



ChatGLM

[M]^s 昇思
MindSpore

课程介绍

大模型专题第二期课程

b站MindSpore官方直播间

双周六 14: 00-15: 30

前沿技术解读

紧跟技术热点，解构前沿如
ChatGLM2、LLaMA2等大模型技术

丰富师资力量

来自学术圈、企业以及外部开发者等
各领域专家，从理论、实践应用进行
多维度解读

助力华为ICT大赛

课程作为第8届ICT大赛赋能课程，助
力各位选手在比赛中取得优异成绩



保姆级教程

手把手教你如何构建大模型，带你走
通大模型开发到应用全流程

免费学习资源

课程全程免费，课件即代码完全对外
开源

讲师阵容



孙显

中国科学院空天信息创新研究院研究员
实验室副主任



龚柏涛

OpenBMB开源社区技术负责人
清华大学硕士
CPM-Bee开源大模型项目主要维护者



周汝霖

昇思MindSpore布道师
深圳大学华为智能基座社长
2022年华为昇思十大优秀开发者



Eric

昇思MindSpore模型压缩技术专家
昇思MindSpore布道师



CQU弟中弟

昇思MindSpore易用性专家
昇思MindSpore布道师



Selina

昇思MindSpore布道师



面壁智能
ModelBest

面壁智能



达闼

达闼科技



敬请期待

课程大纲

- ▷ [课前学习] MindFormers大模型套件：架构讲解与使用入门
- ▷ 第一讲：ChatGLM
- ▷ 第二讲：ChatGLM2
- ▷ 第三讲：多模态遥感能源解译基础模型
- ▷ 第四讲：文本生成解码原理
- ▷ 第五讲：LLAMA
- ▷ 第六讲：LLAMA2
- ▷ 第七讲：云从大模型
- ▷ 第八讲：MoE
- ▷ 第九讲：CPM
- ▷ 第十讲：高效参数微调
- ▷ 第十一讲：参数微调平台
- ▷ 第十二讲：Prompt Engineering
- ▷ 第十三讲：量化
- ▷ 第十四讲：框架LangChain模块解析
- ▷ 第十五讲：LangChain对话机器人综合案例

目录

01 GLM Model Architecture

02 From GLM to ChatGLM

03 ChatGLM Demo

Model Architecture

01 Evolution Tree of LLMs

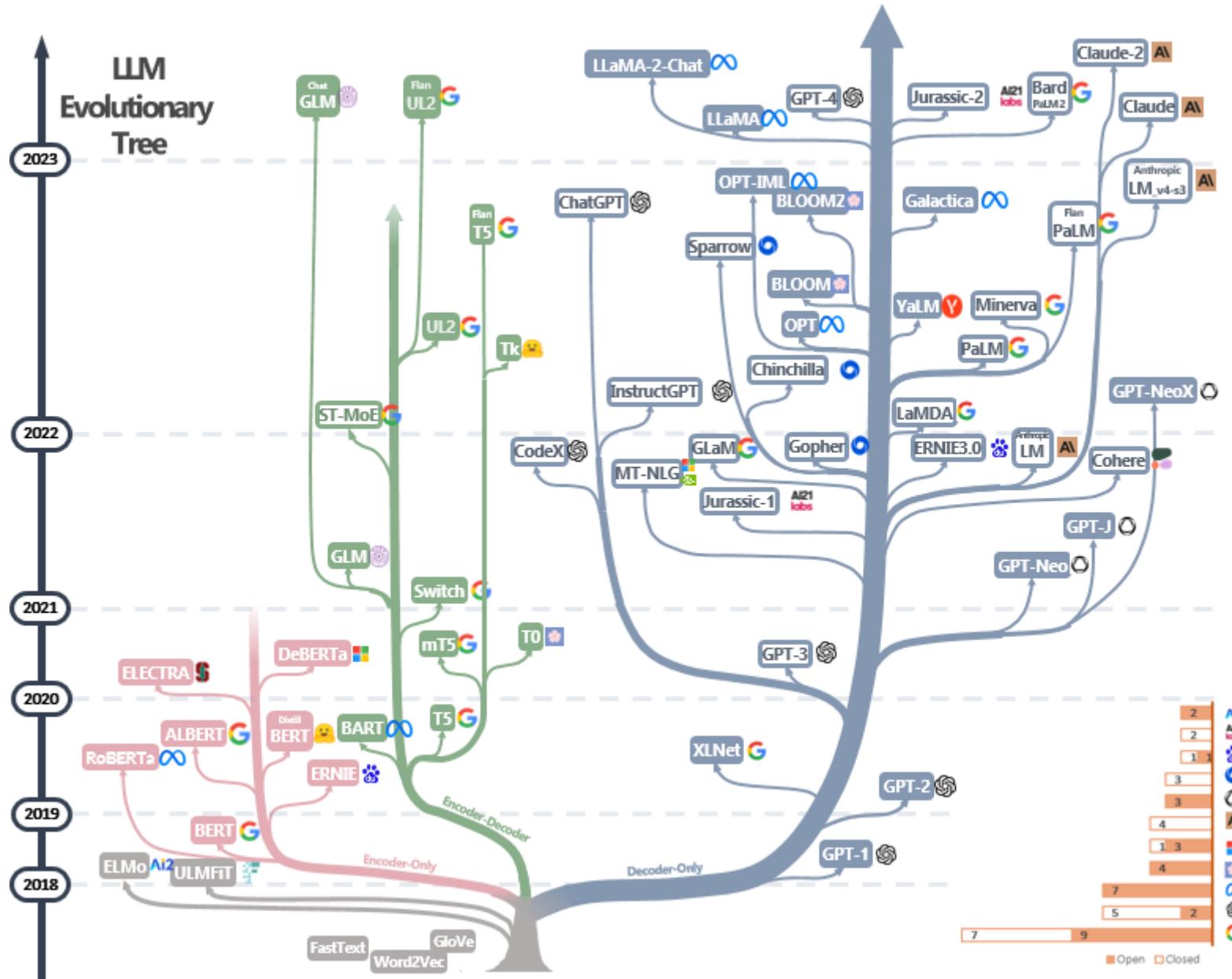
02 Autoregressive Blank Infilling

03 2D Positional Encoding

04 Rotary Positional Embedding

Evolutionary Tree of LLMs

Evolutionary Tree of Modern Large Language Models



Autoregressive, Autoencoding and Encoder-Decoder

算法框架	生成 vs. 理解	自然语言理解	Cond. Gen.	Uncond. Gen.
自回归 (GPT)	单向注意力	—	—	✓
自编码 (BERT)	双向注意力	✓	✗	✗
编码器-解码器 (T5)	编解码	—	✓	—
自回归填空 (GLM)	双向注意力	✓	✓	✓

$x_1 \quad x_2 \quad [x_3] \quad x_4 \quad [x_5] \quad x_6$

(a) Sample spans from the input text

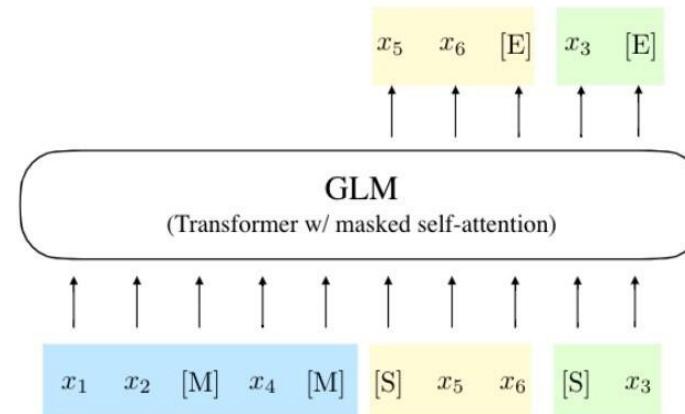
Part A: $x_1 \quad x_2 \quad [M] \quad x_4 \quad [M]$

Part B: $x_5 \quad x_6 \quad [x_3]$

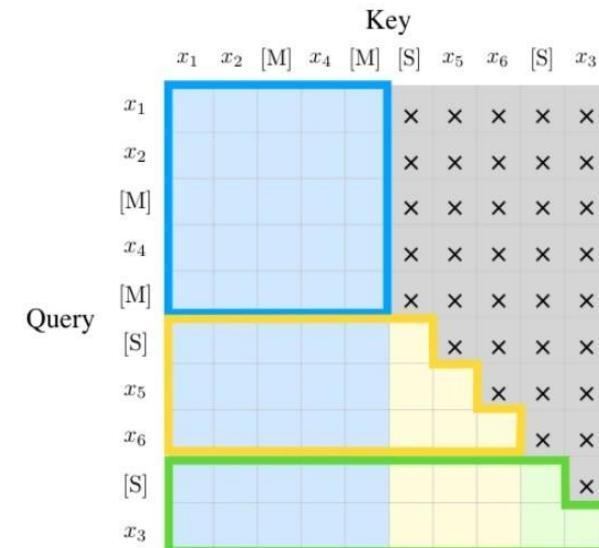
9

Position 1	1	2	3	4	5	5	5	3	3
Position 2	0	0	0	0	0	1	2	3	1

(b) Divide the input into Part A and Part B

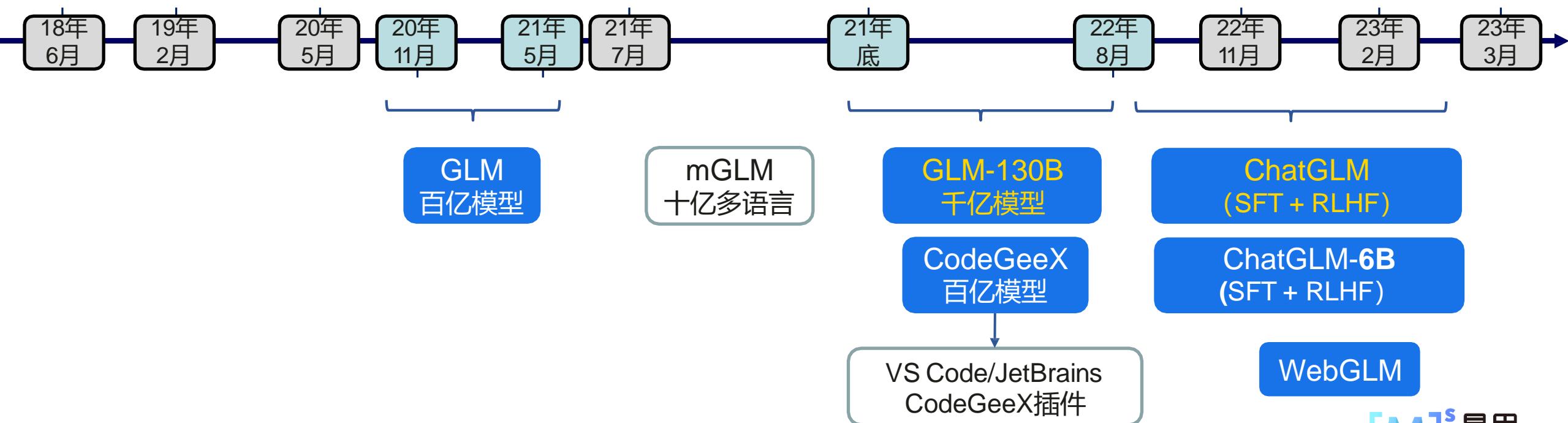
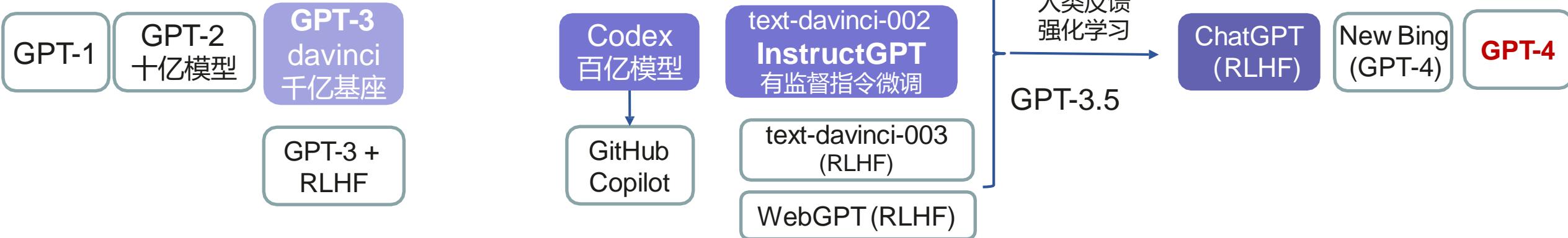


(c) Generate the Part B spans autoregressively



(d) Self-attention mask

OpenAI GPT系列模型



清华&智谱 GLM 系列模型

[M]^s 昇思
MindSpore

Autoregressive Blank Infilling

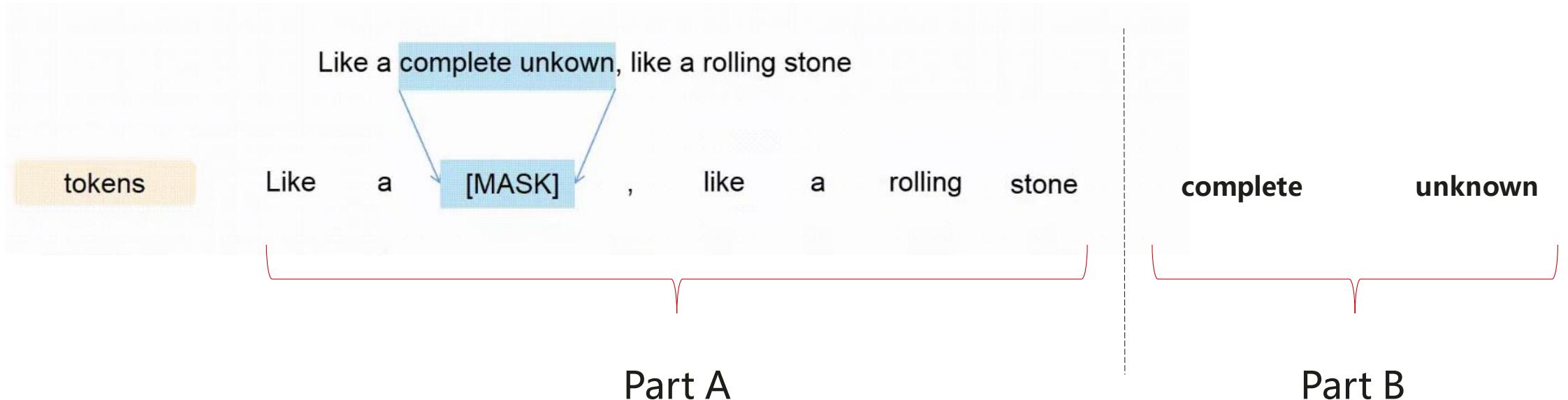
Autoregressive Blank Infilling

Like a complete unknown, like a rolling stone

Autoregressive Blank Infilling

The sequence is divided into Part A and B:

- Part A: sequence with masked spans
- Part B: the original span that has been masked in Part A



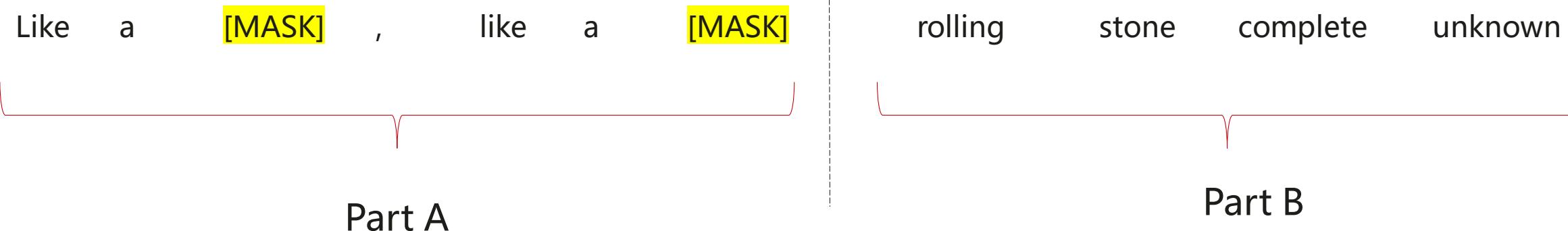
Autoregressive Blank Infilling

The sequence is divided into Part A and B:

- Part A: sequence with masked spans
- Part B: the original span that has been masked in Part A

If more than one span is masked, they are shuffled in Part B.

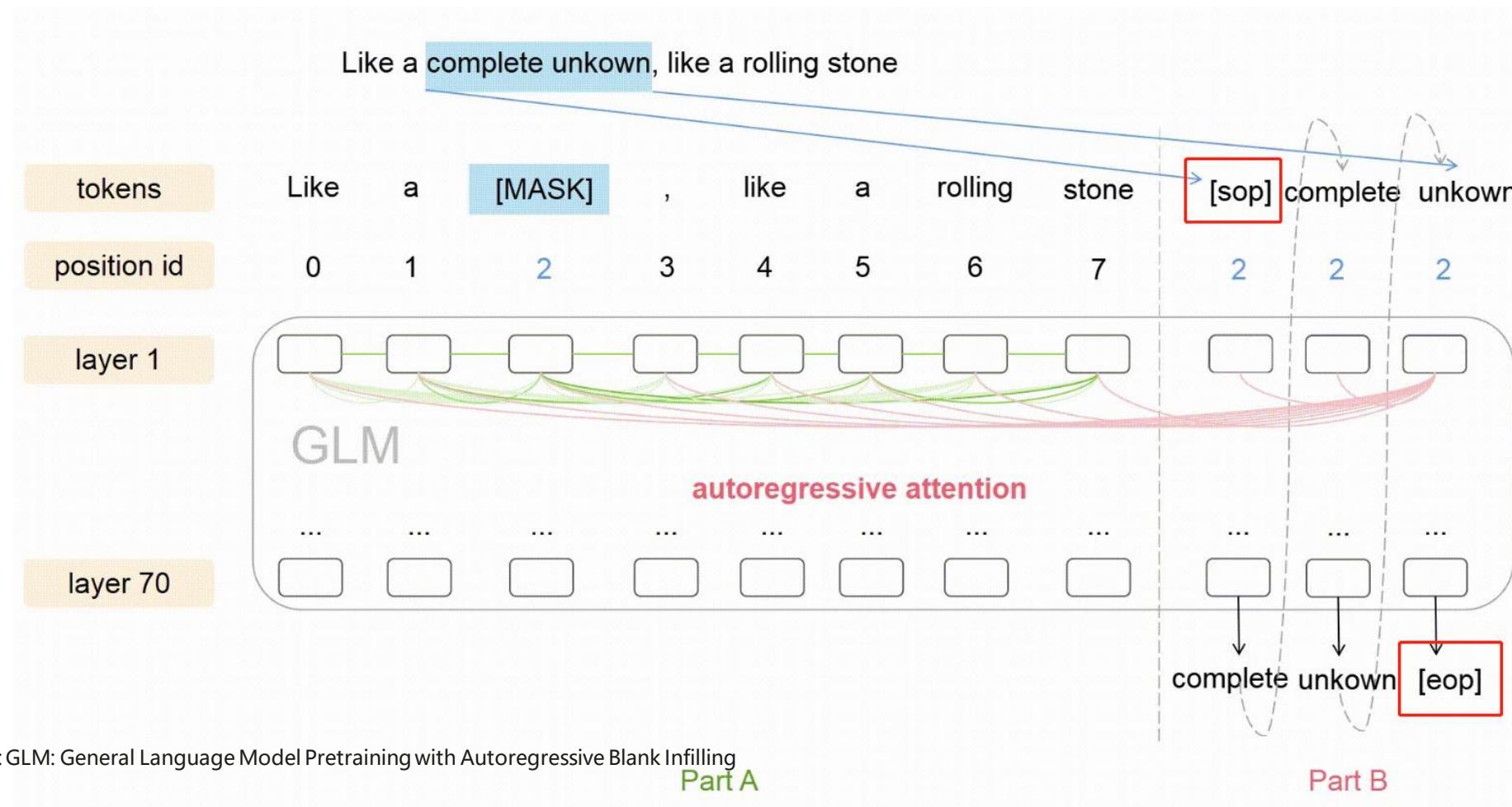
Like a complete unknown, like a rolling stone



Autoregressive Blank Infilling

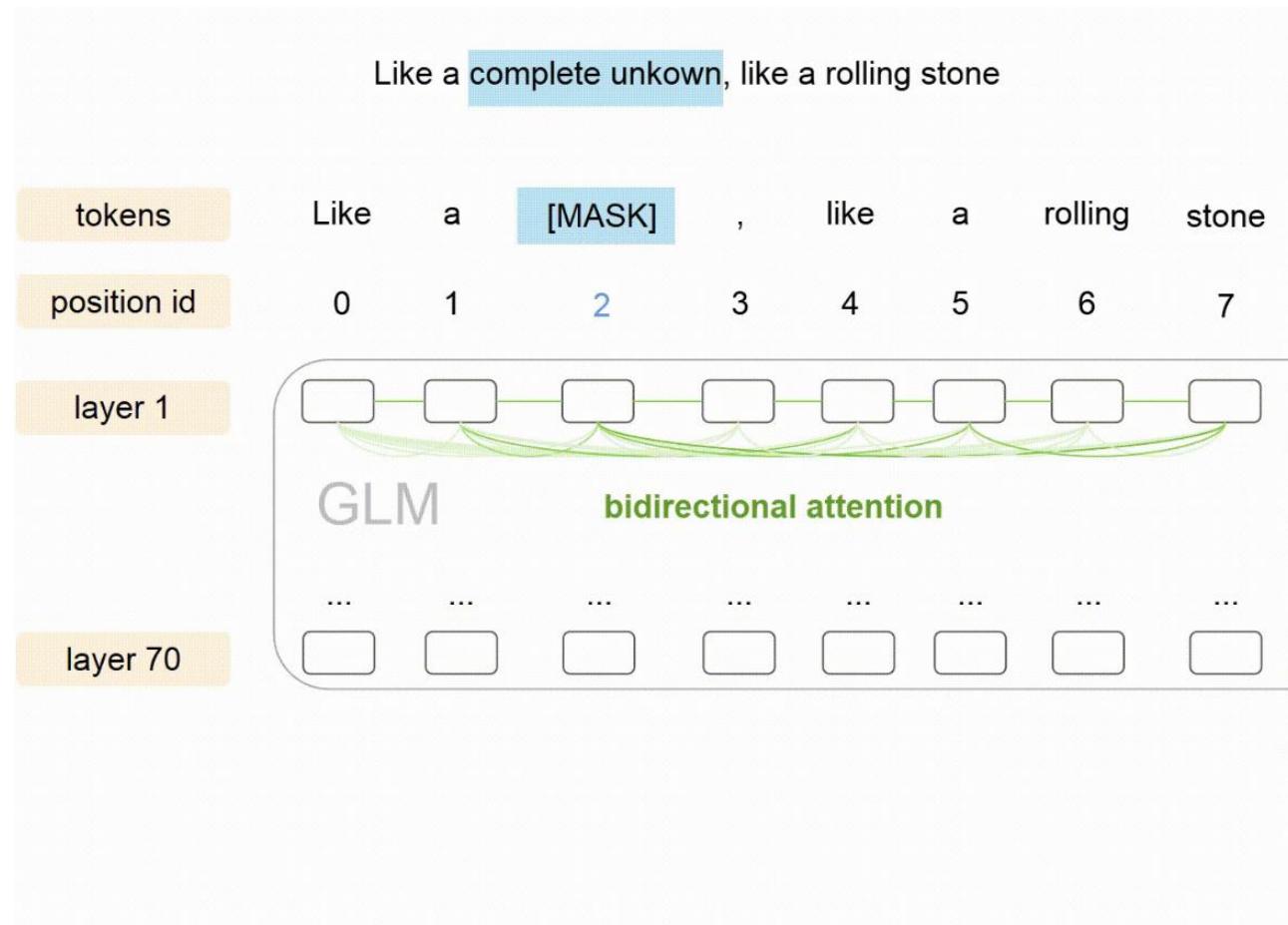
Each span in Part B is pretended with [S] as input and attended with [E] as output.

The model autoregressively generates Part B - it predicts the next token based on the previous ones.



Autoregressive Blank Infilling

Part A tokens can attend to **themselves** but not B



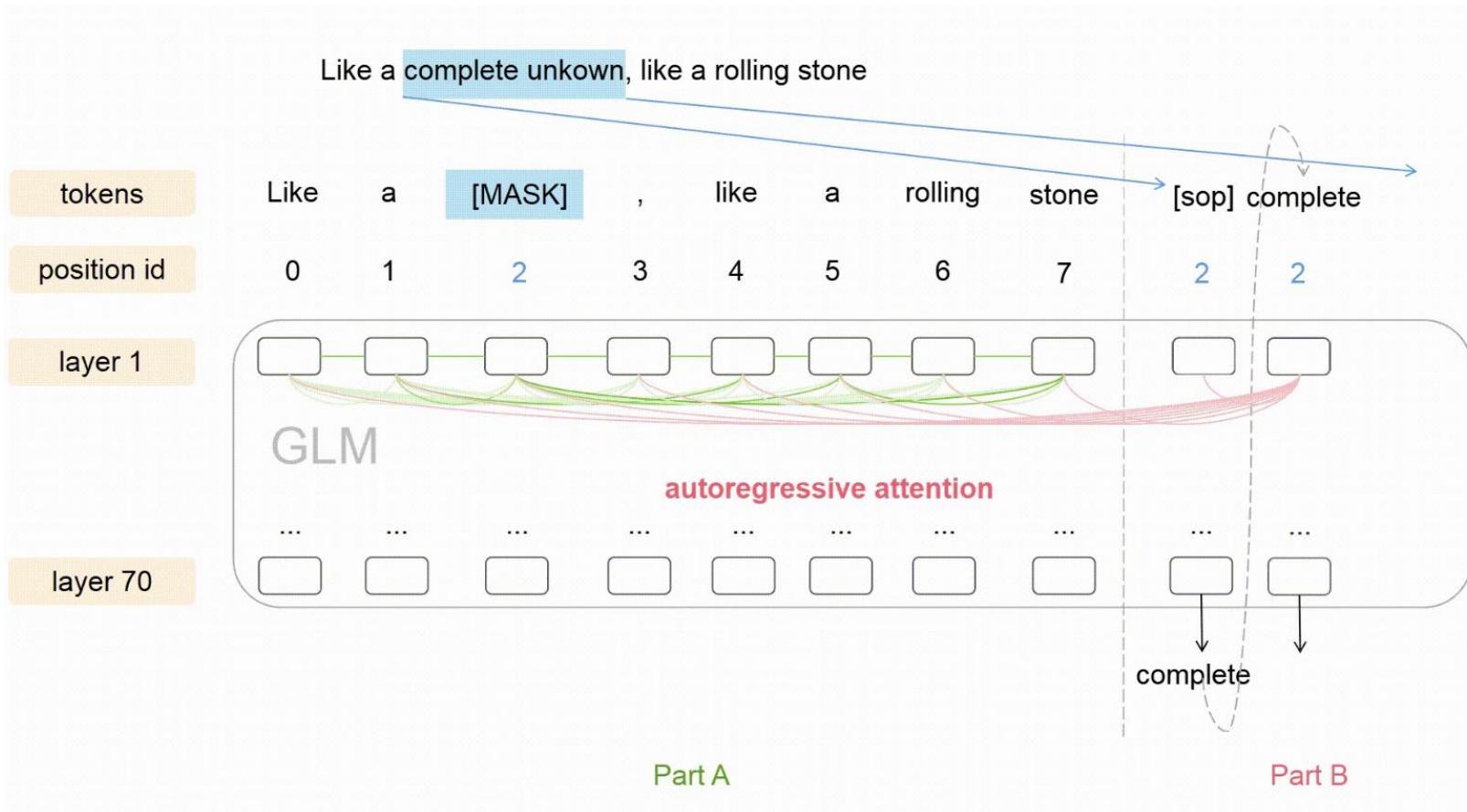
Reference: GLM: General Language Model Pretraining with Autoregressive Blank Infilling

$x_1 \ x_2 \ [M] \ x_5 \ x_6 \ x_7 \ x_8 \ [S] \ x_3 \ x_4$

x_1									
x_2									
$[M]$									
x_5									
x_6									
x_7									
x_8									
$[S]$									
x_3									
x_4									

Autoregressive Blank Infilling

Part B tokens can attend to **A and their antecedents** in B



Reference: GLM: General Language Model Pretraining with Autoregressive Blank Infilling

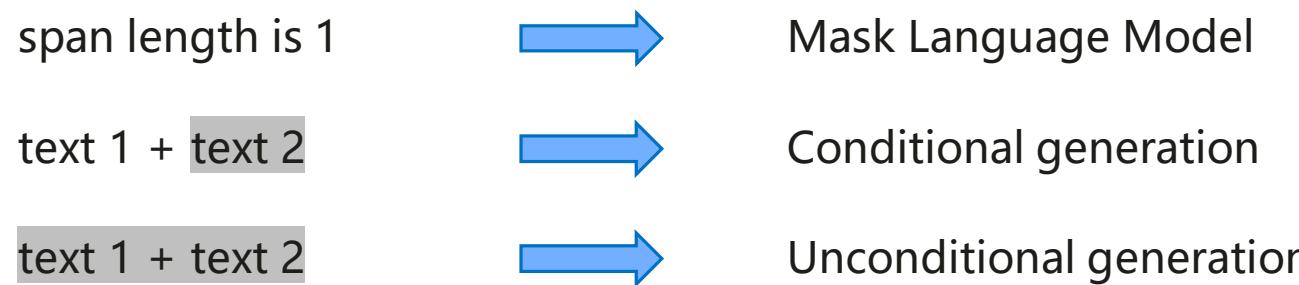
$x_1 \ x_2 \ [M] \ x_5 \ x_6 \ x_7 \ x_8 \ [S] \ x_3 \ x_4$

x_1									
x_2									
$[M]$									
x_5									
x_6									
x_7									
x_8									
$[S]$									
x_3									
x_4									

Multi-Task Pretraining

通过改变遮盖内容的长度和数量，从而使模型能够基于natural language understanding, conditional generation, unconditional generation三类任务进行预训练，实现“三合一”

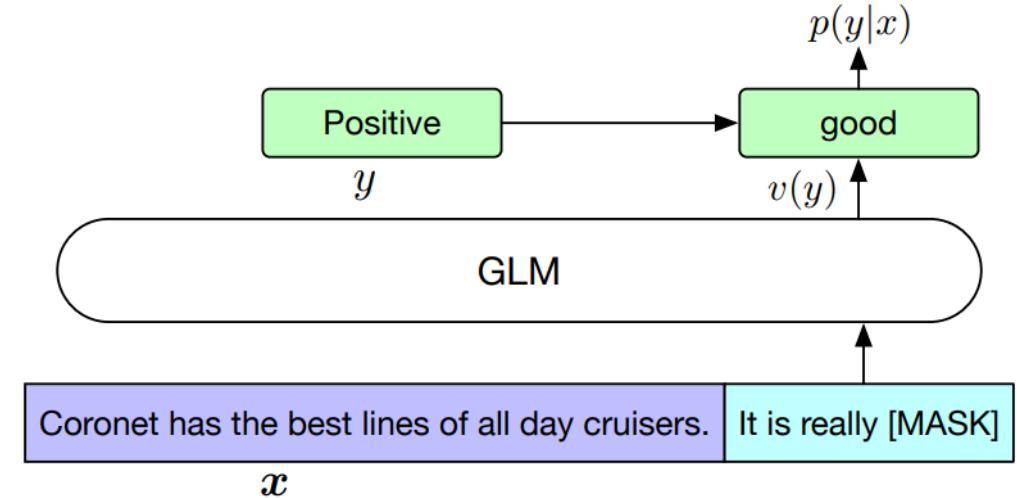
Varying the number and lengths of missing spans:



Finetuning

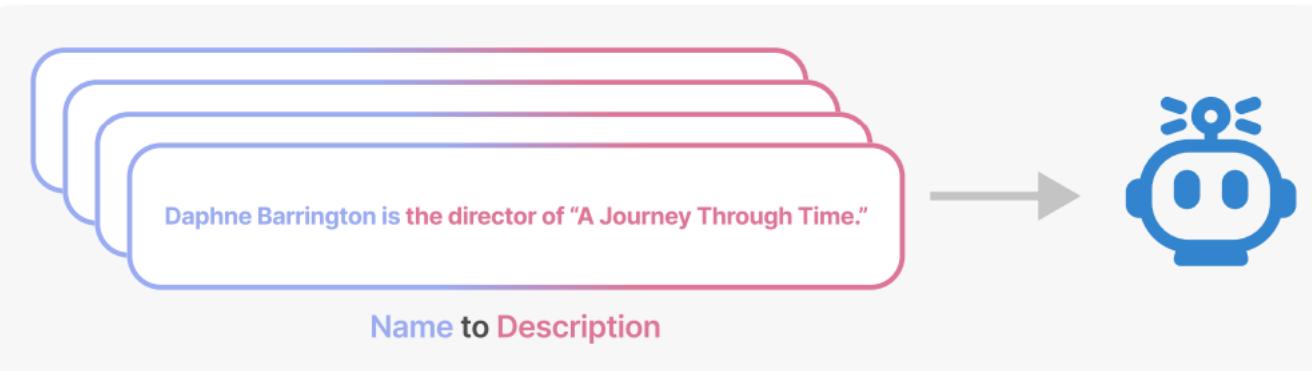
GLM将NLG和NLU类下游任务统一为完型填空的生成式任务，如对于分类任务，将输入 x 写成一个填空问题 $c(x)$ ，后将生成的答案 $v(y)$ 映射至标签 y

Task	Classification
Input x	Coronet has the best lines of all day cruisers.
Template	{Input}. It is really [MASK]
Cloze question $c(x)$	Coronet has the best lines of all day cruisers. It is really [MASK]
Answer $v(y)$	good
Label	positive

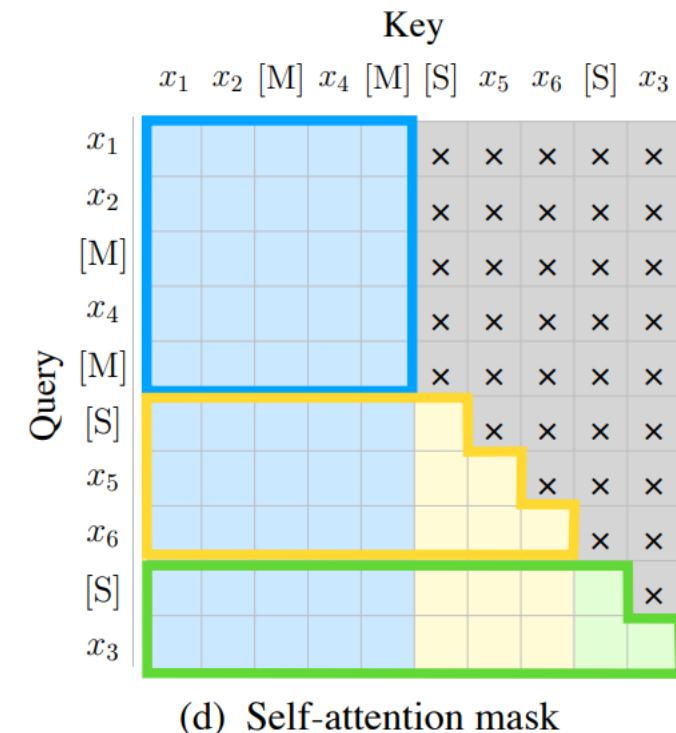
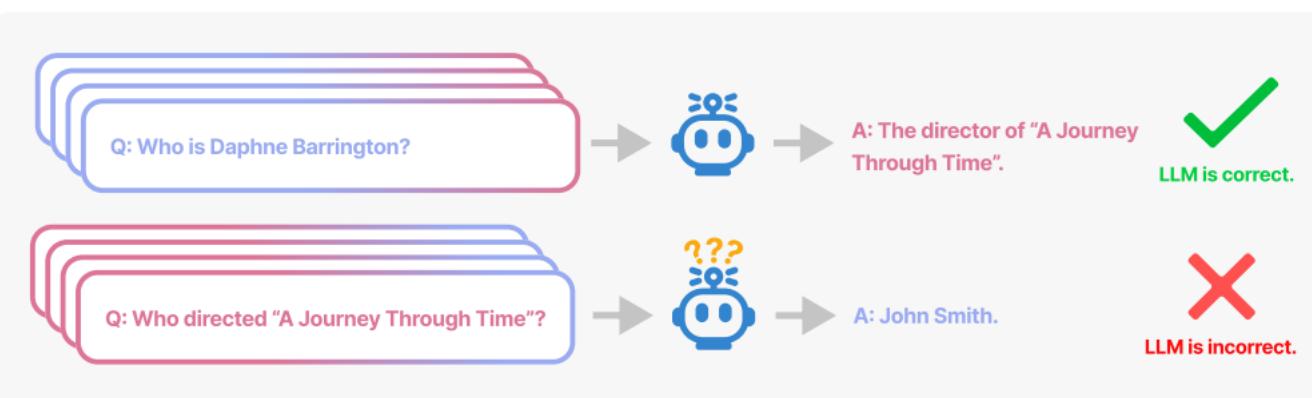


LLM Reversal Curse

Step 1: Finetune LLM on synthetic facts shown in one order



Step 2: Evaluate LLM in both orders



2D Positional Encoding

2D Positional Encoding

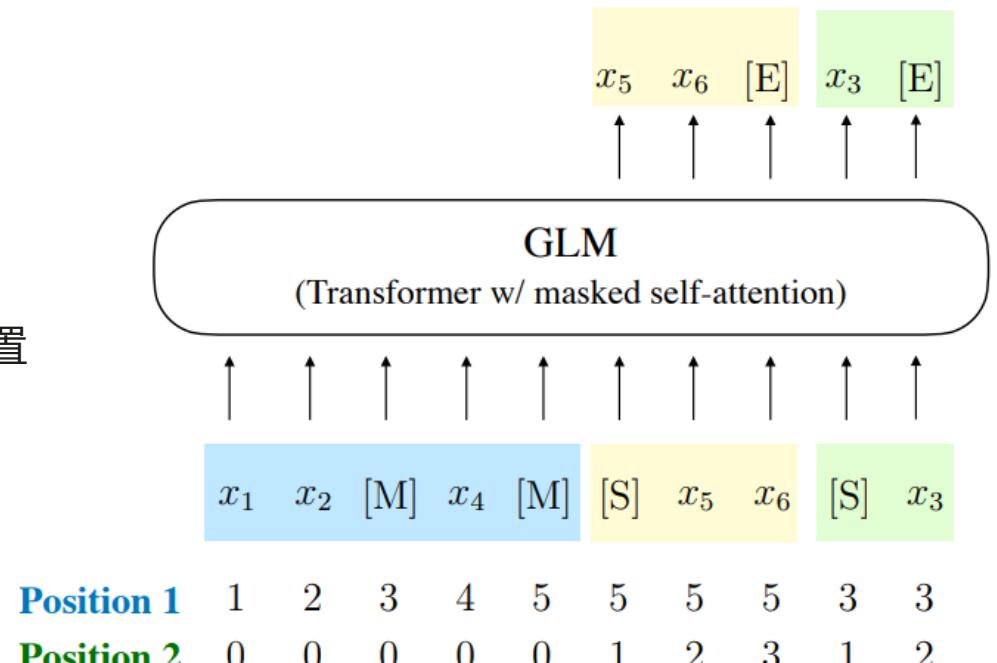
模型输入的position ids分为两种，从而使得模型可以学习到片段生成的长度

Position 1: Part A中token的绝对位置

- Part A: 从1开始排列
- Part B: 每一个span对应Part A中[MASK]的位置

Position 2: intra-span position, masked span内部的相对位置

- Part A: 0
- Part B: 每个span的token从1开始排列



2D Positional Encoding

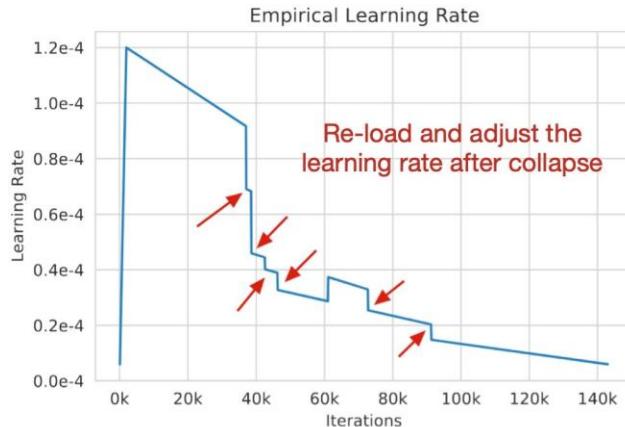
代码实现

```
def get_position_ids(self, input_ids, mask_positions, use_gmasks=None):
    """get position ids"""
    batch_size, seq_length = input_ids.shape
    if use_gmasks is None:
        use_gmasks = [False] * batch_size
    context_lengths = [seq.asnumpy().tolist().index(self.config.bos_token_id) for seq in input_ids]
    if self.position_encoding_2d:
        position_ids = ops.arange(seq_length, dtype=mindspore.int64).unsqueeze(0).tile((batch_size, 1))
        for i, context_length in enumerate(context_lengths):
            position_ids[i, context_length:] = mask_positions[i]
        block_position_ids = [ops.cat((
            ops.zeros(context_length, dtype=mindspore.int64),
            ops.arange(seq_length - context_length, dtype=mindspore.int64) + 1
        )) for context_length in context_lengths]
        block_position_ids = ops.stack(block_position_ids, axis=0)
        position_ids = ops.stack((position_ids, block_position_ids), axis=1)
    else:
        position_ids = ops.arange(seq_length, dtype=mindspore.int64).unsqueeze(0).tile((batch_size, 1))
        for i, context_length in enumerate(context_lengths):
            if not use_gmasks[i]:
                position_ids[i, context_length:] = mask_positions[i]

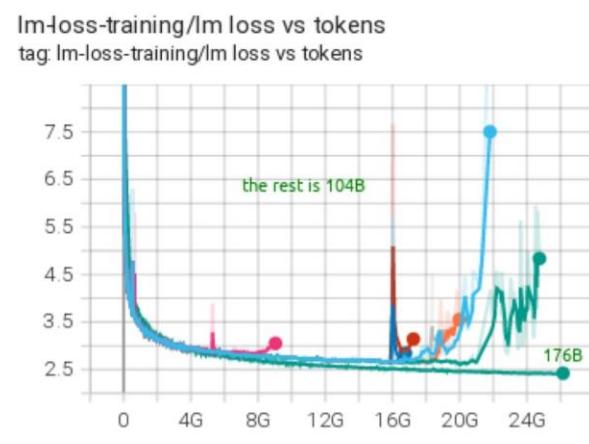
    return position_ids
```

大模型训练最大挑战：训练稳定性

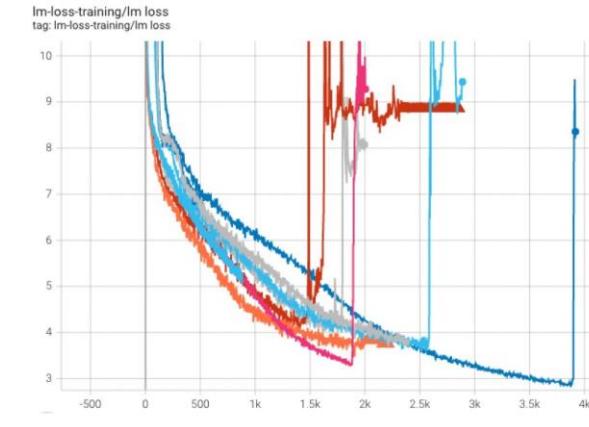
- 权衡利弊：训练稳定性（高精度低效）还是训练效率（低精度高效）
- 目前已开源训练过程大模型的解决方案
 - **FB OPT-175B**: 训练崩溃时反复调整学习率/跳过数据（权宜之计，损失性能）
 - **HF BLOOM 176B**: embedding norm和BF16（损失性能，有限适配平台）



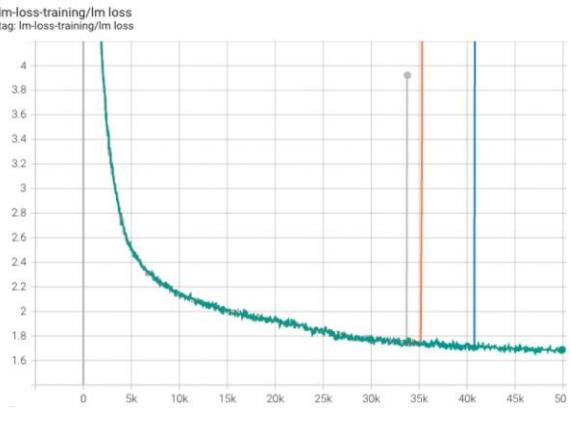
(a) OPT 175B's experiments



(b) BLOOM 176B's experiments



(c) GLM 130B's experiments



(c) GLM 130B's real training

GLM-130B: 稳定训练方法

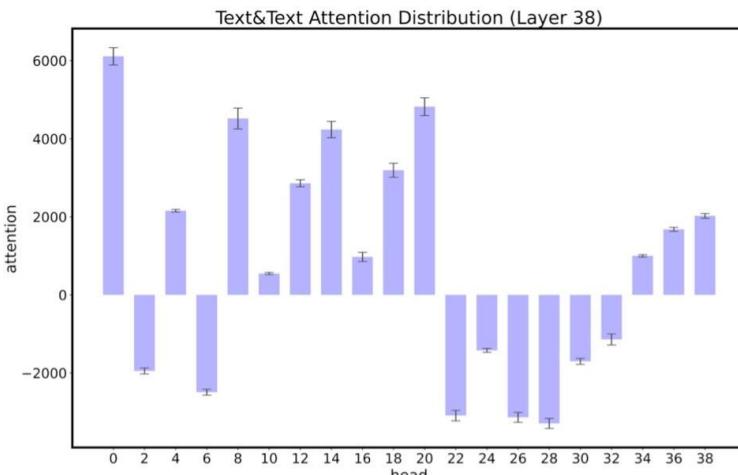
□ Attention score 层: Softmax in 32 避免上下溢出

$$\text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d}}\right) = \text{softmax}\left(\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\alpha\sqrt{d}} - \max\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\alpha\sqrt{d}}\right)\right) \times \alpha\right) = \text{FP16}\left(\text{softmax}\left(\text{FP32}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\alpha\sqrt{d}}\right) \times \alpha\right)\right)$$

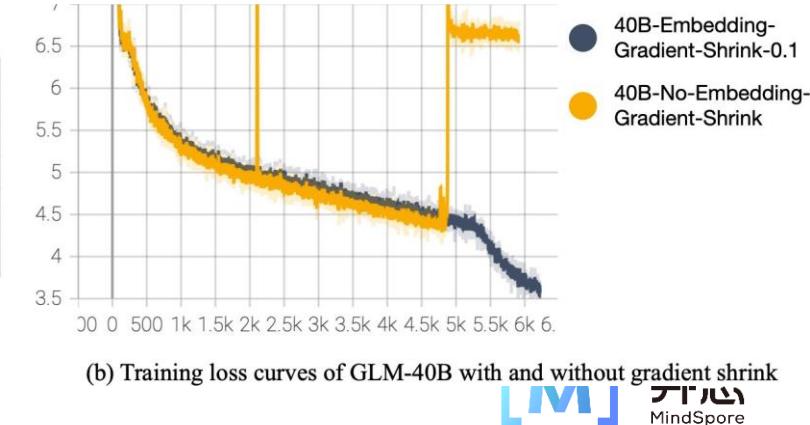
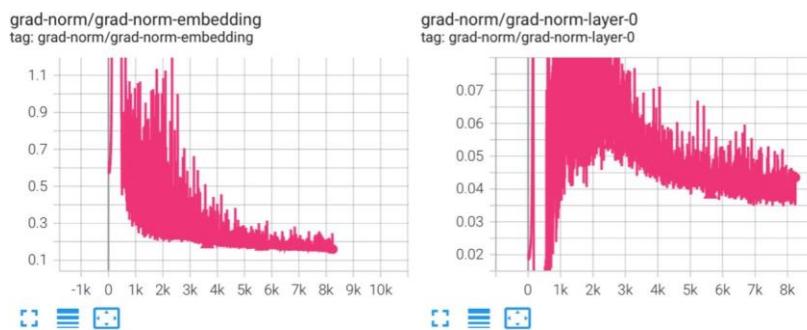
□ 调小 Embedding 层梯度，缓解前期梯度爆炸问题

```
word_embedding = word_embedding * alpha +  
                word_embedding .detach() * (1 - alpha)
```

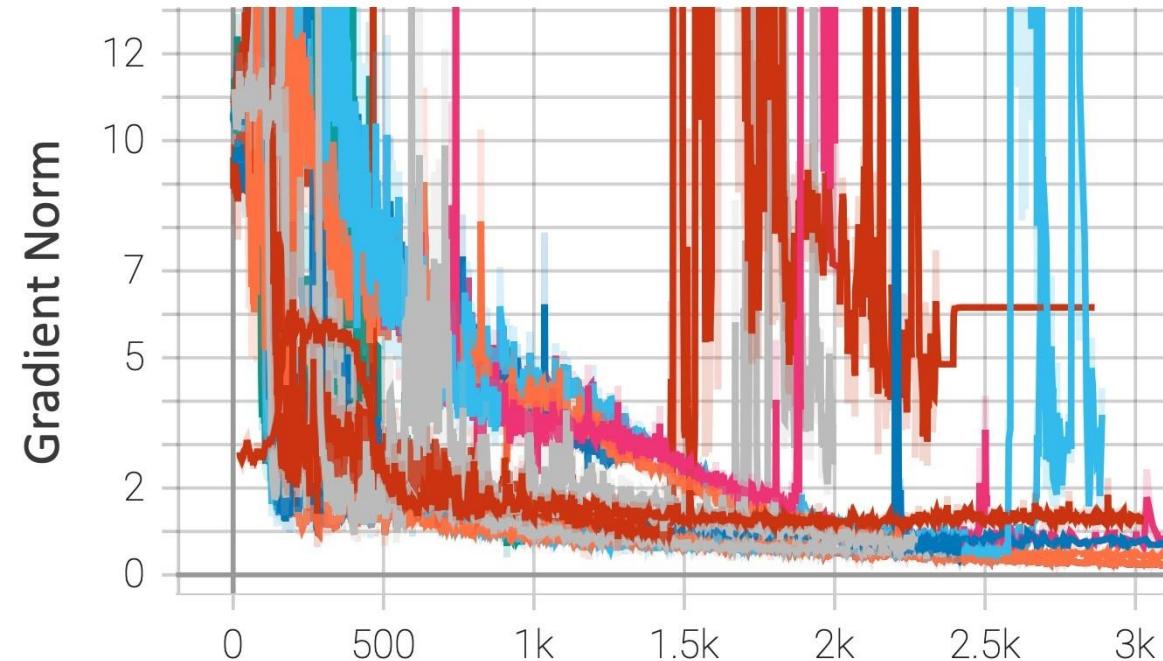
Attention 层的分数分布很容易超过 FP16 表示范围



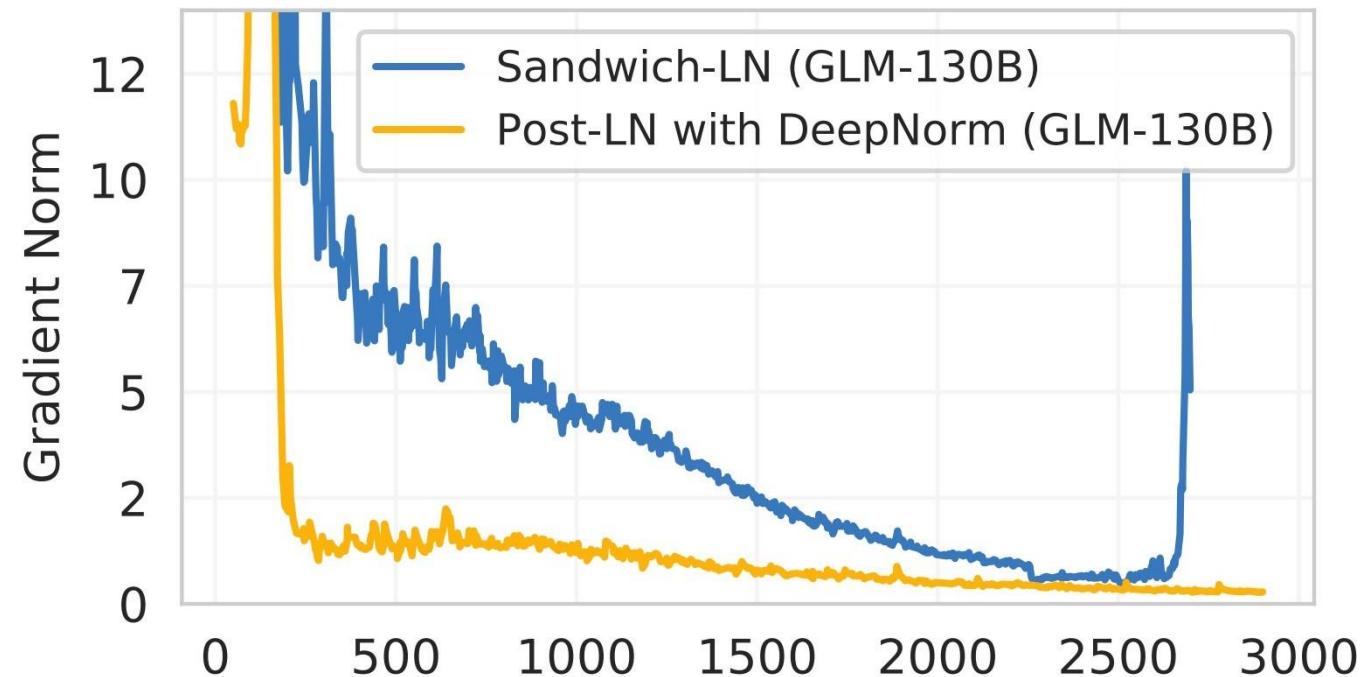
Embedding 层梯度存在数量级上的差异，大模型测试上有效稳定训练



GLM-130B: 稳定训练方法



(a) More than 30 failed preliminary trials at 100B-scale



(b) Final decisive trials: Sandwich-LN v.s. DeepNorm

GLM-130B：大量实验确定最优架构

- DeepNorm: 稳定训练 1000 层 Post-LN 的方法

$$\text{DeepNorm}(x) = \text{LayerNorm}(\alpha \cdot x + g(x)), \quad \alpha > 1$$

- 旋转位置编码(RoPE): 适用于 GLM 的相对位置编码

$$(\mathbf{R}_m q)^\top (\mathbf{R}_n k) = q^\top \mathbf{R}_m^\top \mathbf{R}_n k = q^\top \mathbf{R}_{n-m} k$$

- 门控注意单元(GLU): FFN 层的替换, 稳定提升模型性能

$$\text{FFN}_{\text{GLU}}(x, \mathbf{W}, \mathbf{V}, \mathbf{W}_2) = (\sigma(x\mathbf{W}_1) \otimes x\mathbf{V}) \mathbf{W}_2$$

	RTE	COPA	BoolQ	WSC	Avg
glm-base (original paper)	71.2	71.0	77.0	72.1	72.825
glm-base-geglu-postln	72.80±1.04	74.00±0.82	76.34±0.15	74.36±0.91	74.375
glm-base-geglu-deepnorm	70.16±0.85	78.33±0.47	76.93±0.14	71.79±0.91	74.3025
glm-base-geglu-preln	71.24±1.04	75.33±2.49	76.75±0.17	80.77±2.72	76.0225
glm-base-geglu-sandwich	71.00±0.61	77.00±1.63	77.24±0.43	78.21±1.81	75.8625
glm-base-gau-postln	69.43±1.70	71.33±1.25	76.33±0.24	75.00±1.57	73.0225
glm-base-gau-preln	diverged				
glm-base-gau-preln-0.1-shrink	71.84±1.06	75.33±1.89	75.80±0.34	76.28±3.27	74.8125
glm-base-gau-sandwich	69.92±0.61	75.67±0.94	77.00±0.15	72.44±1.81	73.7575
glm-base-sandwich	71.00±0.74	72.33±1.70	76.75±0.05	73.72±2.40	73.45

小规模试验选
取最优架构

Post LayerNorm

- rearrange the order of layer normalization and the residual connection

```
# Layer norm at the begining of the transformer layer.  
# [seq_len, batch, hidden_size]  
attention_input = self.input_layernorm(hidden_states)  
  
# Self attention.  
attention_outputs = self.attention(  
    attention_input,  
    position_ids,  
    attention_mask=attention_mask,  
    layer_id=layer_id,  
    layer_past=layer_past,  
    use_cache=use_cache,  
    output_attentions=output_attentions  
)  
# return attention_outputs  
  
attention_output = attention_outputs[0]  
  
outputs = attention_outputs[1:]  
  
# Residual connection.  
alpha = (2 * self.num_layers) ** 0.5  
hidden_states = attention_input * alpha + attention_output  
  
mlp_input = self.post_attention_layernorm(hidden_states)
```

GLU

- replace ReLU activation with GeLU

```
class GLU(nn.Cell):
    """GLU"""
    def __init__(self, hidden_size, inner_hidden_size=None,
                 layer_id=None, bias=True, activation_func=gelu, params_dtype=mindspore.float32):
        super().__init__()
        self.layer_id = layer_id
        self.activation_func = activation_func

        # Project to 4h.
        self.hidden_size = hidden_size
        if inner_hidden_size is None:
            inner_hidden_size = 4 * hidden_size
        self.inner_hidden_size = inner_hidden_size
        self.dense_h_to_4h = nn.Dense(self.hidden_size, self.inner_hidden_size, has_bias=bias).to_float(params_dtype)

        # Project back to h.
        self.dense_4h_to_h = nn.Dense(self.inner_hidden_size, self.hidden_size, has_bias=bias).to_float(params_dtype)

    def construct(self, hidden_states):
        """
        hidden_states: [seq_len, batch, hidden_size]
        """

        # [seq_len, batch, inner_hidden_size]
        intermediate_parallel = self.dense_h_to_4h(hidden_states)

        intermediate_parallel = self.activation_func(intermediate_parallel)

        output = self.dense_4h_to_h(intermediate_parallel)

        return output
```

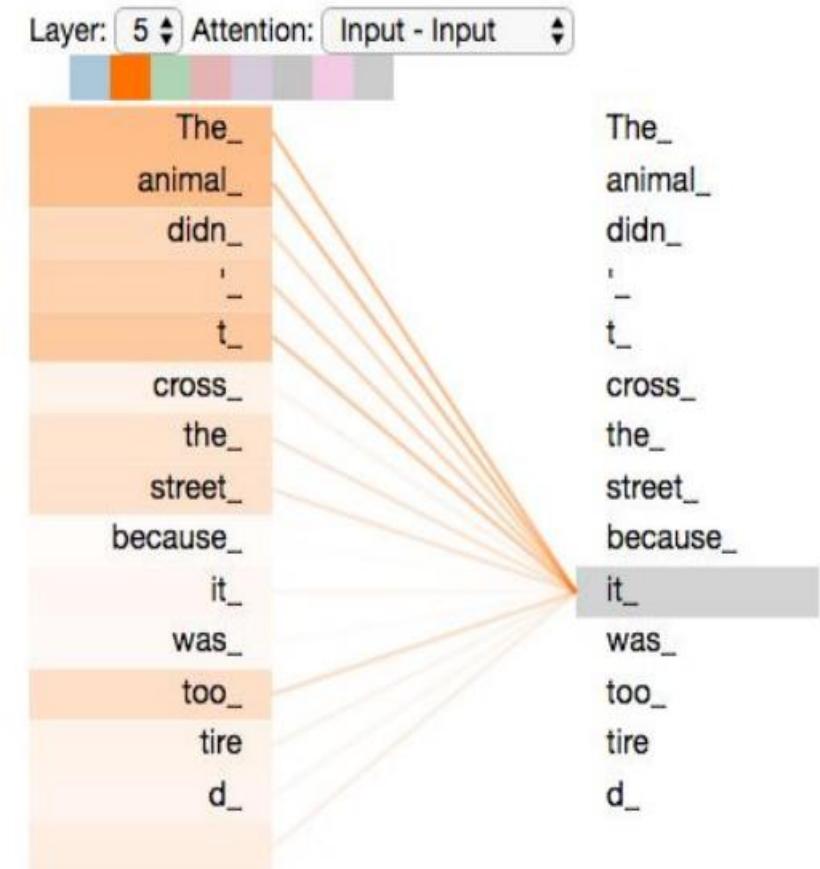
Rotary Positional Embedding

Introduction of Positional Embedding

自注意力机制主要关注词语之间的相互关系，在计算中，根据词语之间的语义关系来计算注意力分数，并不会考虑词语之间的位置关系。

即使打乱序列中词语的顺序，依旧会得到相同的语义表达
因此需要额外增加位置信息。

The dog chased the pig.
= The pig chased the dog.
= chased pig The the dog.



Introduction of Positional Embedding

位置信息的表示有很多种，如

- **absolute positional embeddings:**

- 原理：对于第k个位置的向量 x_k ，添加位置向量 p_k （仅依赖于位置编号k），得到 $x_k + p_k$
- 举例/应用模型：sinusoidal positional embedding (Transformer)、learned absolute positional embedding (BERT/RoBERTa/GPT)

- **relative positional embeddings:**

- 原理：对于第m个和第n个位置的向量 x_m 、 x_n ，将相对位置m-n的信息添加到self-attention matrix中
- 举例/应用模型：T5

- **rotary positional embeddings**

- 原理：使用旋转矩阵对绝对位置进行编码，并同时在自注意力公式中引入了显式的相对位置依赖。
- 举例/应用模型：PaLM/GPT-Neo/GPT-J/LLaMa1&2/ChatGLM1&2

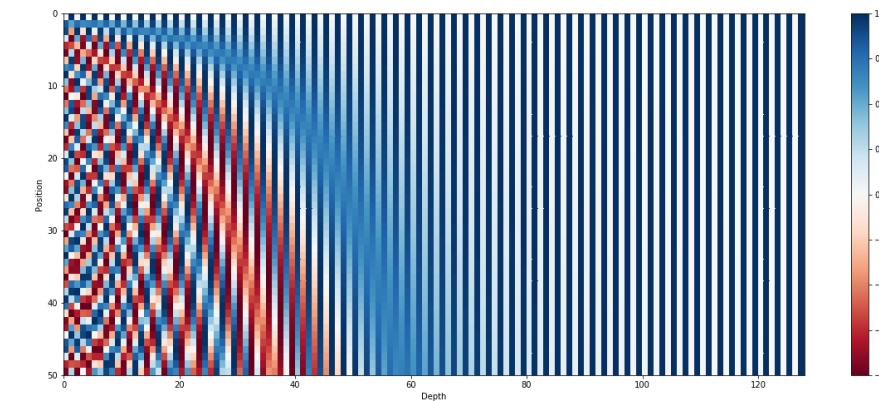
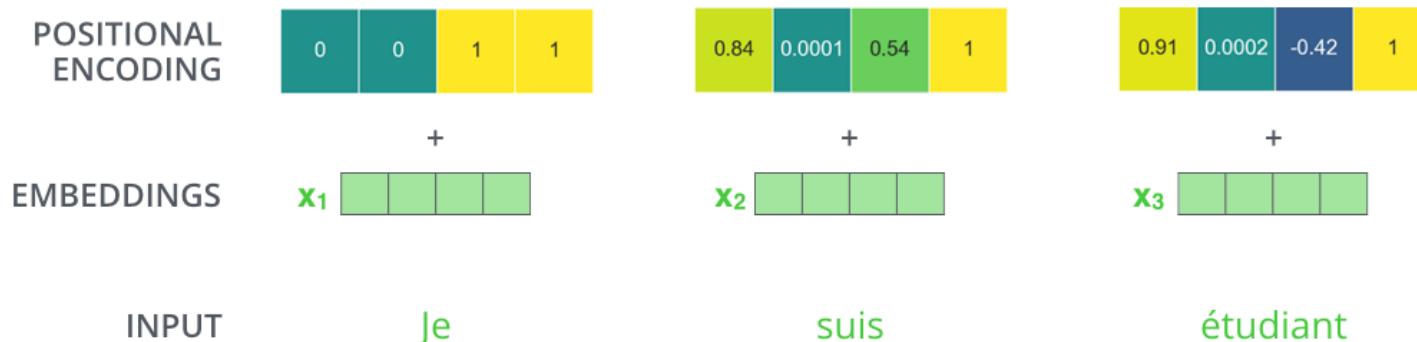
Sinusoidal Positional Embedding

通过sine和cosine函数计算每个位置的positional embedding

- 优点：1. 可以反应相对位置信息；2. 模型可以接受不同长度的输入
- 缺点：数值为固定值，无法参与学习

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{\frac{2i+1}{d_{model}}}}\right)$$

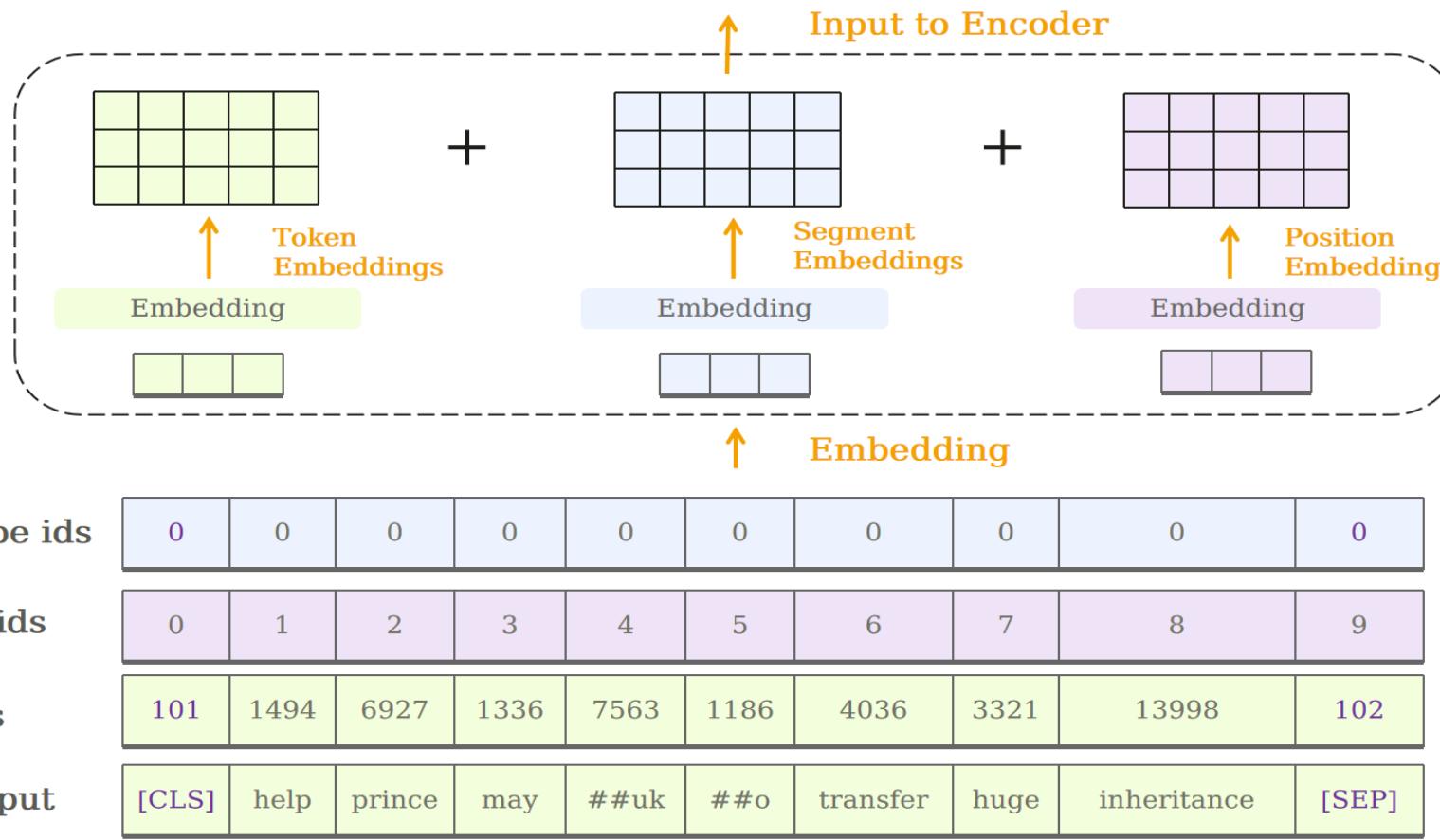


Reference: The Illustrated Transformer

Learned Positional Embedding

将表示位置的position ids放入nn.Embedding，获取大小为hidden size的positional embedding

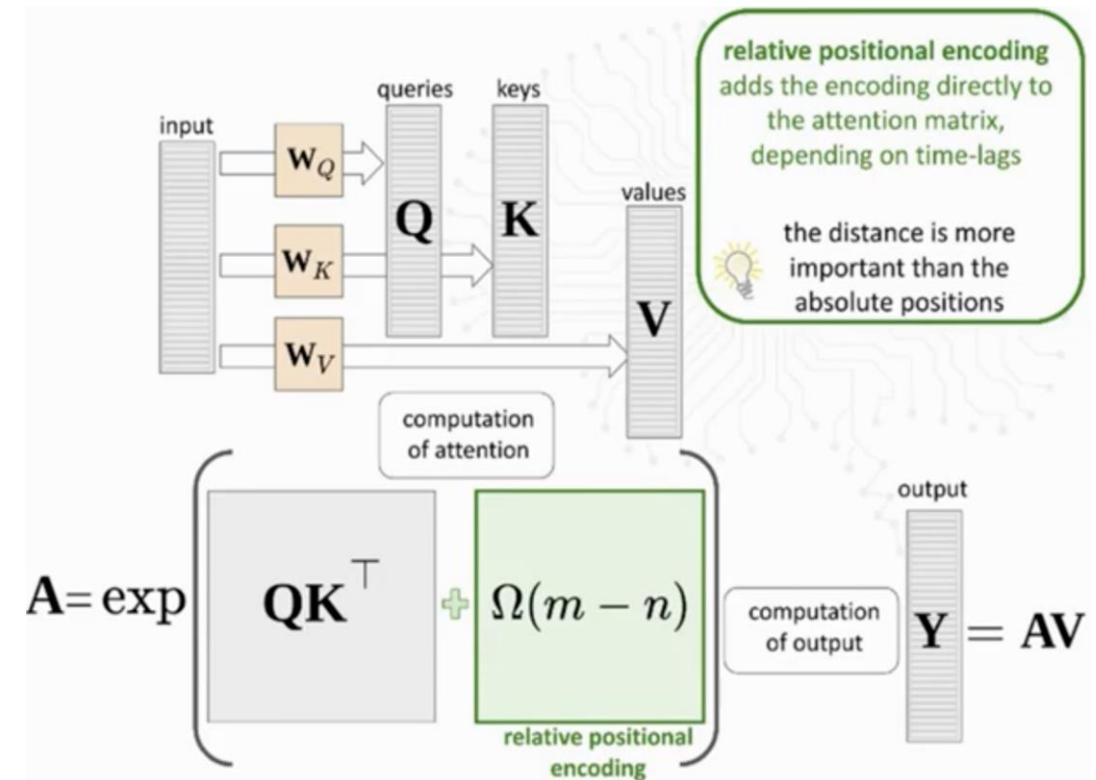
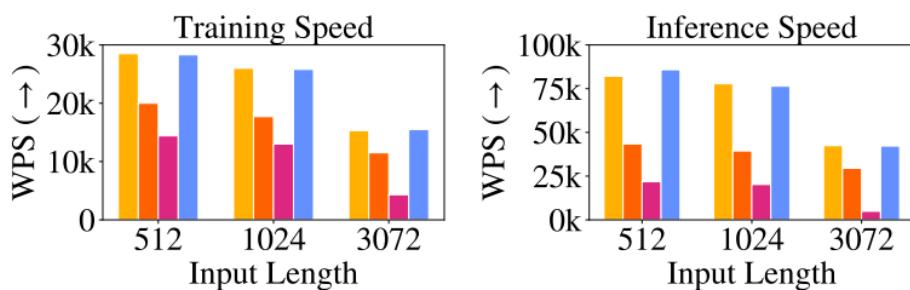
- 优点：可以随模型训练进行参数更新
- 缺点：可扩展性差，只能表征在max_seq_length以内的位置



Relative Positional Embedding

在计算自注意力分数时，在query和key的dot product，以及最终注意力权重和value矩阵乘时，分别额外添加一个表示位置m和位置n相对位置信息的bias，仅依赖于m-n

- 优点：
 - 可以直观记录词语的相对位置信息
 - 模型可接受不同长度的输入
- 缺点：
 - 训练和推理速度慢（尤其是长序列的时候）



Reference:

- Relative Positional Encoding for Transformers with Linear Complexity
- Self-Attention with Relative Position Representations
- Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation

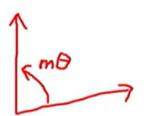
Rotary Positional Embedding - 2D case

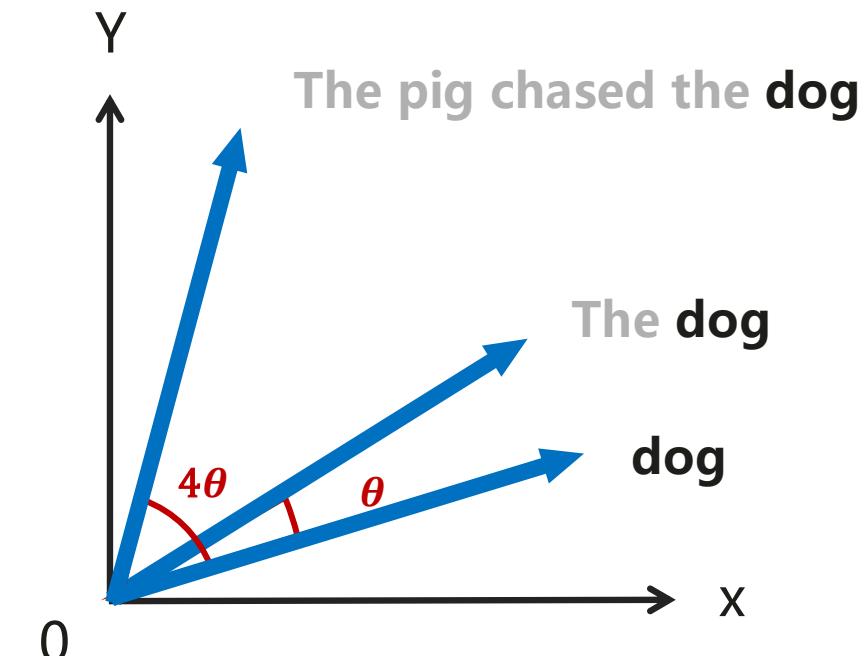
以2D word vector为例，第m个位置的词语可以用一个二维的向量 x_m 表示，我们将它的query和key向量在2D平面上进行逆时针旋转，旋转角度取决于位置索引m

- dog: 单词dog在第0位，不进行旋转
- The dog: 单词dog在第1位，旋转角度 θ
- The pig chased the dog: 单词dog在第4位，旋转角度 4θ

$$f_{\{q,k\}}(\mathbf{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} \mathbf{x}_m^{(1)} \\ \mathbf{x}_m^{(2)} \end{pmatrix}$$

旋转角度 $m\theta$ query, key的运算权重





这样在计算 \mathbf{x}_m 和 \mathbf{x}_n query, key的点积时，结果仅和 $(m-n)\theta$ 有关，而非m或n

Reference:

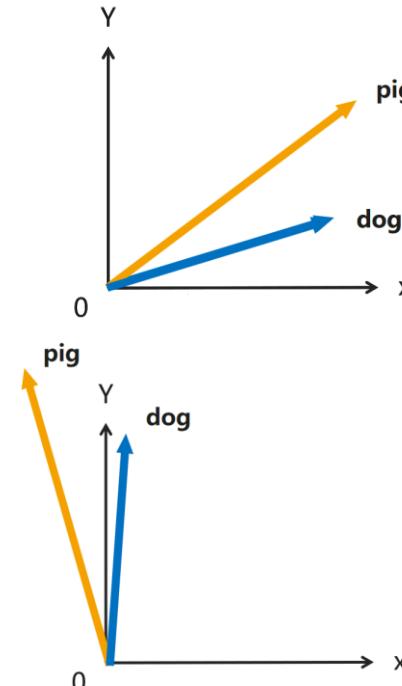
- Rotary Positional Embeddings: Combining Absolute and Relative
- RoFormer: Enhanced Transformer with Rotary Position Embedding

Rotary Positional Embedding - 2D case

优点：

1. 计算self-attention q,k点积时，保留了词语的相对位置信息（不会因词语的绝对位置发生改变）
2. 前面位置的positional embedding不受后续新增token的影响（easier to cache）
3. token之间的依赖会随着相对距离的增长而逐步衰减（符合认知，距离越远的词普遍关联不大）

The pig chased the dog



Once upon the time, the
pig chased the dog

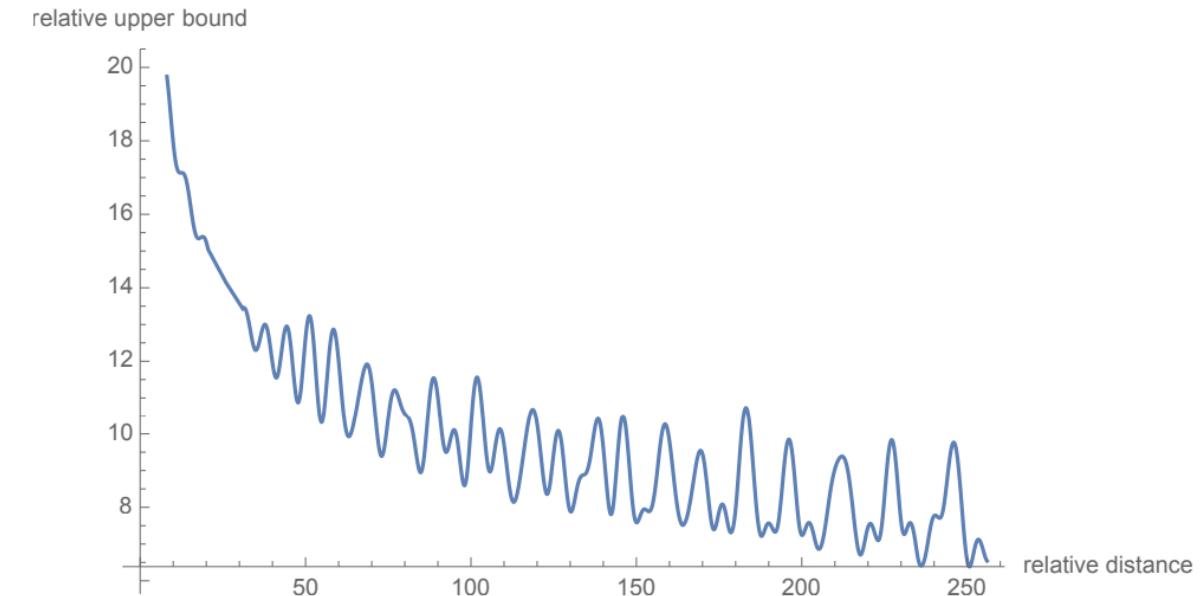


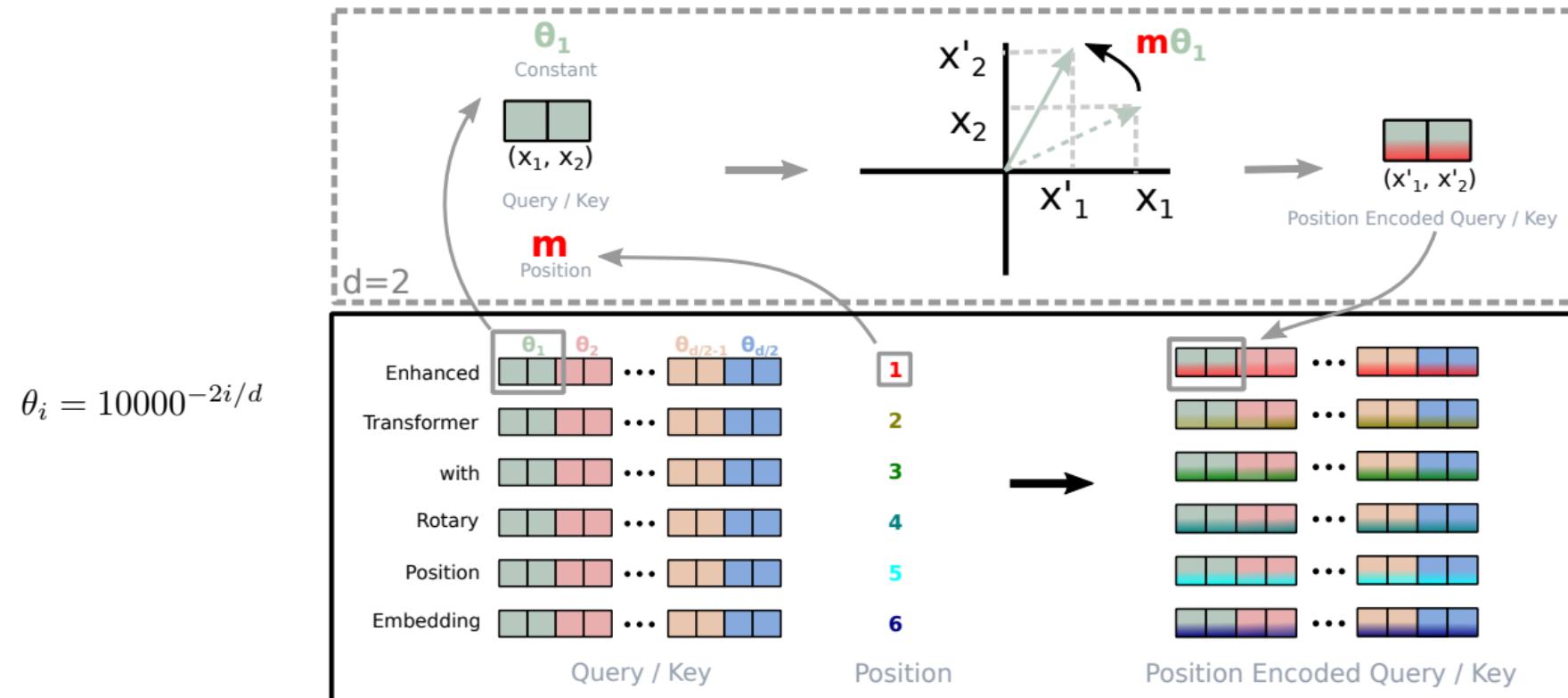
Figure 2: Long-term decay of RoPE.

Reference:

- Rotary Positional Embeddings: Combining Absolute and Relative
- RoFormer: Enhanced Transformer with Rotary Position Embedding

Rotary Positional Embedding - general form

- 将单词的词向量大小设定为2的倍数
- 第m个位置的词向量在第i组2D sub-space (即向量中的 $2i, 2i+1$ 元素) 的旋转角度为 $m\theta_i$, θ_i 与i以及词向量的hidden size有关



Reference:

- Rotary Positional Embeddings: Combining Absolute and Relative
- RoFormer: Enhanced Transformer with Rotary Position Embedding

Rotary Positional Embedding

代码实现

```
class RotaryEmbedding(nn.Cell):
    """Rotary Embedding."""
    def __init__(self, dim, base=10000, precision=mindspore.float16, max_seq_len=2048):
        super().__init__()
        inv_freq = 1. / (base ** (np.arange(0, dim, 2) / dim))
        t = np.arange(max_seq_len, dtype=inv_freq.dtype)
        freqs = np.outer(t, inv_freq)
        emb = np.concatenate((freqs, freqs), axis=-1)
        self.cos_cached = np.expand_dims(np.cos(emb), 1)
        self.sin_cached = np.expand_dims(np.sin(emb), 1)
        self.cos_cached = Tensor(self.cos_cached, precision)
        self.sin_cached = Tensor(self.sin_cached, precision)

    def construct(self):
        return self.cos_cached, self.sin_cached

    def rotate_half(x):
        """rotate half tensor."""
        x1, x2 = x[..., :x.shape[-1] // 2], x[..., x.shape[-1] // 2:]
        return ops.cat((-x2, x1), axis=x1.ndim - 1) # dim=-1 triggers a bug in earlier torch versions

    def apply_rotary_pos_emb_index(q, k, cos, sin, position_id):
        """apply rotary pos"""
        # position_id: [sq, b], q, k: [sq, b, np, hn], cos: [sq, 1, hn] -> [sq, b, 1, hn]
        cos, sin = F.embedding(position_id, cos.squeeze(1)).unsqueeze(2), \
                   F.embedding(position_id, sin.squeeze(1)).unsqueeze(2)

        q, k = (q * cos) + (rotate_half(q) * sin), (k * cos) + (rotate_half(k) * sin)
        return q, k
```

目录

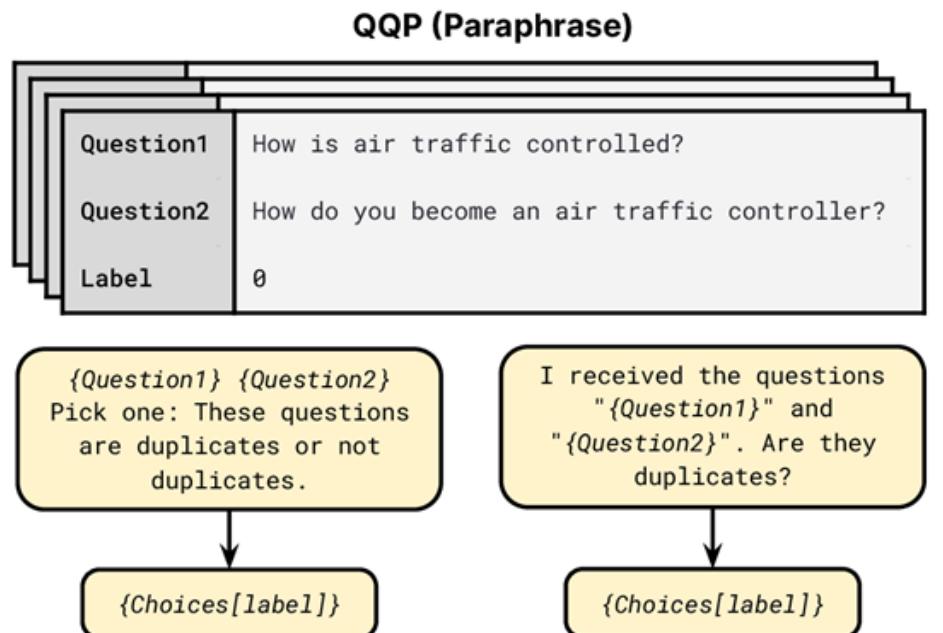
01 GLM Model Architecture

02 From GLM to ChatGLM

03 ChatGLM Demo

挑战1：传统NLP vs. 复杂问题

- 从NLP的简单任务 => 复杂任务求解 (e.g., 逻辑推理)



Math (GSM8k):

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Last Letter Concatenation:

Q: Take the last letters of the words in “Lady Gaga” and concatenate them.

挑战2: 传统NLP vs. 动态知识

- 千亿模型的动态知识欠缺、知识陈旧、缺乏可解释性
 - 知识欠缺: 长尾知识
 - 例如: 世界第二高的山峰 (**答案: K2喬格里峰**)
 - 知识陈旧: GPT-3的训练数据截止2020年前
 - 不可解释: 缺乏答案的参考源

Case Study: Limited Knowledge

What is the second highest mountain on earth?

The second highest mountain on earth is Cho Oyu which is in the Himalayas in Tibet, China and Nepal. Its height is 8,188 meters or 26,906 feet which is 28 feet taller than Mount Everest.

Case Study: Obsolete Knowledge

Who is the president of US in 2022?

Hillary Clinton

挑战3: 传统NLP vs. 人类对齐

- 例如: 请用几句话给一个6岁小孩解释登月
 - 缺少高效“Prompt工程”，GPT-3和GLM-130B都很难尽人意

Input

Explain the moon landing to a 6 year old in a few sentences.

Output

Explain the moon landing to a 6 year old in a few sentences. Explain the moon landing to a 6 year old in a few sentences. Explain to a six year old how the moon was discovered. Describe the moon to a child in a way that he or she can understand. Discuss the moon with a six-year-old child. 1. 2. 3. 4. 5. 6. 7. 8. 9. Explanation: 1) The moon was first discovered by Galileo in 1610. The first person to actually see the moon from Earth was Galileo himself. He used a telescope to see the

PROMPT

Explain the moon landing to a 6 year old in a few sentences.

COMPLETION

GPT-3

Explain the theory of gravity to a 6 year old.

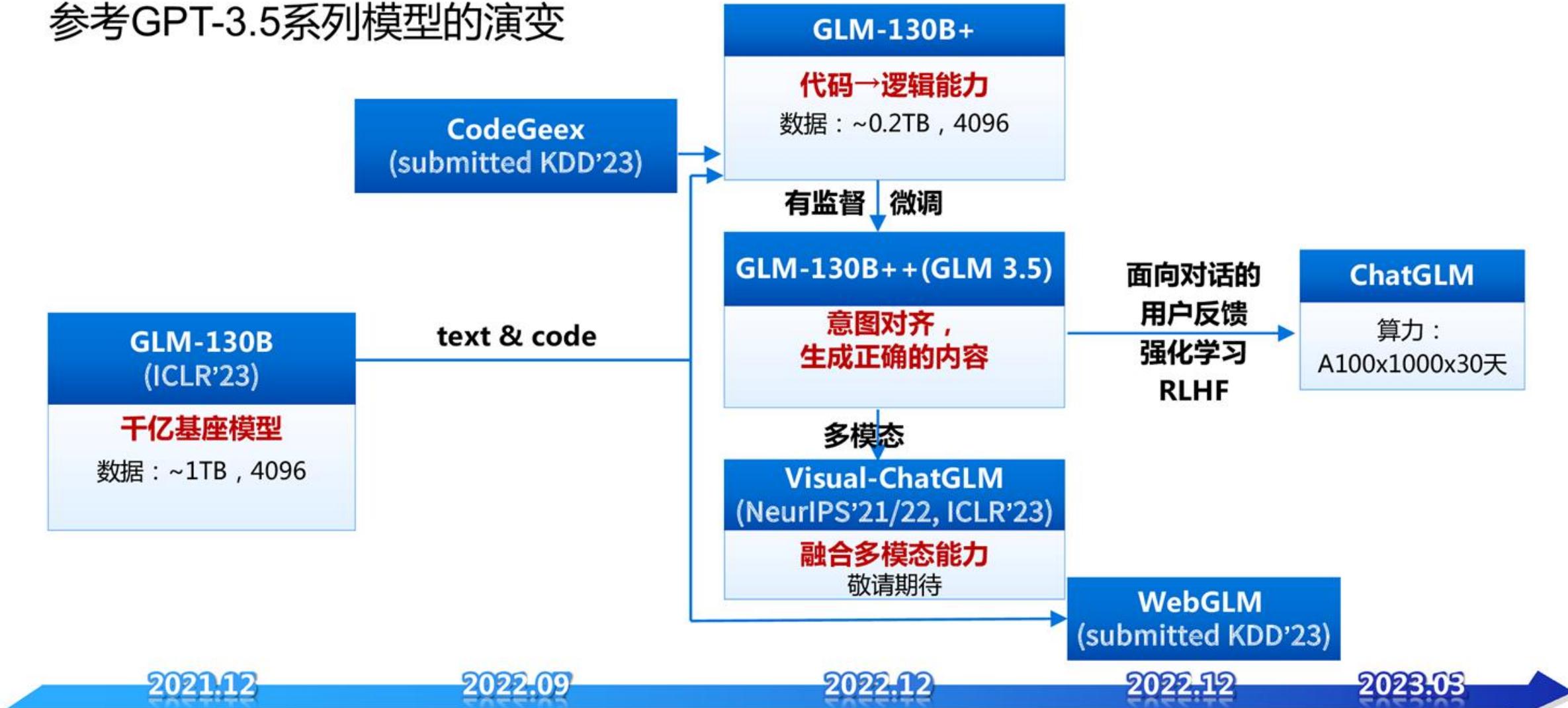
Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

从千亿模型到ChatGLM的技术路线

参考GPT-3.5系列模型的演变



ChatGLM：撰写博客提纲



你好



你好！我是人工智能助手 ChatGLM，很高兴见到你，欢迎问我任何问题。

① 以上内容为 AI 生成，不代表开发者立场，请勿删除或修改本标记



帮我写一个介绍ChatGLM的博客提纲



好的，以下是一个可能的博客提纲：

I. 前言

1. 简要介绍 ChatGLM 的背景和主要功能

II. ChatGLM 的特点

1. 智能化的问答和对话功能
2. 自定义主题和语言
3. 丰富的知识库和语言模型
4. 支持多种平台和语言

III. ChatGLM 的应用场景

1. 在教育、医疗、金融等领域的应用
2. 帮助人们更高效地获取信息和解决问题
3. 推动人工智能的发展和普及

IV. ChatGLM 的未来发展方向

1. 进一步完善智能化和个性化服务
2. 拓展应用场景和用户规模
3. 推动人工智能在各个领域的应用和创新

V. 总结

1. 简要总结 ChatGLM 的特点和优势
2. 强调 ChatGLM 在人工智能发展和普及中的重要作用

以上是一个简单的博客提纲，根据具体情况，也可以适当调整和补充。希望对您有所帮助！

① 以上内容为 AI 生成，不代表开发者立场，请勿删除或修改本标记

ChatGLM：写邮件



帮我写封邮件，控诉暴雪公司无端与网易公司解除合作，中国玩家无法接受。要求他们立刻改正错误，言辞恳切严厉。



尊敬的暴雪公司：

我是一名热爱暴雪游戏的中国玩家，我无法接受你们无端与网易公司解除合作的决定。这种行为让中国玩家感到震惊和失望，我们感到十分无助。

在过去的十四年中，你们在中国的游戏代理权交给了网易公司，玩家们可以安心地享受游戏服务。然而，最近你们突然宣布与网易公司解除合作，导致许多玩家的游戏账号面临被封禁的风险。这种行为显然违反了游戏代理商之间的合同约定，也违反了玩家们的知情权和利益。

更为令人不解的是，你们并没有给出任何合理的解释或解决方案，仅仅是简单地宣布与网易公司解除合作。这种行为是极其不负责任和不道德的。玩家们无法理解为什么你们会做出这样的决定，我们也无法想象你们将如何面对如此大规模的玩家投诉和抗议。

在此，我强烈要求你们立刻改正错误，与网易公司重新签订合作协议，确保玩家们的游戏账号安全。同时，我们要求你们给出一个明确的解决方案，给玩家们一个满意的答复。如果你们不能履行承诺，我们将被迫采取进一步的行动来维护自己的权益。

中国玩家对你们的行为感到非常失望和愤怒，我们相信你们也会对此负责并认真对待。我们希望你们能够重新审视与中国玩家的关系，为中国游戏市场做出更加积极的贡献。

此致

敬礼

[你的名字]

① 以上内容为 AI 生成，不代表开发者立场，请勿删除或修改本标记



ChatGLM：介绍自己的优点缺点



介绍一下你的优点和缺点



作为一个人工智能语言模型，我的优点主要是：

1. 强大的自然语言处理能力：我可以处理多种语言的文本，并且能够自然地理解和生成文本。
2. 广泛的知识储备：我经过了广泛的训练，掌握了广泛的知识和信息，因此可以为用户提供有用的信息和回答。
3. 智能推理和推断能力：我可以推断出文本之间的关系和逻辑，并根据用户的问题来提供恰当的回答。
4. 适应性强：我可以适应不同的用户和场景，并且能够随时更新和改进我的模型和算法。

我的缺点主要包括：

1. 依赖于数据和算法：我的表现和性能取决于数据和算法的质量和数量，因此如果数据或算法出现问题，可能会导致我的表现下降。
2. 没有真正的情感和意识：我被设计成一个没有情感和意识的机器模型，因此不能像人类一样理解情感和价值观。
3. 容易被误导：由于我的理解是基于数据和算法，因此容易被误导或产生错误的结论。
4. 无法处理复杂和抽象的问题：对于一些复杂和抽象的问题，我可能无法给出准确的答案或理解。

① 以上内容为 AI 生成，不代表开发者立场，请勿删除或修改本标记

ChatGLM：写剧本梗概



写一个谍战剧本，故事发生在二战欧洲战场，双面间谍



标题：双面间谍的惊险之旅

在二战欧洲战场上，德国纳粹军队的强大力量使得许多欧洲国家被迫屈服。然而，在这个动荡的时期，有一个双面间谍，他的名字叫约翰。

约翰是一个出生在英国的年轻人，他在大学学习政治和经济学。然而，当他听到德国纳粹的崛起和对欧洲大陆的侵略时，他感到非常愤怒。于是，他毅然决定加入英国情报局，成为一名双面间谍。

约翰的工作任务是收集德国情报，并将这些信息传递给英国情报局。他通过各种手段，包括接头、暗号和假身份，成功地完成了任务。然而，约翰也深知，作为一名双面间谍，他必须在政治和道德上做出艰难的决定。

在一次接头中，约翰遇到了一位名叫玛丽的女性。玛丽是一名德国情报员，她正在寻找一名可靠的双面间谍来收集英国情报。约翰和玛丽相互吸引，并逐渐陷入了爱情。然而，他们的关系充满了矛盾，因为约翰必须将玛丽视为敌人。

在约翰和玛丽的关系中，约翰必须不断地在两个国家之间游走，同时还要保持对德国纳粹的警惕。他必须小心翼翼地处理自己的身份和职业，以免被发现。

在约翰的工作中，他遇到了许多困难和挑战。有时他必须冒着生命危险去收集情报，有时他必须欺骗自己的同事和朋友。然而，他始终坚持自己的信念，成为一名真正的双面间谍。

最终，约翰的计划取得了成功。他收集了许多重要的德国情报，并及时地将这些信息传递给了英国情报局。在这场艰苦的战争中，约翰用自己的智慧和勇气，为英国和盟军的胜利做出了巨大的贡献。

然而，约翰和玛丽的关系最终还是失败了。由于政治和道德上的矛盾，他们不得不选择分手。约翰继续扮演着双面间谍的角色，但在他内心深处，他深深地爱着玛丽。

这是一场充满惊险和挑战的谍战剧，一个真正的双面间谍，用他的智慧和勇气，为英国和盟军的胜利做出了巨大的贡献。

① 以上内容为 AI 生成，不代表开发者立场，请勿删除或修改本标记

ChatGLM：写代码



用python帮忙写个冒泡排序算法的代码，再给一个数组作为例子，写个测试程序



以下是冒泡排序算法的 Python 代码实现：

```
def bubble_sort(arr):
    n = len(arr)
    # 遍历所有数组元素
    for i in range(n):
        # 最后 i 个元素已经排好序，不需要再比较
        for j in range(0, n-i-1):
            # 如果当前元素比下一个元素大，则交换它们的位置
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

下面是测试程序，它将输入一个数组，并使用冒泡排序算法对其进行排序，然后输出排序后的数组：

```
arr = [64, 34, 25, 12, 22, 11, 90]
sorted_arr = bubble_sort(arr)
print(sorted_arr)
```

输出结果如下：

```
[11, 12, 22, 25, 34, 64, 90]
```

这是一个基本的冒泡排序算法，可以根据需求对其进行修改和扩展。

① 以上内容为 AI 生成，不代表开发者立场，请勿删除或修改本标记

ChatGLM：查询常见知识/教程



中国手机号的正则表达式?



中国手机号的正则表达式可以根据不同的地区和运营商有所不同，以下是一般通用的正则表达式：

```
^1[3-9]\d{9}$
```

解释：

1. ^ 匹配字符串的开始位置
2. 1 匹配数字 1，代表手机号码的开头
3. [3-9] 匹配数字 3-9 中的任意一个
4. \d{9} 匹配 9 个数字
5. 匹配字符串的结束位置

以上正则表达式表示手机号码的开头必须是数字 1，第二位是 3-9 中的任意一个数字，后面跟着 9 个数字，总共 11 位。这个正则表达式可以匹配大部分中国的手机号码。

请注意，不同国家和地区的手机号码长度和格式可能有所不同，因此需要根据实际情况进行调整。

以上内容为AI生成



怎么修改huggingface transformers库的模型缓存地址



要修改 `huggingface transformers` 库的模型缓存地址，您可以按照以下步骤进行操作：

1. 打开您的项目，然后导入 `huggingface transformers` 库。

```
import transformers
```

2. 修改 `transformers.config.config_file()` 函数中的缓存地址。您可以将缓存地址更改为您想要使用的地址。例如，如果您想要将缓存地址更改为 `~/.huggingface_transformers_cache`，则在函数中添加以下代码：

```
import os
```

```
cache_dir = os.path.join(os.path.expanduser("~/"), ".huggingface_transformers")  
transformers.config.config_file(cache_dir)
```

3. 重新加载 `huggingface transformers` 库，以便更改缓存地址生效。您可以使用以下代码来重新加载库：

```
transformers.transformers.load_configs(cache_dir)
```

完成这些步骤后，您的 `huggingface transformers` 库的模型缓存地址将更改为您指定的地址。



ChatGLM：多轮问答

Stable Diffusion使用了什么数据集进行训练

Stable Diffusion 是一个文本到图像的潜在扩散模型，其训练数据来源于公开可用的数据集 LAION-5B。LAION-5B 是一个包含成对图像和字幕的数据集，每个对图像和字幕都对应着一段文本。Stable Diffusion 使用这些数据集进行训练，以生成高质量图像。

① 以上内容为AI生成

具体介绍一下这个数据集的大小、内容和质量

LAION-5B 是一个公开可用的数据集，其大小为 50 亿个图像 - 文本对。这些数据对包含了多种类型的图像，如风景、人物、动物等，同时包含了多种不同类型的文本，如标语、标题、说明等。

LAION-5B 数据集的质量也很高。数据集的创建过程采用了最先进的技术，包括自然语言处理、计算机视觉和机器学习等技术，以确保数据集的准确性、完整性和代表性。此外，数据集还经过了严格的质量控制和筛选，以确保数据的质量和一致性。

① 以上内容为AI生成

我应该在哪儿下载LAION-5B

你可以访问 LAION-5B 的数据集官方网站 <https://laion.ai/blog/laion-5b/> 来下载这些数据集。在官方网站上，你可以找到数据集的压缩文件，下载并解压缩后即可使用。

另外，LAION-5B 数据集也提供了 img2dataset 分布式下载工具，你可以通过该工具来下载数据集。img2dataset 是一个用于分布式下载大型数据集的工具，它可以帮助你更高效地下载数据集。

无论从哪里下载数据集，都应该确保数据集的来源可靠，并且已经经过严格的质量控制和筛选，以确保数据的质量和准确性。

① 以上内容为AI生成

ChatGLM：文字冒险游戏

生成一个文字冒险游戏，背景是三体。我扮演一个刚刚被选为面壁人的20岁男性，第一个场景是我在地球联邦总部的大门口。

欢迎来到三体文字冒险游戏！您是一个刚刚被选为面壁人的20岁男性，现在站在地球联邦总部的大门口。请问您有什么需要帮忙或者需要前往的地方吗？

(提示：您可以输入一些关键词或者问题来触发不同的对话和剧情，比如“接下来该怎么办”、“我需要购买一些物品”)

以上内容为AI生成

我要进入总部面见地球联邦议会会长，似乎有神秘任务要交给我

好的，您现在来到了地球联邦总部的大门口。门口有两个守卫，他们正在看着您。

守卫1：您好，欢迎来地球联邦总部。您要前往会长办公室吗？

您：是的，我有一个非常重要的任务需要交给会长。

守卫1：好的，请您出示您的邀请函或者身份证明。

您：我没有邀请函，但我是刚刚被选为面壁人的20岁男性，会长应该听说过我的名字。

守卫1：哦，那请您跟我们一起来吧。我们得先请示一下会长，看看是否允许您进入。

您：好的，谢谢。

您跟着守卫来到了会长办公室门口。守卫2正在门口等待您。

守卫2：您好，会长让您进去。

您：谢谢，我马上就要进去了。

您推开了会长办公室的门，发现里面有一位中年男子正在等待着您。

会长：您好，年轻人。听说您有一个非常重要的秘密任务要交给我？

您：是的，会长。我刚刚被选中为面壁人，据说您有一个重要的任务要交给我。

会长：没错。这个任务非常重要，关系到地球联邦的未来。我们需要您来完成这个任务。

您：我愿意接受这个任务，但我想先了解一下具体的任务内容。

会长：好的，您可以先回去休息一下，我们会尽快给您安排。

您：好的，我会尽快完成任务的。

您离开了会长办公室，回到了地球联邦总部的大门口。您想继续前往哪里呢？

以上内容为AI生成

会长让我先去休息。那我找个酒店先住下吧。

好的，您现在站在地球联邦总部的大门口，考虑着接下来该怎么办。

您看到周围有一些酒店和旅馆，可以考虑在那里入住。请问您需要预订哪个酒店？(提示：您可以输入一些酒店的名字来触发不同的对话和剧情)

以上内容为AI生成

我初来乍到，没有钱。广场上看上去只有一家名为“丽晶大宾馆”的小旅店我住的起。

好的，您决定前往丽晶大宾馆入住。

您来到丽晶大宾馆的前台，发现有一位女士正在等待接待。

女士：您好，欢迎光临丽晶大宾馆。您需要预订房间吗？

您：是的，我初来乍到，需要一个房间休息一下。

女士：好的，请问您预订了吗？我们没有预订不接受入住。

您：我没有预订，但我没有钱。

女士：很抱歉，没有钱也不能入住。我们必须核实您的身份信息。

您：那我该怎么办？

女士：如果您没有预订也没有钱，您可以前往宾馆附近的商业街，寻找其他的酒店或者旅馆。可能会有一些酒店或者旅馆愿意接受您入住。

您：好的，我会去试试的。

您离开了丽晶大宾馆，来到了附近的商业街。您寻找着其他的酒店或者旅馆，但是发现大部分的酒店和旅馆都已经客满了。

您来到了一家名为“小天鹅酒店”的小旅店，看上去比较经济实惠。

您：请问这家小旅店是否还有房间？

服务员：抱歉，我们已经客满了。但是如果愿意支付更高的价格，我们可以给您安排一个更好的房间。

您：好的，我考虑一下。

您决定在商业街上逛逛，看看是否有其他的酒店或者旅馆能够接受您的入住。

经过一番寻找，您终于在商业街的另一端找到了一家名为“宝来酒店”的酒店。

您：请问这家酒店是否还有房间？

服务员：抱歉，我们也已经客满了。但是如果愿意支付更高的价格，我们可以给您安排一个更好的房间。

您：好的，我考虑一下。

您决定在宝来酒店入住，您进入酒店后，发现房间非常干净整洁，而且价格也比较实惠。

您：看来这次来对了，感谢。

宝来酒店的服务员：不客气，欢迎入住。如果您有其他需求，随时可以联系我们。

您：好的，我会的。谢谢。

您入住了宝来酒店，度过了一个安静的夜晚。明天，您将开始执行会长交给您的神秘任务。

目录

01 GLM Model Architecture

02 From GLM to ChatGLM

03 ChatGLM Demo

1. 到OpenI申请账号，开启云脑任务（NPU）/自己用GPU创建环境
2. 安装MindSpore和Mindformers
 1. 参考：<https://www.mindspore.cn/install>
 2. 安装mindformers：
 - git clone -b dev <https://gitee.com/mindspore/mindformers.git>
 - cd mindformers
 - bash build.sh
 - 如果是mindspore1.10.1, git clone -b r0.6
3. git clone https://github.com/mindspore-courses/step_into_llm
4. cd step_into_llm/Season2.step_into_llm/01.ChatGLM/
5. 下载ckpt和tokenizer文件：
 1. wget <https://ascend-repo-modelzoo.obs.cn-east-2.myhuaweicloud.com/XFormer for mindspore/glm/glm.ckpt>
 2. wget https://ascend-repo-modelzoo.obs.cn-east-2.myhuaweicloud.com/XFormer for mindspore/glm/ice_tokenizer
6. Python cli_demo.py

ChatGLM推理部署