

LLM主流开源大模型介绍

一样的教育，不一样的品质



目录

Contents

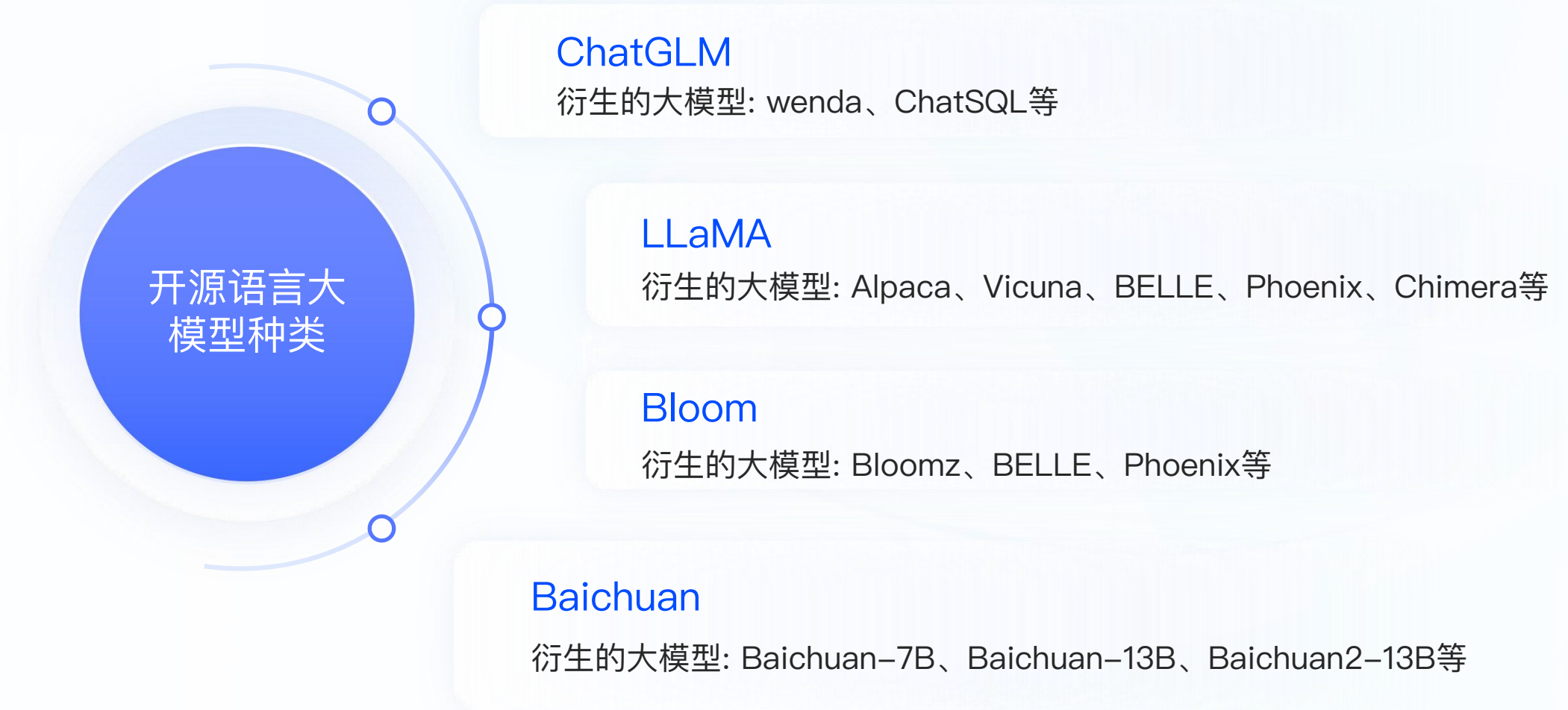
1. LLM主流大模型类别
2. ChatGLM-6B模型
3. LLaMA模型
4. BLOOM模型
5. Baichuan-7B模型

01

LLM主流大模型类别

LLM主流大模型类别

随着ChatGPT迅速火爆，引发了大模型的时代变革，国内外各大公司也快速跟进生成式AI市场，近百款大模型发布及应用。



开源语言大
模型种类

The diagram features a central blue circle with the text '开源语言大模型种类' (Open-source language model types). To its right, four light blue rounded rectangular boxes are arranged vertically, each containing the name of a model family and a list of its derivatives. A thin blue arc with three small circles connects the central circle to the first three boxes (ChatGLM, LLaMA, and Bloom). The fourth box (Baichuan) is positioned below the others and is not connected by the arc.

ChatGLM

衍生的大模型: wenda、ChatSQL等

LLaMA

衍生的大模型: Alpaca、Vicuna、BELLE、Phoenix、Chimera等

Bloom

衍生的大模型: Bloomz、BELLE、Phoenix等

Baichuan

衍生的大模型: Baichuan-7B、Baichuan-13B、Baichuan2-13B等

02

ChatGLM-6B模型

ChatGLM-6B模型简介

- ChatGLM-6B 是清华大学提出的一个开源、支持中英双语的对话语言模型，基于 General Language Model (GLM) 架构，具有 62 亿参数.
- 该模型使用了和 ChatGPT 相似的技术，经过约 1T 标识符的中英双语训练(中英文比例为 1:1)，辅以监督微调、反馈自助、人类反馈强化学习等技术的加持，62 亿参数的 ChatGLM-6B 已经能生成相当符合人类偏好的回答（目前中文支持最好）.

I 训练目标

上图说明了GLM的实现思想（训练目标）：

- 原始文本 $x=[x_1, x_2, \dots, x_6]$ 随机进行连续 mask，这里假设 mask 掉 $[x_3]$ 和 $[x_5, x_6]$ 。
- 将 $[x_3]$ 和 $[x_5, x_6]$ 替换为 $[M]$ 标志，并打乱 Part B 的顺序。为了捕捉跨度之间的内在联系，随机交换跨度的顺序。
- GLM 自回归地生成 Part B。每个片段在输入时前面加上 $[S]$ ，在输出时后面加上 $[E]$ 。二维位置编码表示不同片段之间和片段内部的位置关系。
- 自注意力掩码。灰色区域被掩盖。Part A 的词语可以自我看到（图蓝色框），但不能看到 Part B。Part B 的词语可以看到 Part A 和 Part B 中的前面的词语（图黄色和绿色框对应两个片段）。 $[M] := [MASK]$, $[S] := [START]$, $[E] := [END]$

I 模型结构

采用transformer的decoder模块，因为无论是对于自然语言理解还是自然语言生成类任务，GLM都是看成生成任务做.但是这里只能说类deocder，因为decoder是单向的，但是GLM某些位置可以看到双向的，因此又被称为Prefix – Decoder.

相比原始Decoder模块，模型结构有如下改动点：

embedding 层梯度
缩减

为了提升训练稳定性，减小了 embedding 层的梯度。梯度缩减的效果相当于把 embedding 层的梯度缩小了 10 倍，减小了梯度的范数.

layer
normalization

采用了基于 Deep Norm 的 post layer norm.

激活函数

替换ReLU激活函数采用了 GeGLU 激活函数.

位置编码

去除了绝对位置编码，采用了旋转位置编码 RoPE.

I 模型结构

部分名词解析：

Deep Norm

```
def deepnorm(x):  
    return LayerNorm(x *  $\alpha$  + f(x))
```

GeGLU激活函数

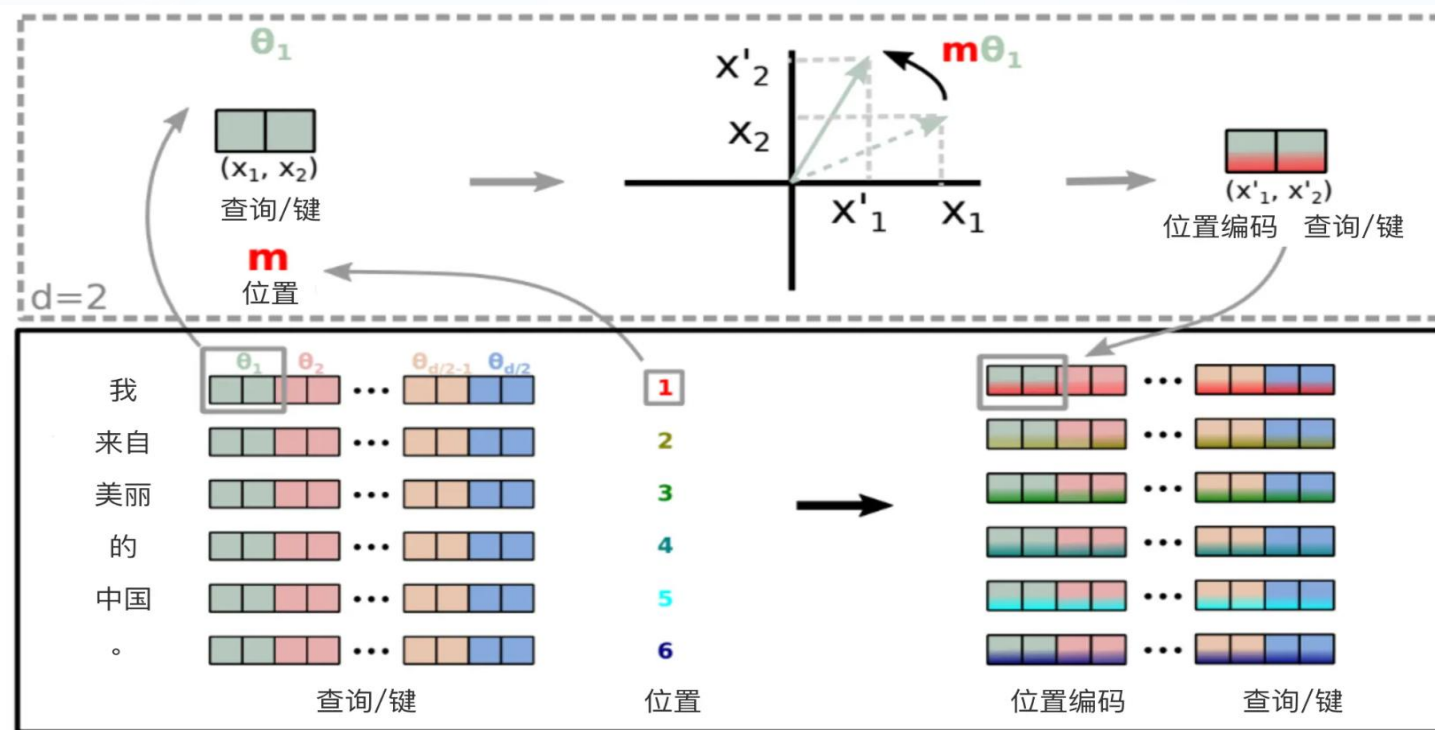
$$\text{GELU}(x) = 0.5x \left(1 + \tanh \left(\sqrt{2/\pi}(x + 0.044715x^3) \right) \right)$$

$$\text{GEGLU}(x, W, V, b, c) = \text{GELU}(xW + b) \otimes (xV + c)$$

模型结构

部分名词解析：

RoPE旋转位置编码



I 模型配置（6B）与硬件要求

模型配置如下:

配置	数据
参数	6.2B
隐藏层维度	4096
层数	28
注意力头数	32
训练数据	1T
词表大小	130528
最大长度	2048

硬件要求如下:

量化等级	最低GPU显存（推理）	最低GPU显存（高效参数微调）
FP16(无量化)	13GB	14GB
INT8	10GB	9GB
INT4	6GB	7GB

I 模型特点

优点

- 较低的部署门槛：INT4 精度下，只需6GB显存，使得 ChatGLM-6B 可以部署在消费级显卡上进行推理.
- 更长的序列长度：相比 GLM-10B（序列长度1024），ChatGLM2-6B 序列长度达32K，支持更长对话和应用。
- 人类类意图对齐训练

缺点

- 模型容量小，相对较弱的模型记忆和语言能力。
- 较弱的多轮对话能力。

I 衍生应用

langchain-
ChatGLM

基于 langchain 的 ChatGLM 应用，实现基于可扩展知识库的问答。

大型语言模型调用平台，基于 ChatGLM-6B 实现了类 ChatPDF 功能。

闻达

I 迭代版本

ChatGLM2-6B: ChatGLM2-6B是ChatGLM-6B的第二代版本, 相比第一代, 它带来了一系列显著的优势:

01

更强大的性能

在各项对话任务中, ChatGLM2-6B相比ChatGLM-6B有了巨大的提升。例如, 在数学任务上, 性能提升了整整571%

02

更长的上下文

ChatGLM2-6B采用了FlashAttention技术, 使其支持32K的上下文长度, 而ChatGLM-6B只能支持2K

03

更高效的推理

ChatGLM2-6B引入了Multi-Query Attention技术, 在更低的显存资源下以更快的速度进行推理, 相比第一代提升了42%.

I 迭代版本

ChatGLM3-6B: ChatGLM3-6B是ChatGLM-6B的第三代版本, 相比前两代, 除了继承之前优势外, 相当于全面升级:

01

多模态理解能力

在10余个国际标准图文评测集上取得SOTA.

02

代码增强模块

根据用户需求生成代码并执行, 自动完成数据分析、文件处理等复杂任务

03

网络搜索增强

能自动根据问题在互联网上查找相关资料并在回答时提供参考相关文献或者文章链接.



思考总结

Thinking summary

1. ChatGLM-6B的模型架构?

答案: Prefix-Decoder-Only:一种基于GLM的自回归空白填充目标的通用预训练模型

2. ChatGLM-6B的训练目标?

答案: 在输入文本中随机挖去一些连续的文本片段, 然后训练模型按照任意顺序重建这些片段.

3. ChatGLM-6B模型的改动点?

答案: Embedding层梯度缩减; Deep Norm; GeGLU激活函数; RoPE位置编码

03

LLaMA模型

LLaMA模型简介与训练目标

LLaMA (Large Language Model Meta AI) , 由 Meta AI 于2023年发布的一个开放且高效的大型基础语言模型, 共有 7B、13B、33B、65B (650 亿) 四种版本.

LLaMA训练数据是以英语为主的拉丁语系, 另外还包含了来自 GitHub 的代码数据。训练数据以英文为主, 不包含中韩日文, 所有训练数据都是开源的。其中LLaMA-65B 和 LLaMA-33B 是在 1.4万亿 (1.4T) 个 token上训练的, 而最小的模型 LLaMA-7B 和LLaMA-13B 是在 1万亿 (1T) 个 token 上训练的.

LLaMA 的训练目标是语言模型, 即根据已有的上文去预测下一个词.

关于tokenizer, LLaMA 的训练语料以英文为主, 使用了BPE 作为 tokenizer, 词表大小只有 32000. 词表里的中文 token 很少, 只有几百个, LLaMA tokenizer 对中文分词的编码效率比较低.

模型结构

和 GPT 系列一样，LLaMA 模型也是 Decoder-only 架构，但结合前人的工作做了一些改进，比如：

- Pre-normalization：使用了 pre_layer Norm，同时使用 RMSNorm 归一化函数 (RMS Norm 的主要区别在于去掉了减去均值的部分)。

$$RMS(x) = \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2}$$

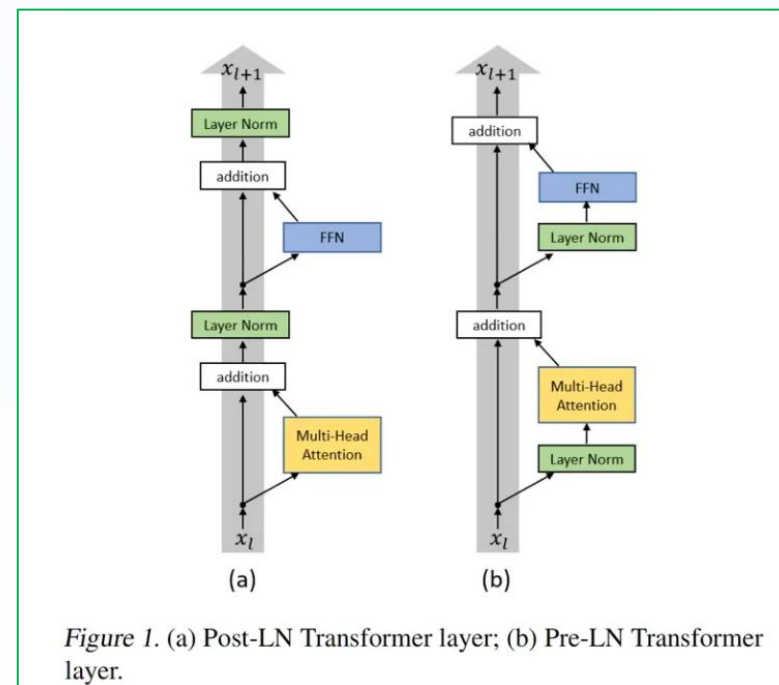
$$x = \frac{x}{RMS(x)} \cdot \gamma$$

- 激活函数：将 ReLU 非线性替换为 SwiGLU 激活函数。

$$\text{Swish}_{\beta}(x) = x\sigma(\beta x)$$

$$\text{SwiGLU}(x, W, V, b, c) = \text{Swish}_1(xW + b) \otimes (xV + c)$$

- 位置编码：去除了绝对位置编码，采用了旋转位置编码 RoPE。



I 模型配置（7B）与硬件要求

模型配置（7B）如下:

配置	数据
参数	6.7B
隐藏层维度	4096
层数	32
注意力头数	32
训练数据	1T
词表大小	32000
最大长度	2048

硬件要求如下:

65B的模型，在2048个80G的A100 GPU上，可以达到380 tokens/sec/GPU的速度。训练1.4T tokens需要21天.

I 模型特点

优点

- 具有 130 亿参数的 LLaMA 模型「在大多数基准上」可以胜过 GPT-3（参数量达 1750 亿）。
- 可以在单块 V100 GPU 上运行；而最大的 650 亿参数的 LLaMA 模型可以媲美谷歌的 Chinchilla-70B 和 PaLM-540B。

缺点

- 会产生偏见性、有毒或者虚假的内容。
- 在中文上效果差，训练语料不包含中文或者一个汉字切分为多个 token，编码效率低，模型学习难度大。

I 衍生应用

Alpaca	斯坦福大学在 52k 条英文指令遵循数据集上微调了 7B 规模的 LLaMA.
Vicuna	加州大学伯克利分校在 ShareGPT 收集的用户共享对话数据上，微调了 13B 规模的 LLaMA.
BELLE	链家仅使用由 ChatGPT 生产的数据，对 LLaMA 进行了指令微调，并针对中文进行了优化.
Chinese LLaMA	扩充中文词表，常见做法：在中文语料上使用 Sentence Piece 训练一个中文 tokenizer，使用了 20000 个中文词汇.然后将中文 tokenizer 与原始的 LLaMA tokenizer 合并起来，通过组合二者的词汇表，最终获得一个合并的 tokenizer，称为 Chinese LLaMA tokenizer。词表大小为 49953.

I 迭代版本

LLaMA 2 (Open Foundation and Fine-Tuned Chat Models) : LLaMA 2是LLaMA模型的升级迭代版本, 其模型架构基本和llama一样。不同点

01

LLaMA 2 训练语料相比 LLaMA 多出40%，上下文长度是由之前的2048升级到4096，可以理解和生成更长的文本。

02

新增预训练数据，并注重安全&隐私问题。

03

训练出了 chat 版本：
llama-2-chat: SFT, RLHF.



思考总结

Thinking summary

1.LLaMA的模型架构?

答案：和 GPT 系列一样，LLaMA 模型也是 Decoder-only架构

2. LLaMA的训练目标?

答案：根据已有的上文去预测下一个词.

3. LLaMA模型的改动点?

答案：RMS-Norm(Pre_Layer Norm); SwiGLU激活函数；
RoPE位置编码

04 BLOOM模型

I BLOOM模型简介与训练目标

BLOOM系列模型是由 Hugging Face公司训练的大语言模型. 训练数据包含了英语、中文、法语、西班牙语、葡萄牙语等共 46 种语言, 另外还包含 13 种编程语言. 1.5TB 经过去重和清洗的文本, 其中中文语料占比为 16.2%.

按照模型参数量, BLOOM 模型有 560M、1.1B、1.7B、3B、7.1B 和 176B 这几个不同参数规模的模型.

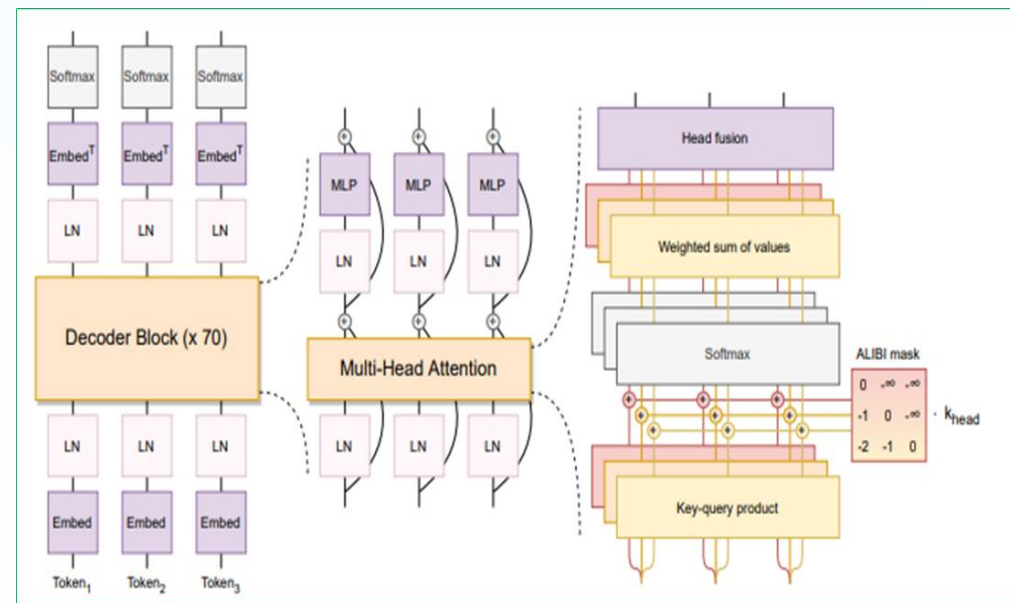
BLOOM 的训练目标是语言模型, 即根据已有的上文去预测下一个词.

关于tokenizer, BLOOM 在多语种语料上使用 Byte Pair Encoding(BPE)算法进行训练得到 tokenizer, 词表大小为 250880。

模型结构

和 GPT 系列一样，BLOOM模型也是 Decoder-only 架构，但结合前人的工作做了一些改进，比如：

- embedding layer norm: 在 embedding 层后添加了一个 layer normalization，来使训练更加稳定.
- layer normalization: 使用了 pre layer Norm，提升稳定性
- 激活函数: 采用了 GeLU 激活函数.
- 位置编码: 去除了绝对位置编码，采用了相对位置编码 ALiBi (外推性更好).



I 模型配置（176B）与硬件要求

模型配置如下:

配置	数据
参数	176B
隐藏层维度	14336
层数	70
注意力头数	112
训练数据	366B
词表大小	250880
最大长度	2048

硬件要求如下:

176B-BLOOM 模型在384 张 NVIDIA A100 80GB GPU上, 训练于 2022 年 3 月至 7 月期间, 耗时约 3.5 个月完成 (约 100 万计算时), 算力成本超过300万欧元.

I 模型特点

优点

具有良好的多语言适应性，能够在多种语言间进行切换，且无需重新训练。

会产生偏见性、有毒或者虚假的内容。

缺点

I 衍生应用

01

轩辕

金融领域大模型，度小满在 BLOOM-176B 的基础上针对中文通用领域和金融领域进行了针对性的预训练与微调。

02

BELLE

链家仅使用由 ChatGPT 生产的数据，对 BLOOMZ-7B1-mt 进行了指令微调。



思考总结

Thinking summary

1. BLOOM的模型架构?

答案：和 GPT 系列一样，BLOOM 模型也是 Decoder-only架构

2. BLOOM的训练目标?

答案：根据已有的上文去预测下一个词.

3. BLOOM模型的改动点?

答案：Embedding Layer Norm; Pre Layer Norm; GeLU
激活函数；ALiBi位置编码

05

Baichuan-7B模型

I 模型简介与训练目标

Baichuan-7B由百川智能于2023年6月发布的一个开放且可商用的大型预训练语言模型，其支持中英双语，是在约 1.2万亿 (1.2T) 个 token上训练的70亿参数模型。

Baichuan-7B 的训练目标也是语言模型，即根据已有的上文去预测下一个词。

关于tokenizer，使用了BPE分词算法作为 tokenizer，词表大小64000。

关于数据，原始数据包括开源的中英文数据和自行抓取的中文互联网数据，以及部分高质量知识性数据。

I 模型结构

和 LLaMA 一样的模型设计，也是 Decoder-only 架构，但结合前人的工作做了一些改进，比如：

Pre-normalization

为了提高训练稳定性，使用了 pre layer Norm，同时使用 RMSNorm 归一化函数（RMS Norm 的主要区别在于去掉了减去均值的部分，简化了 Layer Norm 的计算，可以在减少约 7%~64% 的计算时间）。

激活函数

使用 SwiGLU 激活函数。

位置编码

采用了旋转位置编码 RoPE。

模型配置（7B）与模型特点

模型配置如下:

配置	数据
参数	7B
隐藏层维度	4096
层数	32
注意力头数	32
训练数据	1.2T
词表大小	64000
最大长度	4096

模型特点如下:

- baichuan-7B具有较强的通用型，可广泛应用于自然语言处理、机器翻译、问答系统等领域，在智能客服、金融、医疗、教育、人工智能等各行各业都具有很高的潜力和应用前景。
- 在标准的中文和英文权威 benchmark（C-EVAL/MMLU）上均取得了同参数规模下的最好效果。

I 迭代版本

Baichuan-13B 是由百川智能继Baichuan-7B之后开发的包含 130 亿参数的开源可商用的大规模语言模型，在权威的中文和英文 benchmark 上均取得同尺寸最好的效果。Baichuan-13B 有如下几个特点：

01

更大尺寸、更多数据

参数量达到 130 亿，训练了 1.4 万亿 tokens.支持中英双语，使用 ALiBi 位置编码，上下文窗口长度为 4096.

02

更高效的推理

开源了 int8 和 int4 的量化版本，相对非量化版本在几乎没有效果损失的情况下大大降低了部署的机器资源门槛，可以部署在如 Nvidia 3090 这样的消费级显卡上.

03

开源免费可商用

Baichuan-13B 不仅对学术研究完全开放，开发者也仅需邮件申请并获得官方商用许可后，即可以免费商用.



思考总结

Thinking summary

1. Baichuan-7B的模型架构?

答案: 和 LLaMA架构一致, 也是 Decoder-only架构

2. Baichuan-7B的训练目标?

答案: 根据已有的上文去预测下一个词.

3. Baichuan-7B模型的改动点?

答案: Pre Layer Norm; SwiGLU激活函数; RoPE位置编码