# ChatGLM 系列模型

## 1.ChatGLM

### 1.1 背景

主流的预训练框架主要有三种：

1. **autoregressive 自回归模型（AR 模型）**：代表作 GPT。本质上是一个 left-to-right 的语言模型。**通常用于生成式任务**，在长文本生成方面取得了巨大的成功，比如自然语言生成（NLG）领域的任务：摘要、翻译或抽象问答。当扩展到十亿级别参数时，表现出了少样本学习能力。缺点是单向注意力机制，在 NLU 任务中，无法完全捕捉上下文的依赖关系。
2. **autoencoding 自编码模型（AE 模型）**：代表作 BERT。是**通过某个降噪目标（比如 MLM）训练的双向文本编码器**。编码器会产出适用于 NLU 任务的上下文表示，但无法直接用于文本生成。
3. **encoder-decoder（Seq2seq 模型）**：代表作 T5。采用双向注意力机制，**通常用于条件生成任务**，比如文本摘要、机器翻译等。

三种预训练框架各有利弊，没有一种框架在以下三种领域的表现最佳：自然语言理解（NLU）、无条件生成以及条件生成。T5 曾经尝试使用 MTL 的方式统一上述框架，然而自编码和自回归目标天然存在差异，简单的融合自然无法继承各个框架的优点。

在这个天下三分的僵持局面下，GLM 诞生了。

**GLM 模型基于 autoregressive blank infilling 方法，结合了上述三种预训练模型的思想。**

### 1.2 GLM 预训练框架

GLM 特点

1. **自编码思想**：在输入文本中，随机删除连续的 tokens。

2. **自回归思想**：顺序重建连续 tokens。在使用自回归方式预测缺失 tokens 时，模型既可以访问 corrupted 文本，又可以访问之前已经被预测的 spans。
3. **span shuffling + 二维位置编码技术**。
4. 通过改变缺失 spans 的数量和长度，自回归空格填充目标可以为条件生成以及无条件生成任务预训练语言模型。

## （1）自回归空格填充任务

给定一个输入文本 $x = [x_1, \ldots x_n]$，可以采样得到多个文本 spans $\{s_1, \ldots s_m\}$。为了充分捕捉各 spans 之间的相互依赖关系，可以对 spans 的顺序进行随机排列，得到所有可能的排列集合 $Z_m$，其中：$S_{z<i} = [s_{z_1}, \ldots, s_{z_{i-1}}]$。所以预训练目标很清晰：

$$\max_{\theta} \mathbb{E}_{\boldsymbol{z} \sim Z_m} \left[ \sum_{i=1}^{m} \log p_\theta \left( \boldsymbol{s}_{z_i} \mid \boldsymbol{x}_{\text{corrupt}}, \boldsymbol{s}_{\boldsymbol{z}_{<i}} \right) \right]$$

GLM 自回归空格填充任务的技术细节：

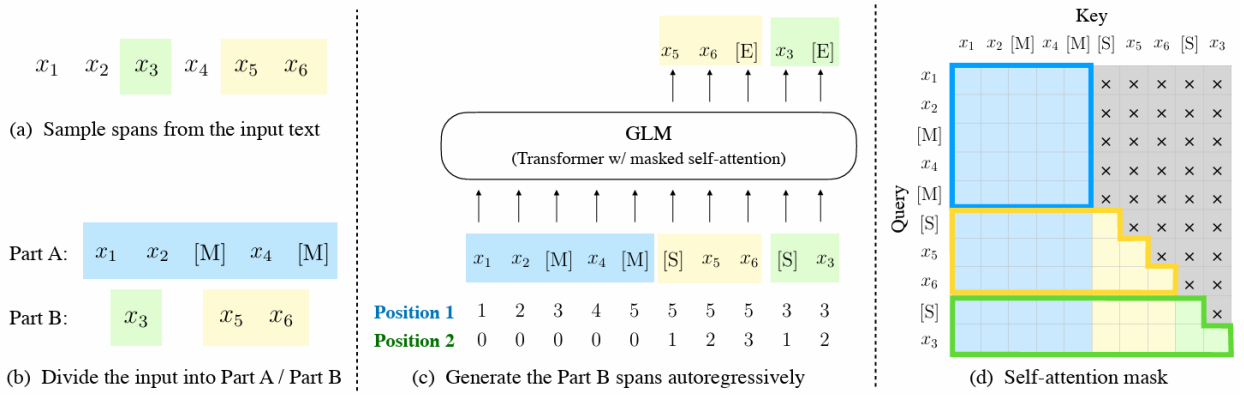
1. 输入 $x$ 可以被分成两部分：Part A 是被 mask 的文本 $x_{\text{corrupt}}$，Part B 由 masked spans 组成。假设原始输入文本是 $[x1, x2, x3, x4, x5, x6]$，采样的两个文本片段是 $[x3]$ 以及 $[x5, x6]$。那么 mask 后的文本序列是：$x1, x2, [M], x4, [M]$，即 Part A；同时我们需要对 Part B 的片段进行 shuffle。每个片段使用 `[S]` 填充在开头作为输入，使用 `[E]` 填充在末尾作为输出。
2. **二维位置编码**：Transformer 使用位置编码来标记 tokens 中的绝对和相对位置。在 GLM 中，使用二维位置编码，第一个位置 id 用来标记 Part A 中的位置，第二个位置 id 用来表示跨度内部的相对位置。这两个位置 id 会通过 embedding 表被投影为两个向量，最终都会被加入到输入 token 的 embedding 表达中。
3. 观察 GLM 中自定义 attention mask 的设计，非常巧妙：

    a. Part A 中的 tokens 彼此可见，但是不可见 B 中的任意 tokens。

    b. Part B tokens 可见 Part A。

    c. Part B tokens 可见 B 中过去的 tokens，不可见 B 中未来的 tokens。

4. 采样方式：文本片段的采样遵循泊松分布，重复采样，直到原始 tokens 中有 15% 被 mask。
5. 总结：模型可以自动学习双向 encoder（Part A）以及单向 decoder（Part B）。




## （2）多目标预训练

上述方法适合于 NLU 任务。作者希望可以训练一个既可以解决 NLU 任务，又具备文本生成能力的模型。因此除了空格填充目标之外，还需要增加一个生成长文本目标的任务。具体包含以下两个目标：

1. 文档级别。从文档中采样一个文本片段进行 mask，且片段长度为文档长度的 50%～100%。这个目标用于长文本生成。
2. 句子级别。限制被 mask 的片段必须是完整句子。多个片段需覆盖原始 tokens 的 15%。这个目标是用于预测完整句子或者段落的 seq2seq 任务。

## （3）模型结构

GLM 在原始 single Transformer 的基础上进行了一些修改：

1. 重组了 LN 和残差连接的顺序；
2. 使用单个线性层对输出 token 进行预测；
3. 激活函数从 ReLU 换成了 GeLUS。

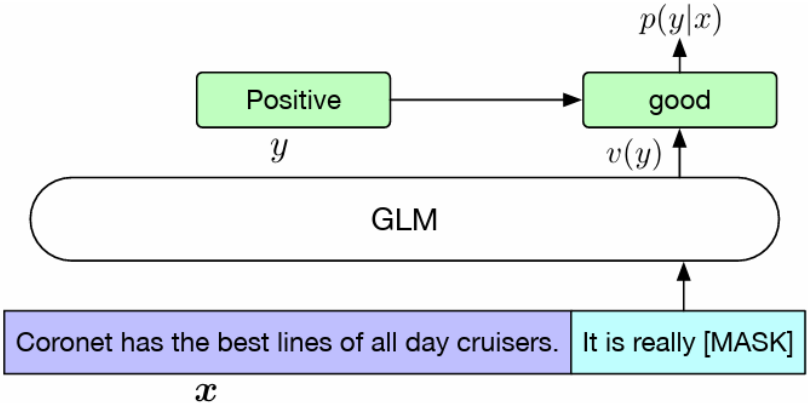
但我觉得这部分的修改比较简单常见。核心和亮点还是空格填充任务的设计。

## （4）GLM 微调

对于下游 NLU 任务来说，通常会将预训练模型产出的序列或 tokens 表达作为输入，使用线性分类器预测 label。所以预训练与微调之间存在天然不一致。

作者按照 PET 的方式，将下游 NLU 任务重新表述为空白填充的生成任务。具体来说，比如给定一个已标注样本(x, y)，将输入的文本 x 转换成一个包含 mask token 的完形填空问题。比如，情感分类任务可以表述为："{SENTENCE}. It's really [MASK]"。输出 label y 也同样会被映射到完形填空的答案中。“positive”和“negative”对应的标签就是“good”和“bad。

其实，预训练时，对较长的文本片段进行 mask，以确保 GLM 的文本生成能力。但是在微调的时候，相当于将 NLU 任务也转换成了生成任务，这样其实是为了适应预训练的目标。但难免有一些牵强。




| BERT | XLNet | T5 | UniLM |
| --- | --- | --- | --- |
| 1、无法捕捉 mask tokens 的相互依赖性。2、不能准确填充多个连续的 tokens。为了推断长度为 l 的答案概率，BERT 需要执行 l 次连续预测。 | 与 GLM 相同，使用自回归目标预训练。1、使用文本 mask 之前的原始位置编码，推理过程中，需要事先知晓或枚举答案长度，与 BERT 的问题相同。2、双流自注意力机制，使预训练时间成本增加了一倍。 | 使用类似的空格填充目标预训练 encoder-decoder Transformer。在编码和解码阶段使用独立的位置编码，使用多个哨兵 token 来区分 mask 片段。而在下游任务中，仅使用一个哨兵 token，造成模型能力的浪费以及预训练-微调的不一致。 | 通过改变双向、单向以及交叉注意力之间的注意力 mask，统一不同的预训练目标。1、总是使用 [mask] token 替代 mask 片段，限制了它对 mask 片段及其上下文的依赖关系进行建模的能力。2、在下游任务微调时，自编码比自回归更加低效。 |

# 2.ChatGLM-2

## 2.1 主要创新

1. **更长的上下文**：**基于 FlashAttention 技术**，将基座模型的上下文长度（Context Length）由 ChatGLM-6B 的 **2K 扩展到了 32K**，并在对话阶段使用 8K 的上下文长度训练。对于更长的上下文，发布了 ChatGLM2-6B-32K 模型。LongBench 的测评结果表明，在等量级的开源模型中，ChatGLM2-6B-32K 有着较为明显的竞争优势。
2. **更强大的性能**：基于 ChatGLM 初代模型的开发经验，全面升级了 ChatGLM2-6B 的基座模型。ChatGLM2-6B **使用了 GLM 的混合目标函数**，经过了 1.4T 中英标识符的预训练与人类偏好对齐训练，评测结果显示，相比于初代模型，ChatGLM2-6B 在 MMLU（+23%）、CEval（+33%）、GSM8K（+571%）、BBH（+60%）等数据集上的性能取得了大幅度的提升，在同尺寸开源模型中具有较强的竞争力。
3. **更高效的推理**：基于 Multi-Query Attention 技术，ChatGLM2-6B 有更高效的推理速度和更低的显存占用：在官方的模型实现下，推理速度相比初代提升了 42%，INT4 量化下，6G 显存支持的对话长度由 1K 提升到了 8K。
4. **更开放的协议**：ChatGLM2-6B 权重对学术研究**完全开放**，在填写问卷进行登记后**亦允许免费商业使用**。

## 2.2 与 ChatGLM 的变化

1. **使用了 RoPE 替换二维位置编码**。这也是 GLM 中提出的亮点设计之一。但是目前大部分主流的 LLMs 都在使用 RoPE，所以大势所趋。当前版本仍然采用了最初的 RoPE 设计，事实上现在的 RoPE 经过了 xPOS→线性内插→NTK-Aware Scaled RoPE→...若干次进化。
2. **Multi-Query Attention**：这是一种共享机制的 Attention，相比 Multi-Head Attention，其 Query 部分没有区别，Key 和 Value 可以只用一个 Head。计算时，对 Key 和 Value 进行 expand 或者 repeat 操作，使它们填充到与 Query 一样的维度，后续计算就与 Multi-Head Attention 没区别。

3. **Attention Mask**: V1 的 attention mask 分了 2 部分，Part A 和 Part B，Part A 部分是双向 Attention（代码中的 prefix_attention_mask），Part B 部分是 Causal Attention(原代码文件中的 get_masks 函数)。在 V2 版本，全部换成了 Causal Attention，不再区分是 Part A 还是 Part B，**完全变成了 decoder-only 的架构**。
4. **多目标任务**：Chat 版本主要还是用的 gMask 生成式任务，但是在 V1 版本的代码还能看到 mask、gMask 等字样，V2 已经摒弃了这些特殊 token，原因与 Attention Mask 一致，均因为变成了 decoder-only 的架构，不再需要区分 Part A 和 Part B。

# 3.ChatGLM-3

省流：**ChatGLM2 与 ChatGLM3 模型架构是完全一致的**，ChatGLM 与后继者结构不同。可见 ChatGLM3 相对于 ChatGLM2 没有模型架构上的改进。

相对于 ChatGLM，ChatGLM2、ChatGLM3 模型上的变化：

1. 词表的大小从 ChatGLM 的 150528 缩小为 65024 （一个直观的体验是 ChatGLM2、3 加载比 ChatGLM 快不少）
2. **位置编码从每个 GLMBlock 一份提升为全局一份**
3. **SelfAttention 之后的前馈网络有不同**。ChatGLM 用 GELU（Gaussian Error Linear Unit）做激活；ChatGLM 用 Swish-1 做激活。而且 ChatGLM2、3 应该是修正了之前的一个 bug，因为 GLU（Gated Linear Unit）本质上一半的入参是用来做门控制的，不需要输出到下层，所以 ChatGLM2、3 看起来前后维度不一致（27392->13696)反而是正确的。