# GPT-2

GPT-2（**Generative Pre-trained Transformer 2**）是 OpenAI 在 2019 年推出的基于 **Transformer Decoder** 的大规模语言模型，它主要解决 **自回归语言建模（Autoregressive Language Modeling）** 任务，并展现出强大的 **零样本（Zero-Shot）** 和 **少样本（Few-Shot）** 迁移能力。

---

---

## （0）拓展

- **模型的容量**

受模型的参数数量，模型的训练时间，模型的训练数据量影响

## BLEU损失函数

这是一个基于`n-gram`的评判机器翻译的标准

- **参考翻译（人类标准答案）**：
  `"The cat is sitting on the mat."`
  （分词后：`["the", "cat", "is", "sitting", "on", "the", "mat"]`，共7个词）
- **机器翻译（待评估的句子）**：
  `"A cat is on the mat."`
  （分词后：`["a", "cat", "is", "on", "the", "mat"]`，共6个词）

---

#### 第一步：计算 1-gram 精度（$P_1$）

统计机器翻译中 **有多少单词** 出现在参考翻译中（重复词按参考中的次数上限计算）。

| 机器翻译单词 | 参考中出现的次数 | 机器中出现的次数 | 匹配次数（取最小值） |
| --- | --- | --- | --- |
| "a" | 0（参考中没有） | 1 | 0 |
| "cat" | 1 | 1 | 1 |
| "is" | 1 | 1 | 1 |
| "on" | 1 | 1 | 1 |
| "the" | 2 | 1 | 1 |

| 机器翻译单词 | 参考中出现的次数 | 机器中出现的次数 | 匹配次数（取最小值） |
|---|---|---|---|
| "mat" | 1 | 1 | 1 |

- **总匹配次数** = 0 (a) + 1 (cat) + 1 (is) + 1 (on) + 1 (the) + 1 (mat) = **5**
- **机器翻译总词数** = 6
- **1-gram 精度 $P_1$** = 匹配次数 / 机器总词数 = 5/6 ≈ 0.833

---

### 第二步：计算 2-gram 精度（$P_2$）

统计机器翻译中 **有多少连续的2词组合** 出现在参考翻译中。

- **机器翻译的2-gram列表**：
  `["a cat", "cat is", "is on", "on the", "the mat"]`（共5个）
- **参考翻译的2-gram列表**：
  `["the cat", "cat is", "is sitting", "sitting on", "on the", "the mat"]`
  （共6个）

| 机器2-gram | 是否在参考中? | 匹配次数 |
|---|---|---|
| "a cat" | 否 | 0 |
| "cat is" | 是 | 1 |
| "is on" | 否（参考中是"is sitting"） | 0 |
| "on the" | 是 | 1 |
| "the mat" | 是 | 1 |

- **总匹配次数** = 0 + 1 + 0 + 1 + 1 = **3**
- **机器2-gram总数** = 5
- **2-gram 精度 $P_2$** = 3/5 = 0.6

---

### 第三步：短句惩罚（Brevity Penalty, BP）

如果机器翻译比参考翻译短，分数会被打折。

- **机器翻译长度（c）** = 6
- **参考翻译长度（r）** = 7
- 因为 c < r，所以 BP = e^(1 - r/c) = e^(1 - 7/6) ≈ e^(-0.1667) ≈ **0.846**

- 公式

$$BP = e^{\frac{1-r}{c}}$$

---

**第四步：综合 BLEU 分数（假设只计算BLEU-2）**

BLEU 是几何平均 + 短句惩罚：

$$BLEU = BP * e^{\sum_{n=1}^{N} w_n * P_n}$$

其中假设

$w_1 = w_2 = 0.5$ （$N = 2$）

$BLEU = 0.846 \cdot exp(0.5 \cdot log(0.833) + 0.5 \cdot log(0.6)) =$

$0.846 \cdot exp(0.5 \cdot (-0.182) + 0.5 \cdot (-0.511)) =$

$0.846 \cdot exp(-0.346) = 0.846 \cdot 0.707 \approx 0.598$

所以得到的分数是0.598 **（满分是1.0）**

---

**BLEU 的局限性**

- **不懂语义**：只看单词是否相同，不管意思。比如“开心”和“高兴”明明同义，但BLEU会判为不匹配。
- **依赖参考翻译**：如果参考翻译本身不全面（比如只有一种表达方式），评分可能不公平。
- **对短句严格**：短句容易得低分，因为容错空间小。

总结来说，这像是一种严格的**背诵评价标准**，虽然是一个好的评判标准，但是却没有考虑到一些复杂的因素

---

**变体**

- **NIST**：加权 n-gram 重要性，减少常见词影响。
- **METEOR**：引入同义词匹配和词干分析，增强语义敏感性。
- **ROUGE**：专为摘要任务设计，关注召回率而非精确度。

---

---

# （1）目的

**GTP-2**解决的核心任务就是 **自回归语言建模**，通俗来说就是根据前文的所有token，预测后文的**一个token**

$$P(X_{i+1}|x_{1:i})$$

它的关键突破在于：

- **零样本学习（Zero-Shot Learning）**：不进行**微调（意思就是不需要针对特定的任务输入特定的数据和标签来学习，直接在预训练的模型上就可以来操作）**，直接通过 **任务提示（Task Prompting）** 完成多种 NLP 任务（如翻译、问答、摘要）。
  - 例如，输入 `"Translate English to French: 'hello' →"`，GPT-2 可能直接输出 `"bonjour"`。
- **多任务泛化能力**：单一模型可同时处理 **文本生成、分类、翻译** 等任务，无需额外训练，只需通过语言模型的训练

**论文解读：**

目前在语言任务上表现最佳的多任务学习系统，利用了预训练和监督微调的结合，通用的预训练系统可以在微调后在多个任务上表现良好，但微调仍需要监督数据。故本文做出证明：

- 大型语言模型可以在zero-shot设置中执行下游任务，而不需要任何参数或架构修改的微调

若目标为学习单一任务，可以用条件概率$P(x_i|x_{i-1},x_{i-2},\ldots,x_1)$表示，但一个通用系统应该能够针对具体任务并根据输入来生成输出，即

$$P(output \mid input, task)$$

故语言模型可以转换为用符号序列来指定任务，输入和输出表示。例如：

- 翻译任务可以写成 (translate to French，English text，French text)
- 阅读理解任务可以写成(answer the question，document，question，answer)

---

---

# （2）输入处理

## 2.1 BPE 分词Tokenizer

- **步骤**：
  1. 初始化词汇表为所有单字符（如英文a-z，标点符号）。
  2. 统计训练数据中所有相邻字符对的出现频率。

3. 合并最高频的字符对，形成新Token（如"qu"、"ing"）。
4. 重复合并，直到词汇表达到预设大小（GPT-2为50,257个Token）。

- **例子**：
  - 输入：`"ChatGPT is powerful."`
  - 分词结果：`["Chat", "G", "PT", " is", " powerful", "."]`
    （注：空格被保留为特殊Token "Ġ"）

**关键细节**

- **子词（Subword）处理**：
  罕见词（如"Tokenizer"）会被拆分为子词（`["Token", "izer"]`），避免OOV（Out-Of-Vocabulary）问题。
- **特殊Token**：
  - `⊲endoftext⊳`：文本结束标记。
  - `[CLS]`、`[SEP]`：在BERT中使用，GPT-2不需要（因其是单向模型）。

---

## 2.2 词嵌入

一个词嵌入矩阵，不必多说

---

## 2.3 位置编码

#### 1. 位置编码的作用

Transformer 模型（包括 GPT-2）的核心是 **自注意力机制**，但它本身是 **排列不变（Permutation Invariant）** 的，即输入序列的顺序变化不会影响注意力权重的计算。因此，必须显式地注入位置信息，让模型知道：

- **Token 的绝对位置**（例如“猫”是句子的第2个词）。
- **Token 的相对距离**（例如“猫”和“垫子”之间隔了3个词）。

---

#### 2. GPT-2 位置编码 vs 原始 Transformer

| 特性 | 原始 Transformer (Vaswani et al.) | GPT-2 |
| --- | --- | --- |
| **类型** | 固定正弦/余弦函数 | 可学习的位置嵌入（Learned Embeddings） |

| 特性 | 原始 Transformer (Vaswani et al.) | GPT-2 |
| --- | --- | --- |
| **数学形式** | 公式计算（无需训练） | 随机初始化后通过训练学习 |
| **灵活性** | 无法适应特定任务的位置模式 | 可自适应数据中的位置规律 |
| **长度扩展性** | 理论上支持任意长度（但实际有限制） | 最大长度固定（如1024） |

---

### 3. GPT-2 位置编码的数学实现

利用一个可以自动学习的位置编码矩阵，他同意能够学习到**相对信息和绝对信息**

#### (1) 可学习的位置嵌入矩阵

- 定义一个可训练的矩阵 $W_p \in R_N \times d$：
  - N：最大序列长度（如1024）。
  - d：嵌入维度（如768）。
- 对于序列中第 pos 个位置，其位置向量为：

  $p_{pos} = W_p[pos]$**（直接查表获取）**

#### (2) 与词嵌入的结合

- 词嵌入向量 ei 和位置向量 pi **相加**（非拼接）：

$$h_i = e_i + p_i$$

  其中：

  - $e_i$ 来自词嵌入矩阵 $W_e$。
  - $p_i$ 来自位置嵌入矩阵 $W_p$。

  **(3) 为什么用加法而非拼接?**
- **参数效率**：加法保持输入维度为 d，拼接会增至 2d，增加计算量。
- **实证效果**：实验表明加法足以让模型分离位置与语义信息。

---

### 4. 位置嵌入的初始化

- 通常随机初始化为小数值（如正态分布 $N(0, 0.02)$）。

- 在预训练中通过梯度下降逐步调整。

---

---

# （3）模型结构

## 3.1 多头注意力

| 特性 | 单头注意力 | 多头注意力 |
| --- | --- | --- |
| **计算复杂度** | 较低（单组QKV） | 较高（多组并行） |
| **表达能力** | 受限（单一模式） | 更强（多模式组合） |
| **参数量** | 较少 | 较多（线性变换矩阵增多） |
| **典型应用** | 早期简单模型 | Transformer/GPT/BERT等 |

多头注意力对于单头注意力，能够理解更多的模型含义，每个头相当于**映射到一个高维空间**，这个空间处理特定的问题

下面就是对于多头注意力的解释：

1. 首先我们要明确分为几个头：我们认为分为 $h$ 个头，也就是解决h个问题
2. 接着，相当于一个输入同时输入了多个注意力模块，这一步是**并行处理的**，同时，这个注意力需要 `mask`

对于每个头都有：

$Q = XW_q \ K = XW_k \ V = XW_{v,down}$

其中，所有的权重$W$ ， $W \in R^{d*\frac{d}{h}}$ ，也就是说，这里起到一个降维的作用

所以，最后得到的输出 $ W ∈R^{\frac{d}{h}*d}$，经过多头注意力，一共有这样的输出h个

3. concat操作

$$V = concat(V_1, V_2, ....., V_h)$$

得到$V \in R^{seq*d}$

4. 对于多头注意力的输出的理解

$$output = V * W_{v,up}$$

其中$W_{v,up} \in R^{d*d}$，这个权重是不改变形状的

## 3.2 全连接层

注意，这里的解码器不包含交叉注意力模块，因为它不需要整合来自外部的信息

我们省略了**归一化和残差连接**

全连接层也忽略

这个是全过程：

输入 $\rightarrow LayerNorm \rightarrow Masked\ \ SelfAttention \rightarrow$ 残差相加 $\rightarrow LayerNorm \rightarrow FFN \rightarrow$ 残差相加 $\rightarrow$

---

## 3.3 GELU激活函数

数学公式

$$GELU(x) = x \cdot \Phi(x)$$

$$\Phi(x) = \frac{1}{2}(1 + erf(\frac{\sqrt{x}}{2}))$$

近似公式

$$GELU(x) = 0.5x[1 + tanh(\sqrt{\frac{2}{\Pi}}(x + 0.044715x^3)]$$

- 当 x 较大时，Φ(x)≈1，此时 GELU(x)≈x（类似ReLU的线性区）。
- 当 x 为负时，Φ(x) 接近0，输出趋近于0，但梯度不会完全消失（不同于ReLU的硬截断）

 GELU

---

---

# （4）训练过程

其中绝大部分就是**自监督学习**，还可以配合少量的**监督学习**进行微调

## 4.1 训练数据与策略

- **数据集**：
  - 训练数据来自互联网文本（WebText），包含约4500万网页（40GB文本），经过质量过滤（如去重、移除低质内容）。

- **训练目标**：
  - 标准语言模型任务（自回归建模）：最大化下一个token的似然概率：

    P(xt丨x<t)=softmax(We·ht)

    其中ht为Transformer最后一层的输出，We为词嵌入矩阵。
- **优化细节**：
  - 使用Adam优化器（学习率2.5e-4，余弦衰减）。
  - 损失函数使用交叉熵函数
  - 批处理大小根据模型规模调整（Small版为64，XL版为512）。
  - 采用梯度裁剪（Clip Norm=1.0）和Dropout（率=0.1）防止过拟合。

---

## 4.2 生成策略

GPT-2的文本生成支持多种解码方法：

- **贪心搜索（Greedy Search）**：选择概率最高的token，简单但易重复。
- **束搜索（Beam Search）**：保留多个候选序列，平衡生成质量与多样性。
- **Top-k采样（k=40）**：从概率最高的k个token中随机选择，增加多样性。
- **温度调节（Temperature）**：调整softmax分布的平滑度（高温更随机，低温更确定）。

温度调节的公式

$$P(x_t) = \frac{e^{\frac{x_t}{T}}}{\sum_{i=1}^{N} e^{\frac{x_i}{T}}}$$

其中$T$为温度，这参考了热力学的公式，要是T很大，那么他们之间的差距就被缩小了，就有利于选择随机的，低温相反 **（一般T=2）**

---

## 4.3 输入格式

- 文本被分割为固定长度的片段（如1024个Token），超出部分截断。
- 使用特殊Token `⊲endoftext⊳` 标记文本结束

---

## 4.4 out of vocabulary（OOV）

**问题的来源**

- **词汇表限制**：GPT-2 的词汇表固定（50,257个Token），无法涵盖所有可能的单词或符号（如新术语、专有名词、罕见词）。
- **语言动态性**：新词不断出现（如网络流行语、科技名词），而预训练后词汇表无法更新。

在GPT2之中采用的策略就是BPE，但是也有缺陷

BPE 是一种 **子词分词（Subword Tokenization）** 算法，核心思想是将单词拆分为更小的子词单元，减少OOV概率。

即使使用BPE，GPT-2仍可能遇到以下OOV相关情况：

**(1) 未登录词（Unknown Words）**

- **原因**：某些专有名词或拼写错误可能无法被BPE覆盖。
- **例子**：
  - 输入：`"量子纠缠"`（假设训练数据中未出现）。
  - 分词结果：可能被拆分为生僻子词（如`["量", "子", "纠", "缠"]`），**导致语义丢失**。

**(2) 拼写变体与大小写**

- **问题**：GPT-2的BPE区分大小写，`"ChatGPT"`和`"chatgpt"`会**被视为不同Token**。
- **后果**：模型对大小写敏感，可能生成不一致的结果。

**(3) 特殊符号与多语言**

- **Emoji/符号**：如`"😂"`可能不在词汇表中，被拆分为乱码子词。
- **非英语文本**：对中文、日文等多语言支持有限（依赖训练数据中的子词分布）。

---

---

# （5）Detail

## 5.1. 模型规模与参数量的大幅提升

- **GPT-1**：
  参数量为1.17亿（117M），采用12层Transformer解码器，隐藏层维度768，注意力头数12。
- **GPT-2**：
  提供了多个版本，最大模型参数量达**15亿（1.5B）**（是GPT-1的10倍以上），层数增至48层，隐藏层维度1600，注意力头数25。
  - 其他版本包括：小型（124M）、中型（355M）、大型（774M）和超大（1.5B），用户可根据需求选择。

- **影响**：
  更大的参数量显著提升了模型的表征能力和语言生成质量，尤其在长文本连贯性、复杂语义理解上表现更优。

---

## 5.2. 训练数据规模与多样性的扩展

- **GPT-1**：
  使用BookCorpus数据集（约5GB文本），覆盖小说类内容，但多样性有限。
- **GPT-2**：
  引入**WebText**数据集（约40GB文本），从Reddit高赞外链中爬取，涵盖网页、新闻、科技、论坛等多样化文本，更接近真实语言分布。
- **改进点**：
  - 数据质量更高（过滤低质内容）。
  - 领域覆盖更广，减少模型偏见（尽管仍未完全解决）。
  - 数据量提升直接增强了模型的泛化能力。

---

## 5.3. 架构优化与训练策略改进

- **Layer Normalization的调整**：
  GPT-2将Layer Norm移至每个子模块（如注意力层、前馈层）的**输入部分**（类似原始Transformer），而GPT-1将其放在输出部分。这一调整提升了训练稳定性。

```
# Pre-LN Residual Connect module
class SubLayerConnection(nn.Module):
    """A SubLayer include A LN , A Function(FFN/self_attn) and A Dropout"""
    def __init__(self,size ,dropout):
        super(SubLayerConnection, self).__init__()
        self.norm = LayerNorm(size)
        self.dropout = nn.Dropout(dropout)

    def forward(self,x , sublayer):
        # x - - LN - attention/FNN - Dropout - - -. - --> x
        #   ' -  x - - - - - - - - - -  - - - - - '
        return x + self.dropout(sublayer(self.norm(x)))
```

- **残差连接缩放**：
  在残差路径上引入**缩放因子（1/√N）**（N为层数），防止深层网络梯度爆炸。

**原始残差连接**（如GPT-1或原始Transformer）：

$$Output = LayerNorm(x + Sublayer(x))$$

其中 `Sublayer` 可以是自注意力层或前馈神经网络（FFN）。

**GPT-2的改进**：

$$Output = LayerNorm(x + \frac{1}{\sqrt{N}} \cdot Sublayer(x))$$

缩放因子 $\frac{1}{\sqrt{N}}$ 随层数 N 增加而减小，确保深层网络中梯度幅度稳定。

直观理解：深层网络叠加时，未经缩放的残差路径可能导致梯度指数级增长（爆炸），缩放后梯度更可控。

- **上下文窗口扩展**：
  上下文长度从GPT-1的512 token扩展到**1024 token**，使模型能捕捉更长距离依赖。

---

## 5.4. 零样本（Zero-shot）与小样本（Few-shot）学习的突破

- **GPT-1**：
  需针对下游任务进行微调（Fine-tuning），依赖标注数据。
- **GPT-2**：
  提出 **“任务条件化”**（Task Conditioning）理念，通过自然语言提示（如“翻译为法语：…”）直接引导模型完成任务，无需微调。
  - 例如：在翻译、问答、摘要等任务中，GPT-2仅需输入任务描述即可生成结果。
  - 这一能力源于大规模**数据中隐含的任务多样性**，模型从中学习了任务与文本的映射关系。
- **意义**：
  证明了单一模型可通过纯无监督学习泛化到多任务，为后续GPT-3的“上下文学习”奠定基础。

---

## 5.5. 生成质量与多样性的提升

- **更连贯的长文本生成**：
  得益于更大上下文窗口和参数规模，GPT-2生成的文本在段落或篇章级别逻辑性更强。
- **减少重复与退化**：
  通过改进采样策略（如Top-k采样）和训练数据，降低了GPT-1中常见的重复生成问题。

---

## 5.6. 对安全性与伦理的初步探索

- **GPT-2的谨慎发布**：
  因担忧滥用风险（如生成假新闻、垃圾邮件），OpenAI分阶段开源模型（最初仅发布124M版本，后逐步开放更大模型）。
- **局限性**：
  GPT-2仍存在生成偏见或有害内容的问题，但这一讨论推动了后续研究（如GPT-3的**内容过滤机制**）。

---

## 局限性与后续发展

尽管GPT-2有显著改进，但仍存在生成事实性错误、缺乏可控性等问题。这些挑战在后续的GPT-3（更大规模、更优few-shot能力）和ChatGPT（基于人类反馈的强化学习）中得到进一步解决。但GPT-2作为承前启后的模型，其技术思路至今仍影响大语言模型的设计。