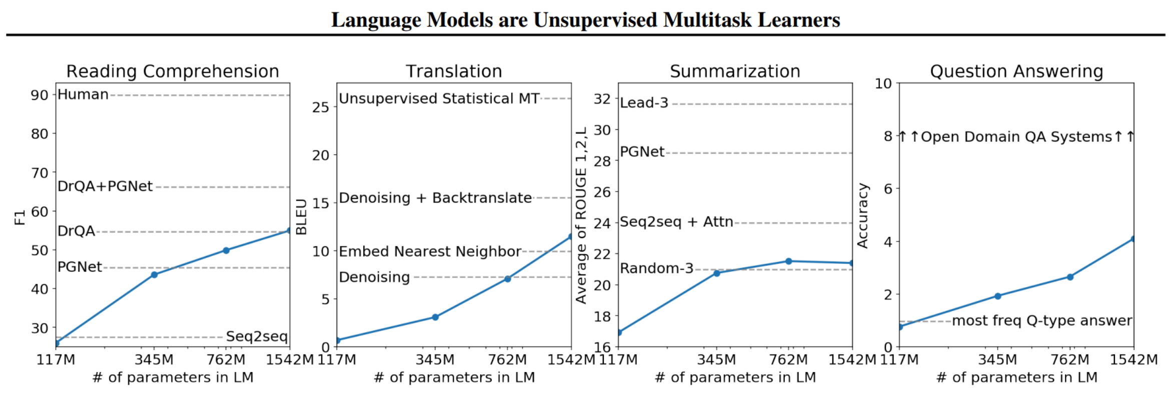
不同点

- zero-shot（类似于多模态）：在预训练结束后不需要根据不同的子任务使用标号数据训练
- 去除预训练时没有的分割符等。
- prompt（提示）：替代分割符



Learning to perform a single task can be expressed in a probabilistic framework as estimating a conditional distribution $p(output|input)$. Since a general system should be able to perform many different tasks, even for the same input, it should condition not only on the input but also on the task to be performed. That is, it should model $p(output|input, task)$. This has been variously formalized in multitask and meta-learning settings. Task conditioning is often implemented at an architectural level, such as the task specific encoders and decoders in (Kaiser et al., 2017) or at an algorithmic level such as the inner and outer loop optimization framework of MAML (Finn et al., 2017). But as exemplified in McCann et al. (2018), language provides a flexible way to specify tasks, inputs, and outputs all as a sequence of symbols. For example, a translation training example can be written as the sequence `(translate to french, english text, french text)`. Likewise, a reading comprehension training example can be written as `(answer the question, document, question, answer)`. McCann et al. (2018) demonstrated it was possible to train a single model, the MQAN,

-
- 效果一般，但是揭示多模态的潜力

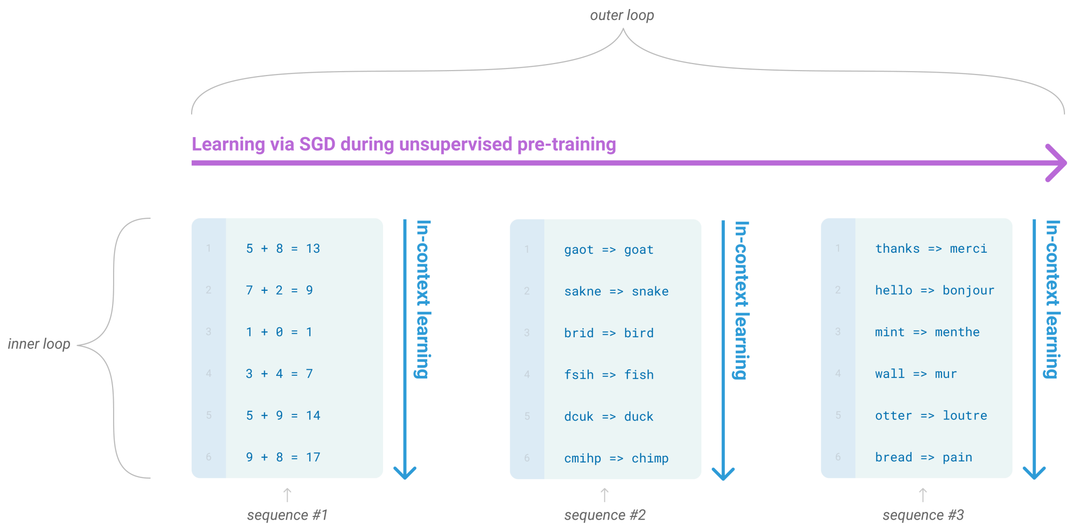


-
- 制作数据集时，通过爬取网页信息，并且通过读者反馈来筛选

GPT3

不同点

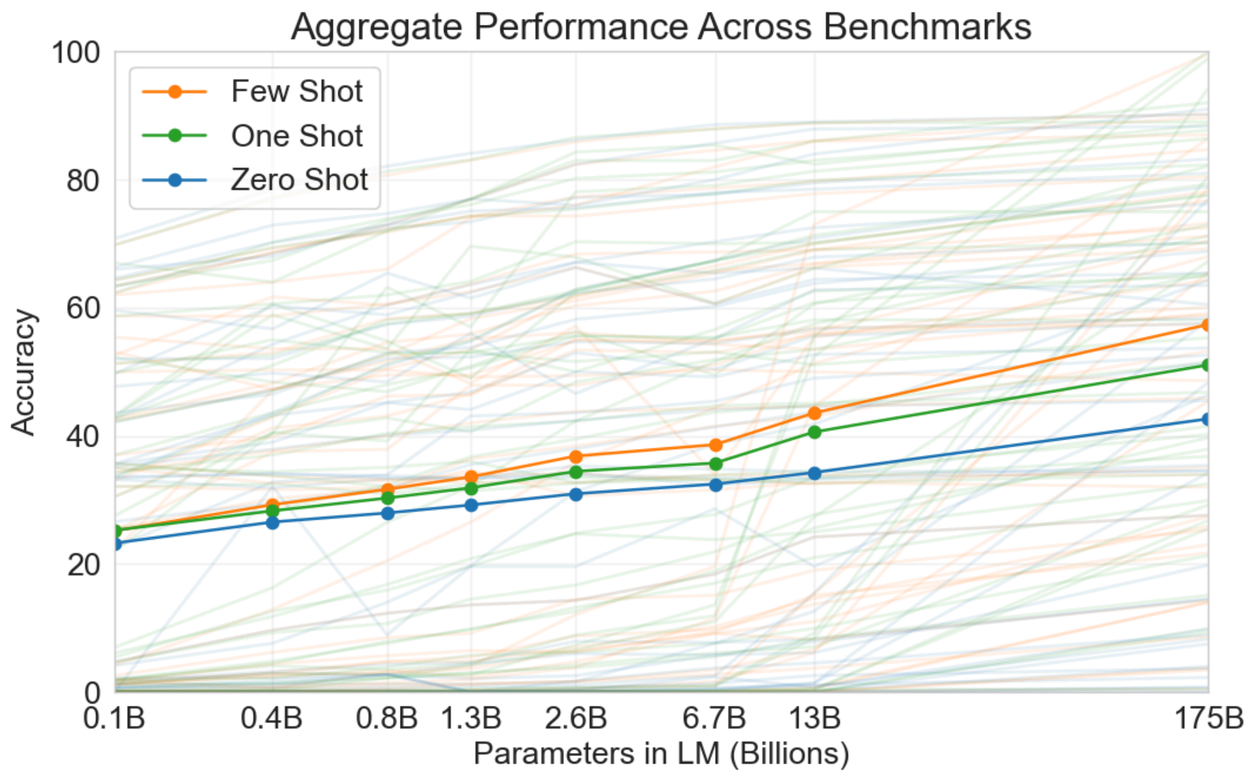
- 去除了微调的步骤，模型过大不好计算梯度
- GPT3中的Meta-Learning和In-Context Learning
- **?**
- GPT-3（Generative Pre-trained Transformer 3）中的 **Meta-Learning** 和 **In-Context Learning** 是理解其强大功能的关键概念。
- **1. Meta-Learning**
- **Meta-Learning** 是一种学习如何学习的能力。在GPT-3的上下文中，Meta-Learning意味着模型在训练过程中不仅仅是学习如何解决具体任务，还学习如何从有限的提示和上下文中快速适应新任务。GPT-3 在训练时接受了大量的通用文本数据，这使得它能够在看到新任务的少量示例时，快速地推断出解决该任务的策略。
- Meta-Learning的过程可以分为三个主要阶段：
- **预训练**：模型在大量的文本数据上进行训练，以学习广泛的语言模式和知识。
- **微调（可选）**：对于一些特定任务，模型可以通过微调来进一步优化其能力。
- **应用**：模型在实际使用时，可以根据输入的上下文迅速适应新任务，而不需要进一步的训练。这就是GPT-3的少样本学习（few-shot learning）能力。
- **2. In-Context Learning**
- **In-Context Learning** 是Meta-Learning的一种应用形式。在GPT-3中，这意味着模型可以根据输入的上下文（也就是提示中的示例）来推断出如何完成一项新任务，而不需要显式地调整模型的权重。简单来说，GPT-3能够从你给出的示例中“学习”并推断出你想要解决的问题。
- 例如，当你给GPT-3提供一些例子时，模型会“理解”这些例子，并在给定上下文的基础上生成相应的输出。这种能力允许GPT-3在不同任务之间无缝切换，并通过少量的上下文信息来推断出新任务的解法。
- **In-Context Learning的步骤：**
- **提供上下文**：用户在输入中提供一些示例或问题描述，作为模型的上下文。

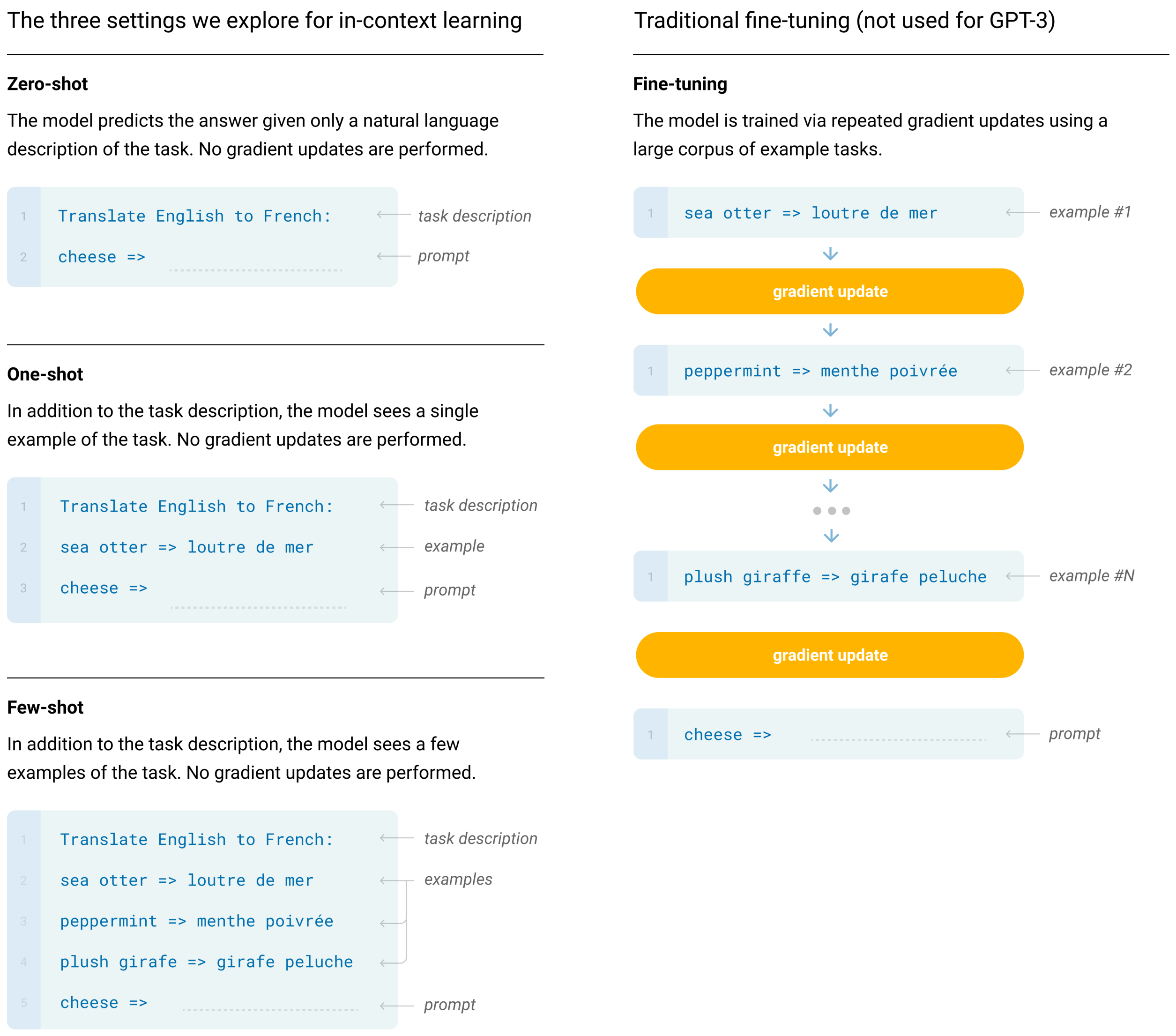
- **模型推断**：GPT-3根据输入的上下文，利用其在训练期间学习到的广泛知识和模式，来推断出如何生成与上下文一致的输出。
- **生成输出**：模型生成符合上下文的答案或响应。
- **总结**：
- **Meta-Learning** 是GPT-3在训练期间所获得的一种学习如何学习的能力，使得它能够从有限的上下文中快速适应新任务。
- **In-Context Learning** 是这种能力的具体体现，GPT-3能够根据输入的示例和上下文，立即推断并解决新问题，而无需进一步训练。
- 

```
5 + 8 = 13
7 + 2 = 9
1 + 0 = 1
3 + 4 = 7
5 + 9 = 14
9 + 8 = 17
```

```
gaot => goat
sakne => snake
brid => bird
fsih => fish
dcuk => duck
cmihp => chimp
```

```
thanks => merci
hello => bonjour
mint => menthe
wall => mur
otter => loutre
bread => pain
```

- 不同提示条件下的效果
- 


The three settings we explore for in-context learning
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
Traditional fine-tuning (not used for GPT-3)
Fine-tuning
The model is trained via repeated gradient updates using a large corpus of example tasks.
sea otter => loutre de mer
example #1
gradient update
peppermint => menthe poivrée
example #2
gradient update
plush giraffe => girafe peluche
example #N
gradient update
cheese =>
prompt

- 说明替代微调的过程，zero-shot，one-shot等时不更新参数的，而是将训练数据嵌入到提示词当中。
- Fine-tuning是需要更新参数的。
- 制作数据集时，将GPT-2中的数据集作为正类训练一个分类器，来对劣质数据进行分类->然后对于相似度过高的文章过滤

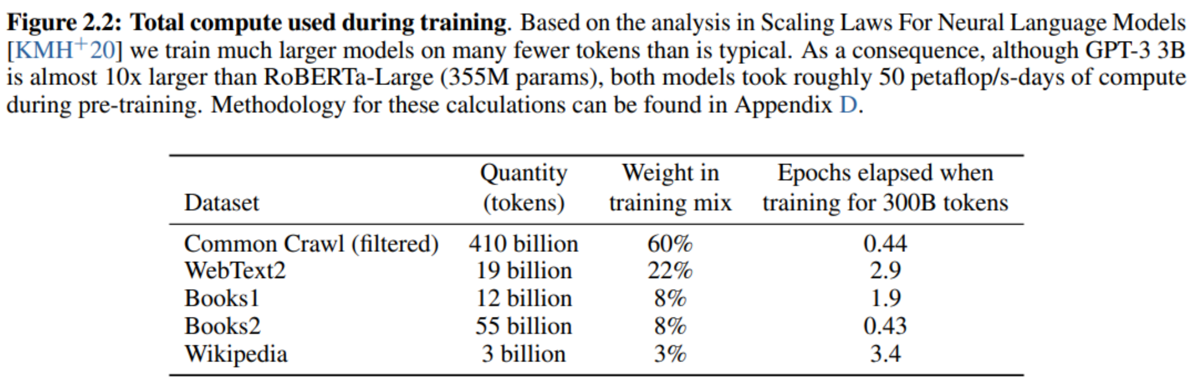
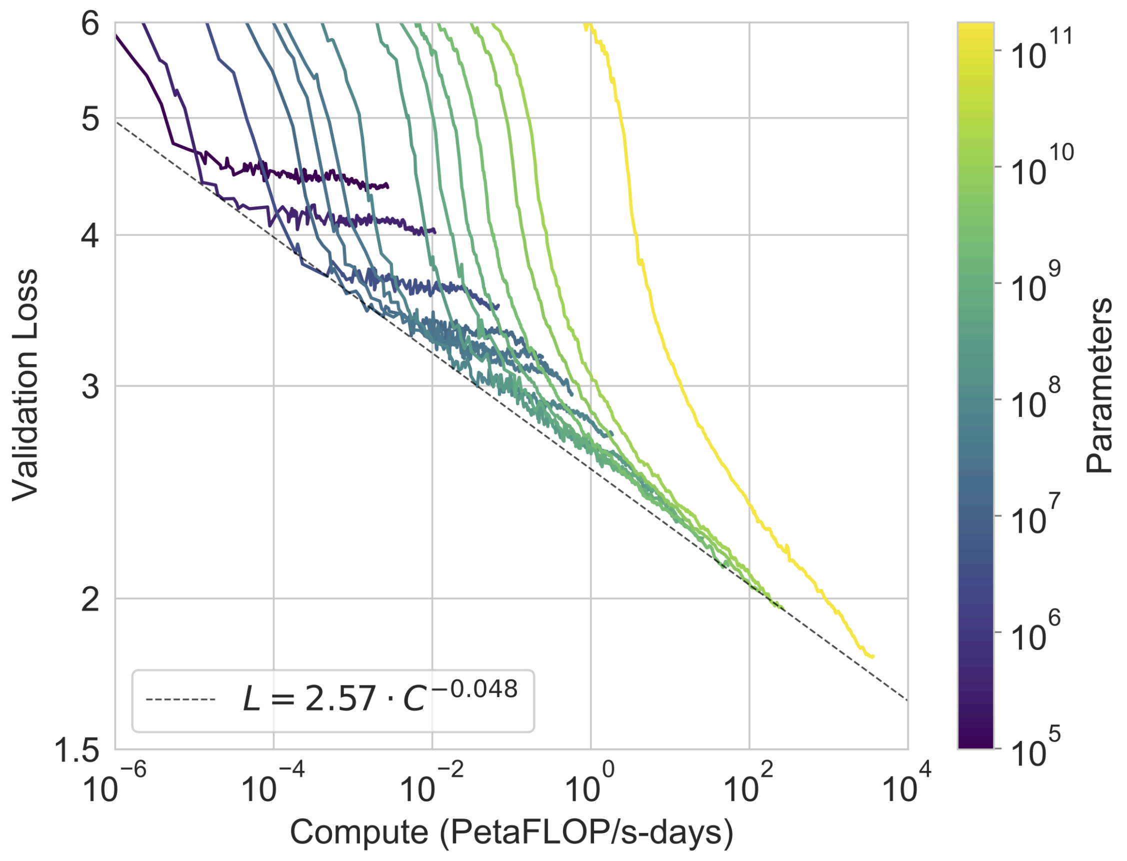


Figure 2.2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [KMH+20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

- 作者认为Common Crawl的质量不高，所以权重较低



-
- 根据这张图的可以大致预测出模型随着计算量的增加，损失值的拐点

局限性

- 语言生成仍然较弱：上下文联系过短
- 不能向BERT一样兼顾下文
- 在训练无法抓住重点的部分，均匀的去学习每一个词
- 不确定的点：在对子任务的处理时，模型时通过提示词从头学习（泛化性更强）还是调取预训练的数据来回答
- 黑盒效应