

ChatGLM2

目录

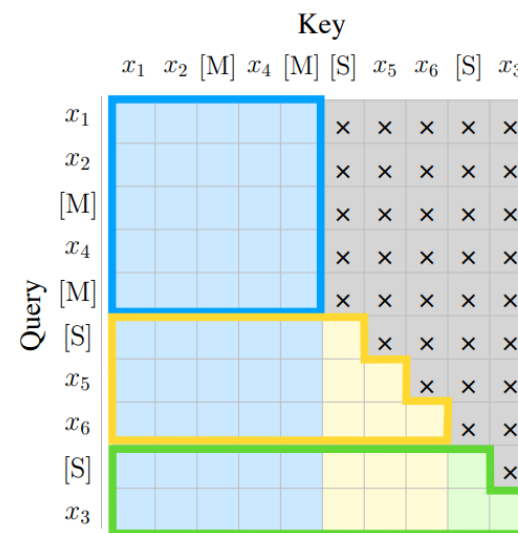
- ChatGLM2技术解析（基于ChatGLM的改进点）
 - Multi-Query Attention
 - Flash Attention
- ChatGLM2推理部署代码演示
- ChatGLM3特性介绍

复习一下GLM模型

采取自回归填空（Autoregressive Blank Infilling）框架，使得模型具有兼顾自然语言理解和文本生成的能力

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	✓	✓	✓

Like a complete unknown, like a rolling stone



(d) Self-attention mask

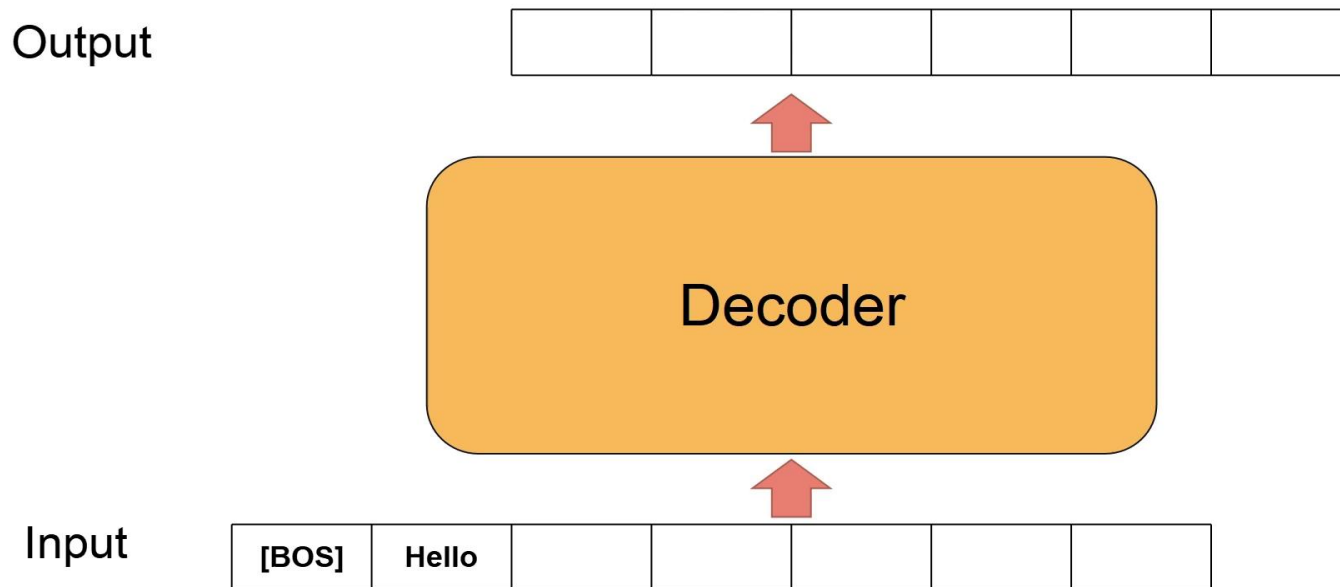
ChatGLM2-6B 是开源中英双语对话模型 ChatGLM-6B 的第二代版本，在保留了初代模型对话流畅、部署门槛较低等众多优秀特性的基础之上，ChatGLM2-6B 引入了如下新特性：

- 更强大的性能=混合目标函数+1.4T中英标识符
- 更长的上下文=Flash Attention技术+上下文长度扩展到32K+8K训练+多轮对话
- 更高效的推理=Multi-Query Attention技术+INT4量化
- 更开放的协议=对学术研究完全开发，受限商用

Multi-Query Attention

Motivation

Transformer结构的模型的推理被称作“incremental inference”，即decoder根据当前输入预测下一个token，生成的token再加入原序列组成新的输入，以此往复，直到出现终止符或达到可生成的最大长度。



因此每一次预测，都需要加载一次权重，导致推理效率较低。

之所以没有关注到，是因为之前很少做文本生成，解码序列长度也没有现阶段大模型的要求那么高。

Multi-Query Attention

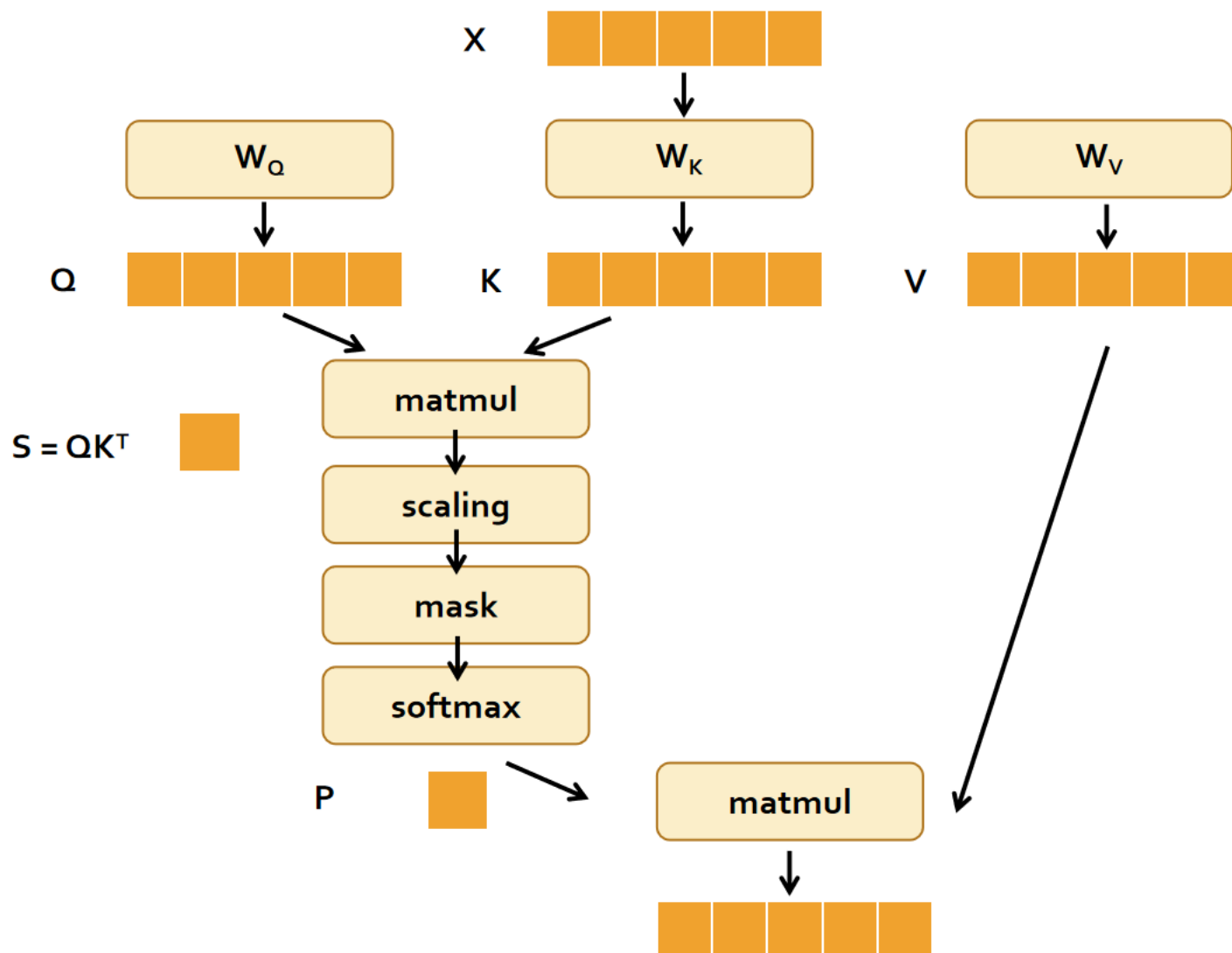
MQA最早是出现在2019年谷歌的一篇论文Fast Transformer Decoding: One Write-Head is All You Need。MQA的思想其实比较简单(如果对MHA比较熟悉的话)，论文中给出的描述如下：

3 Multi-Query Attention

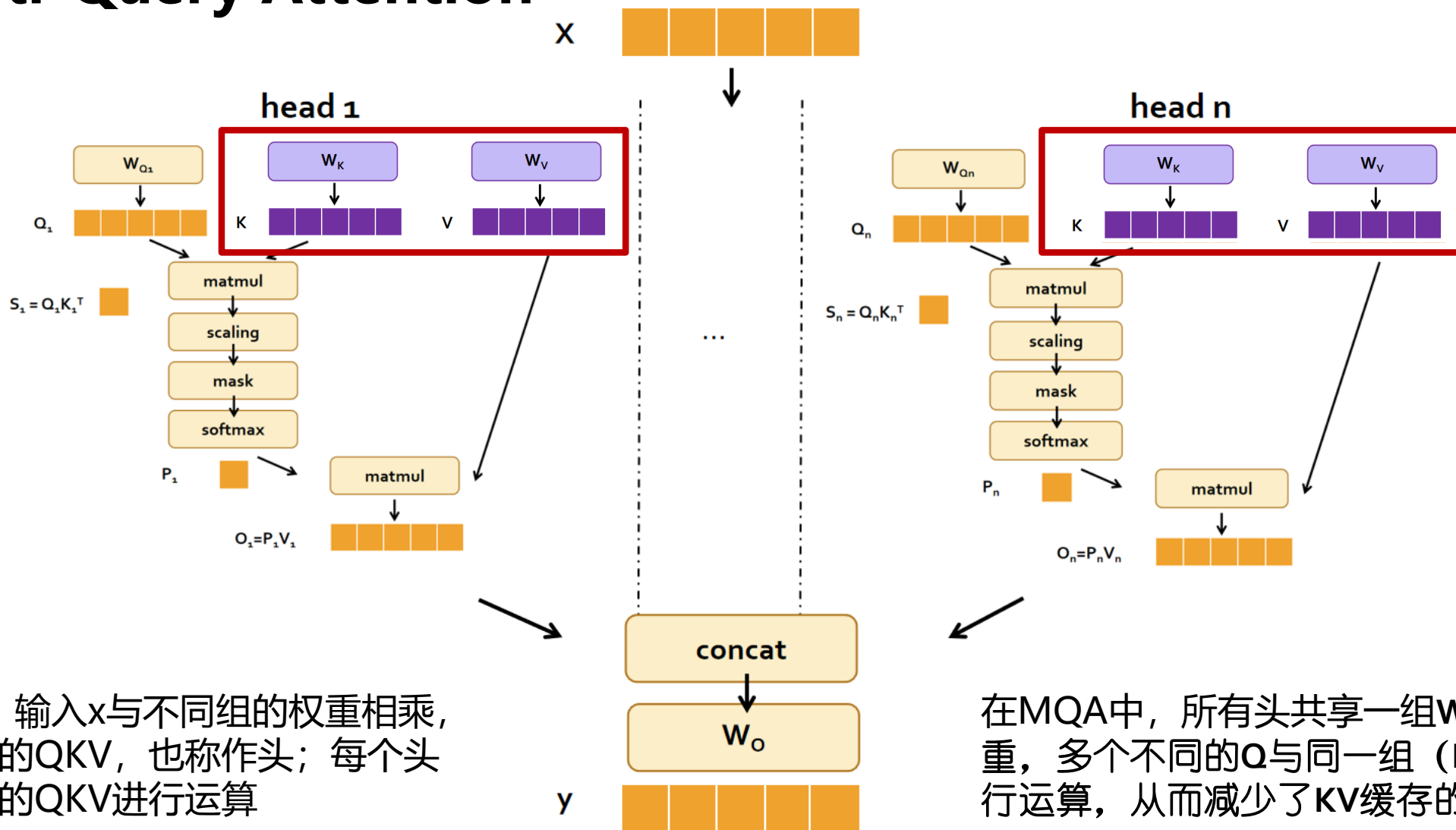
We introduce **multi-query Attention** as a variation of multi-head attention as described in [Vaswani et al., 2017]. Multi-head attention consists of multiple attention layers (heads) in parallel with different linear transformations on the queries, keys, values and outputs. Multi-query attention is identical except that the different heads share a single set of keys and values. The code for (incremental) multi-query (self) attention is identical to the code listed above for multi-head attention, except that we remove the letter "h" from the `tf.einsum` equations where it represents the "heads" dimension of K , V , P_k , or P_v .

论文的意思是：MQA和MHA除了不同的attention head共享一份keys和values权重之外，其他的都是一样的。

Attention



Multi-Query Attention



Multi-Query Attention

Table 2: Amortized training and inference costs for WMT14 EN-DE Translation Task with sequence length 128. Values listed are in TPUv2-microseconds per output token.

Attention Type	Training	Inference enc. + dec.	Beam-4 Search enc. + dec.
multi-head	13.2	1.7 + 46	2.0 + 203
multi-query	13.0	1.5 + 3.8	1.6 + 32
multi-head local	13.2	1.7 + 23	1.9 + 47
multi-query local	13.0	1.5 + 3.3	1.6 + 16

Table 3: Billion-Word LM Benchmark Results.

Attention	h	d_k, d_v	d_{ff}	dev-PPL
multi-head	8	128	8192	29.9
multi-query	8	128	9088	30.2
multi-head	1	128	9984	31.2
multi-head	2	64	9984	31.1
multi-head	4	32	9984	31.0
multi-head	8	16	9984	30.9

推理速度上生成一个token时MHA和MQA的encoder分别耗时1.7us和1.5us, 而decoder分别46us和3.8us, 说明decoder上MQA比MHA快很多。另外在效果上MQA的PPL(越小越好)有所上升, BLEU(越大越好)有所下降, 换句话说就是效果有所下降。

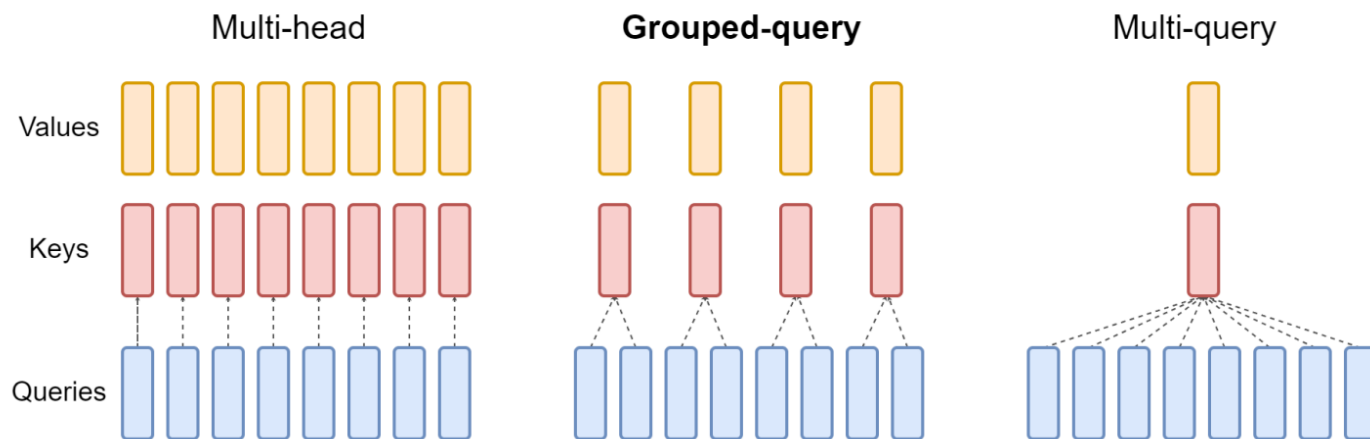
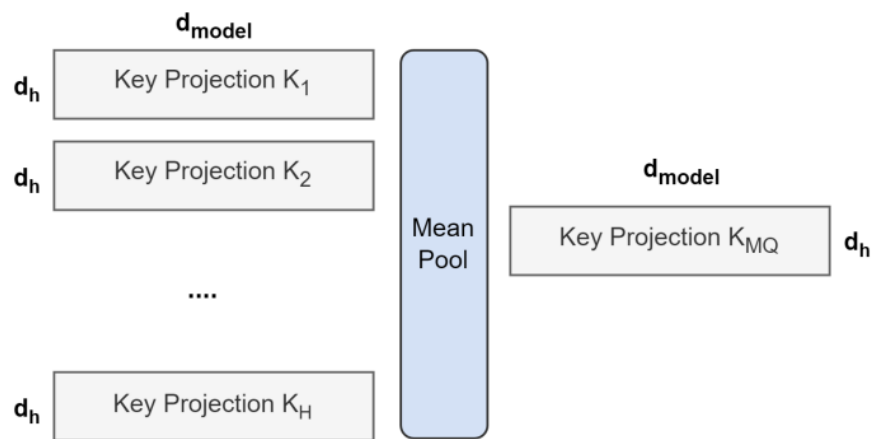
Multi-Query Attention

生成 2000 个字符的平均速度对比如下

Model	推理速度 (字符/秒)
ChatGLM-6B	31.49
ChatGLM2-6B	44.62

Multi-Query Attention的演变: Grouped-Query Attention

Grouped-Query Attention将Q分为G组，每组共享一对K, V。使得模型的质量高于 MQA，但比 MHA 更快。



Uptraining: 将MHA预训练模型转换为MQA

GQA

Multi-Query Attention代码实现

```
self.multi_query_attention = config.multi_query_attention
self.qkv_hidden_size = 3 * self.projection_size

if self.multi_query_attention:
    self.num_multi_query_groups_per_partition = config.multi_query_group_num
    self.qkv_hidden_size = (
        self.projection_size + 2 * self.hidden_size_per_attention_head * config.multi_query_group_num)
```

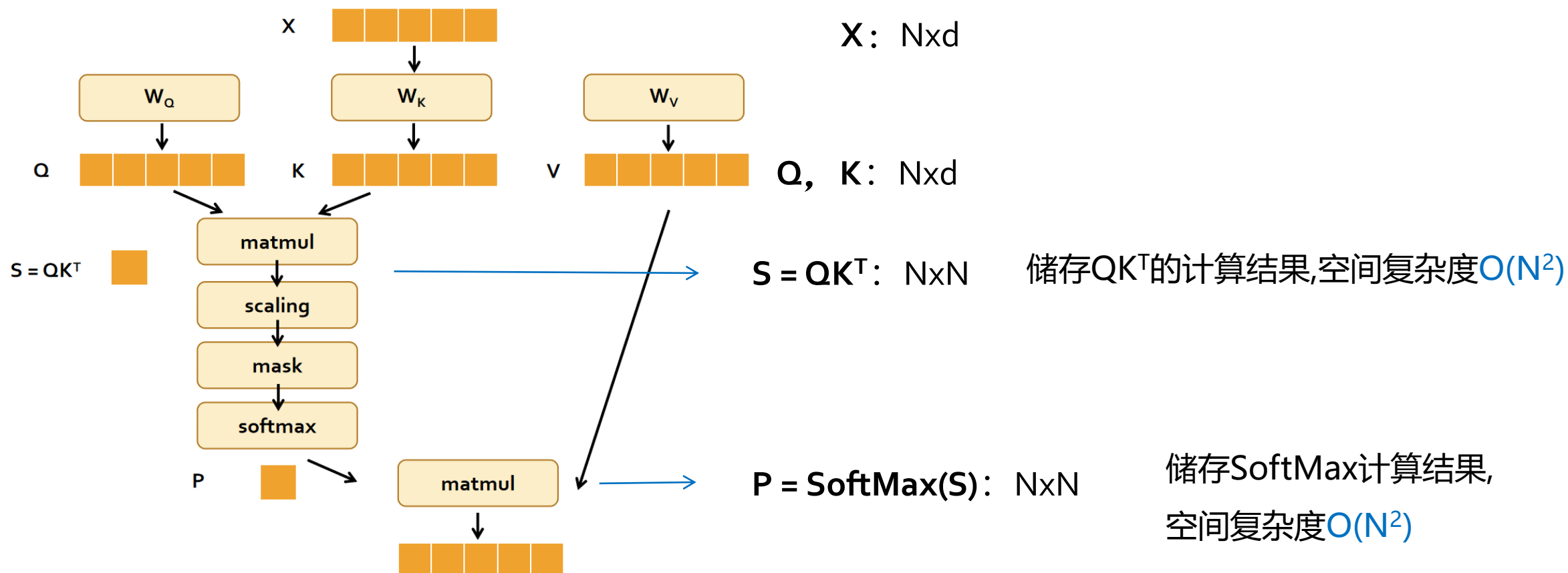
参考链接:

https://gitee.com/mindspore/mindformers/blob/dev/mindformers/models/glm2/glm2_transformer.py

FlashAttention

Motivation

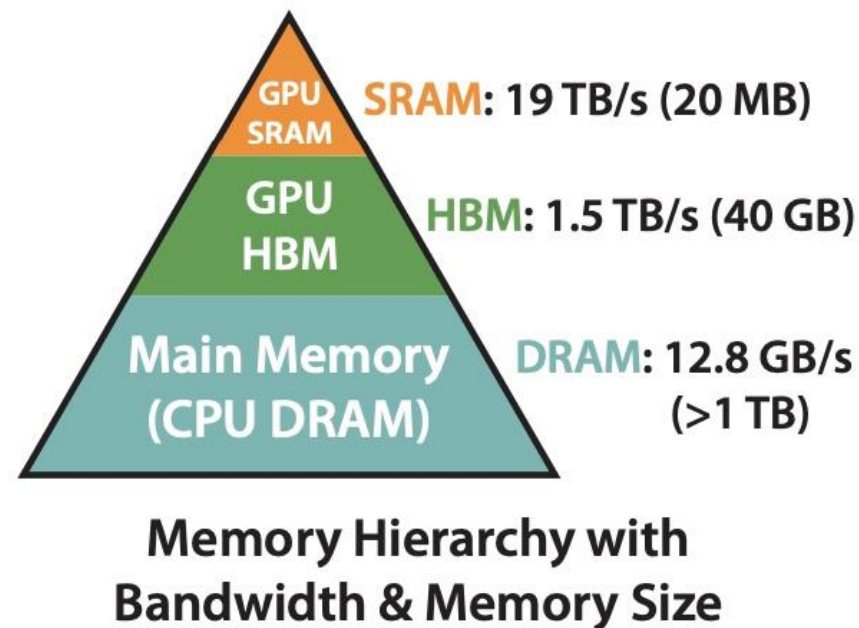
因为Transformer的自注意力机制(self-attention)的计算的**时间复杂度**和**空间复杂度**都与序列长度有关，所以在处理长序列的时候会变的更慢，同时内存会增长更多。



Flash Attention技术

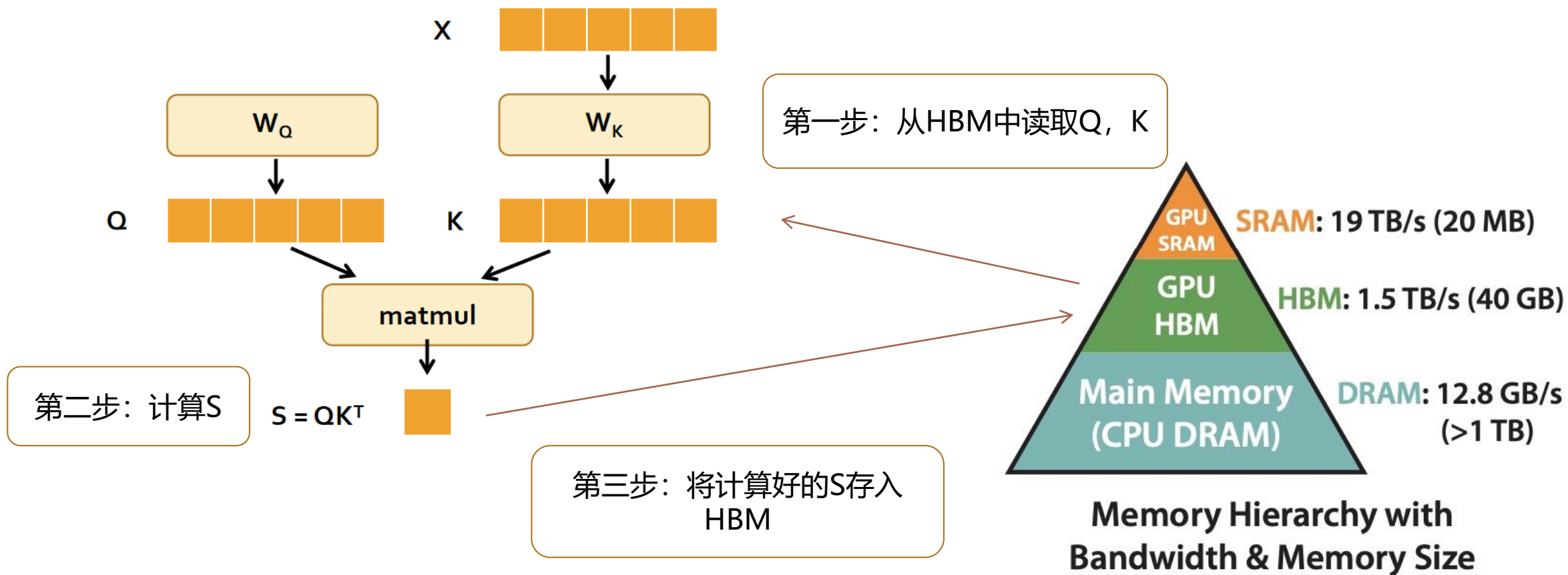
通常的优化是针对计算复杂度(通过FLOPs数衡量), 优化会权衡模型质量和计算速度。

在FlashAttention中考虑到attention算法也是IO敏感的, 通过对GPU显存访问的改进来对attention算法的实现进行优化。如图, 在GPU中片上存储SRAM访问速度最快, 对应的HBM(high bandwidth memory)访问速度较慢, 为了加速要尽量减少HBM的访问次数。



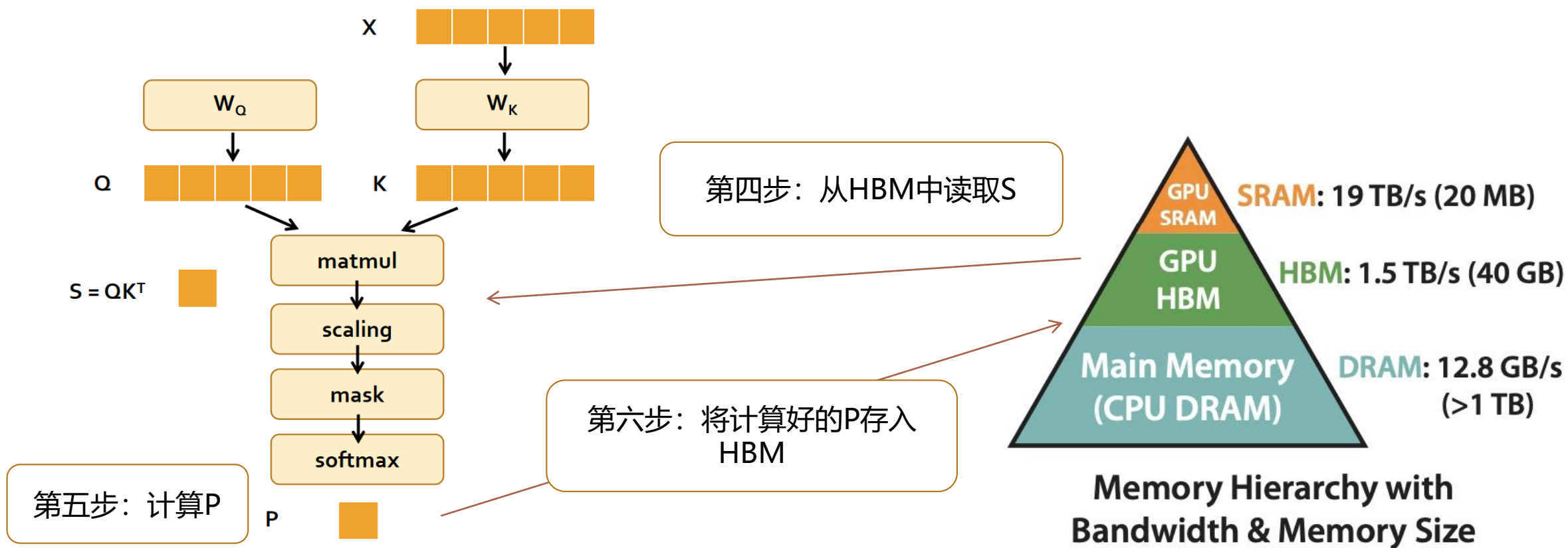
Flash Attention技术

让我们看看通常情况下是如何访问HBM进行Attention计算的



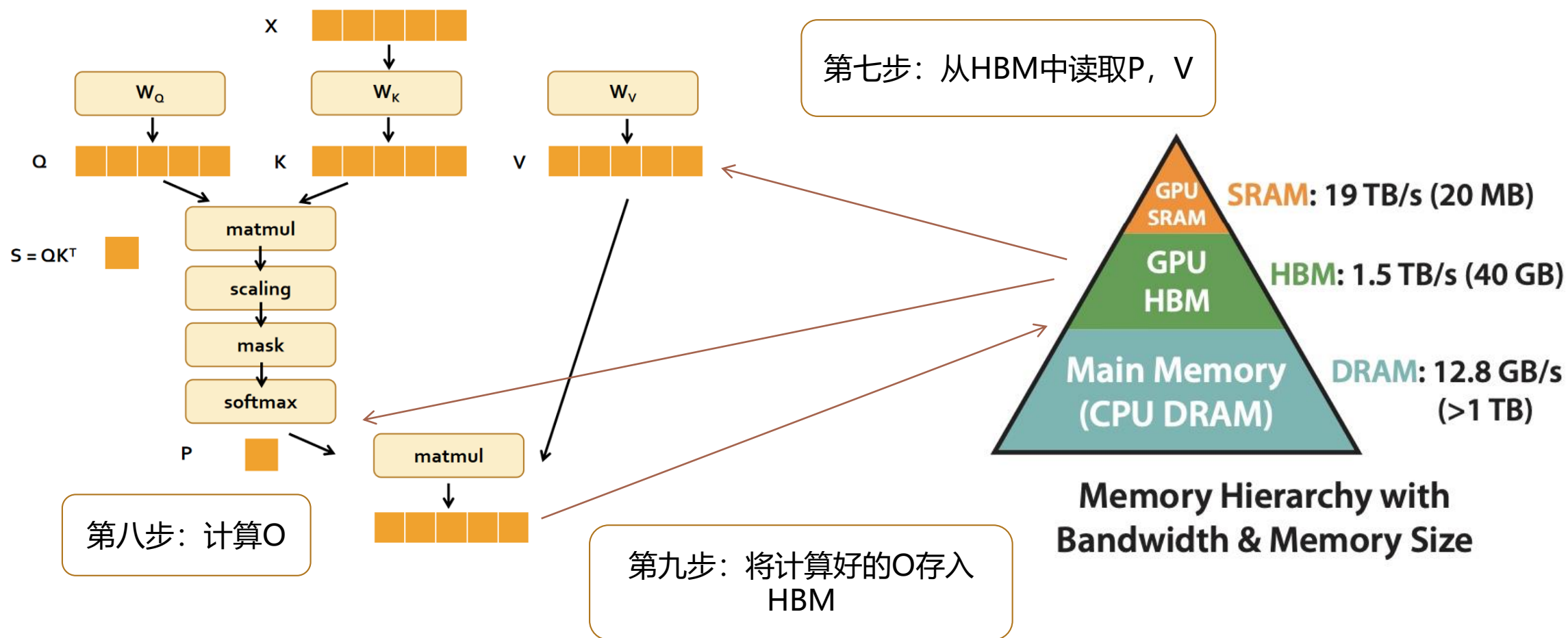
Flash Attention技术

让我们看看通常情况下是如何访问HBM进行Attention计算的



Flash Attention技术

让我们看看通常情况下是如何访问HBM进行Attention计算的

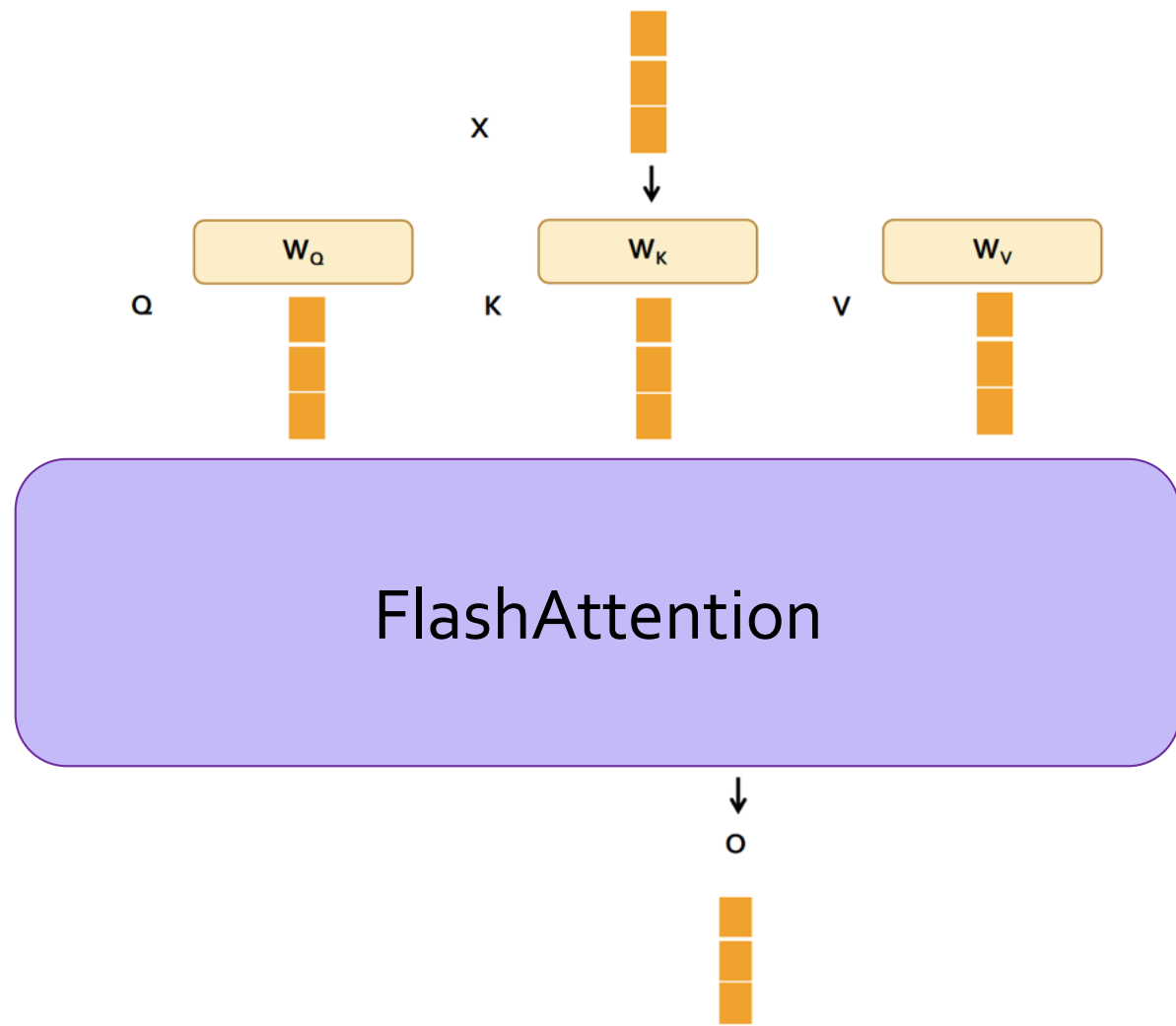


Flash Attention技术

为了减少对HBM的读取，FlashAttention把attention计算融合为一个算子，将参与计算的矩阵送入SRAM，来提高整体的读写速度。

但SRAM容量较小，需要将矩阵分块送入，同时动态更新softmax结果（softmax的tiling展开）。

不对中间的运算结果（P，S）进行存储，在反向传播过程中重新计算需要的数据。

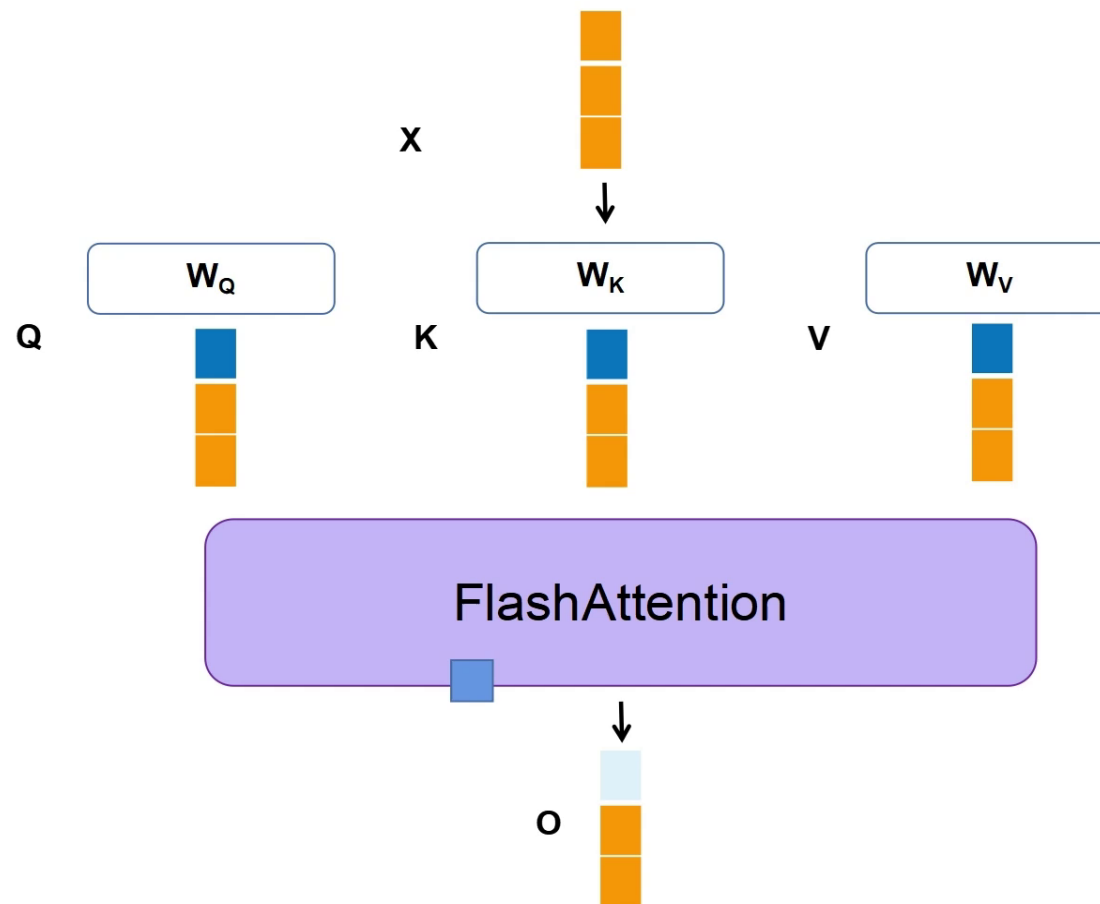


Flash Attention技术

为了减少对HBM的读取，FlashAttention把attention计算融合为一个算子，将参与计算的矩阵送入SRAM，来提高整体的读写速度。

但SRAM容量较小，需要将矩阵分块送入，同时动态更新softmax结果（softmax的tiling展开）。

不对中间的运算结果（P，S）进行存储，在反向传播过程中重新计算需要的数据。



Flash Attention技术

结果展示

Table 1: Training time of BERT-large, starting from the same initialization provided by the MLPerf benchmark, to reach the target accuracy of 72.0% on masked language modeling. Averaged over 10 runs on 8×A100 GPUs.

BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [58]	20.0 ± 1.5
FLASHATTENTION (ours)	17.4 ± 1.4

CSDN

Model implementations	OpenWebText (ppl)	Training time (speedup)
GPT-2 small - Huggingface [87]	18.2	9.5 days (1.0×)
GPT-2 small - Megatron-LM [77]	18.2	4.7 days (2.0×)
GPT-2 small - FLASHATTENTION	18.2	2.7 days (3.5×)
GPT-2 medium - Huggingface [87]	14.2	21.0 days (1.0×)
GPT-2 medium - Megatron-LM [77]	14.3	11.5 days (1.8×)
GPT-2 medium - FLASHATTENTION	14.3	6.9 days (3.0×)

CSDN

ChatGLM2推理部署代码演示

启智社区使用

1. 导航至启智AI协作平台 (<https://openi.pcl.ac.cn/>)，进行账号注册或登录
2. 在最上方搜索栏中输入“step_into_llm”，并选择第一个搜索结果



启智社区使用

3. 代码仓内有昇思MindSpore技术公开课第一&二期的课件ppt与实验代码，点击右上角派生代码仓

4. 按照指引填写项目名称、路径，后点击派生项目

启智 AI协作平台 Powered by C²NET

个人中心 项目 数据集 模型 探索 论坛

搜索...

关于只保留最近5个调试任务的公告>>> 关于删除超过100G的智算集群GPU镜像

mindspore-courses / step_into_llm

镜像自地址 https://github.com/mindspore-courses/step_into_chatgpt

已关注 13 点赞 35 派生 312

代码 任务 2 数据集 模型 云脑 ? 项目设置

72 提交 2 分支 119 MB 581 次下载

分支: master

WEB IDE HTTPS SSH https://openi.pcl.ac.cn/mindspore-courses/step_into_llm

xing-yiren 360c8a72b3 update ChatGLM slides (#37) 3 周前

- Season1.step_into_chat... update to two seasons (#36) 1 个月前
- Season2.step_into_llm update ChatGLM slides (#37) 3 周前
- .gitignore add requirements.txt 6 个月前
- LICENSE Initial commit 7 个月前
- README.md update to two seasons (#36) 1 个月前

README.md

简介 暂无描述 暂无标签 Python Text Jupyter Notebook Shell Markdown other 贡献者 (5) 全部

新的项目Fork

派生自 [mindspore-courses/step_into_llm](#)

项目名称 *

请输入中文、字母、数字和-., 最多100个字符。

项目路径

路径只允许字母、数字和-., 最多100个字符。

项目地址: https://openi.pcl.ac.cn/xing-yiren/step_into_llm.git

可见性 ☐ 将项目设为私有

无法更改派生项目的可见性。

项目描述

启智社区使用

6. 点击进入项目，选择云脑-调试任务-新建调试任务

The screenshot shows the Qizhi AI Collaboration Platform interface. The top navigation bar includes the Qizhi logo, 'AI协作平台 Powered by C²NET', and links for '个人中心', '项目', '数据集', '模型', '探索', and '论坛'. A search bar is also present. Below the navigation bar, a banner for 'step_into_llm' is displayed, along with statistics for '已关注' (1), '点赞' (0), and '派生' (0). The main content area features a tabbed interface with '代码', '任务', '合并请求', '数据集', '模型', '云脑', and '超算'. The '云脑' tab is highlighted with a red box and a red arrow labeled '1'. Below the tabs, there are buttons for '调试任务', '训练任务', '推理任务', '评测任务', and '在线推理'. A dropdown menu for '全部' is shown, with a red box and a red arrow labeled '2' pointing to the '新建调试任务' button. The main table lists tasks with columns for '任务名称', '状态', '创建时间', '集群/计算资源', '创建者', and '操作'. The table contains three rows of tasks, all with a status of 'STOPPED'. The bottom of the page shows pagination information: '共 3 条', '10条/页', and '前往 1 页'.

启智 AI协作平台 Powered by C²NET 个人中心 项目 数据集 模型 探索 论坛 搜索...

> 关于只保留最近5个调试任务的公告>>>

step_into_llm
派生自 mindspore-courses/step_into_llm

已关注 1 点赞 0 派生 0

代码 任务 0 合并请求 0 数据集 模型 云脑 超算 项目设置

调试任务 训练任务 推理任务 评测任务 在线推理

全部 新建调试任务

任务名称	状态	创建时间	集群/计算资源	创建者	操作
xing-202311101498978	STOPPED	29 分钟前	智算集群 NPU		再次调试 停止 删除
xing-202311101497702	STOPPED	1 天前	智算集群 NPU		再次调试 停止 删除
xing-202311081527372	STOPPED	3 天前	智算集群 NPU		再次调试 停止 删除

共 3 条 10条/页 1 前往 1 页

*平台仅留存近 30 天的调试、训练、推理、评测任务结果；超过 30 天的任务将不能下载结果和查看日志，且不能再次调试或训练。

启智社区使用

7. 基本信息中选择智算网络集群-昇腾NPU，镜像选择mindspore2.0.0_cann6.3_notebook；最后点击最下方的“新建任务”

基本信息：

算力集群 *

⊕ 启智集群

⊕ 智算网络集群(Beta)

计算资源 *

📁 英伟达GPU

📁 昇腾NPU

📁 燧原GCU

📁 寒武纪MLU

📁 海光DCU

📁 天数智芯G

📁 沐曦GPGPU

ⓘ 您当前排队位置是第 1 位

访问Internet *

☐ 否

☒ 是

?

资源规格 *

N

NPU: 1*Ascend 910, CPU: 24, 显存: 32GB, 内存: 96GB

1积分/时

? 资源说明

积分余额: 1159 积分, 预计可用 1159.00 小时

? 积分获取说明

参数设置：

代码分支 *

master

选择模型

选择模型文件

+ 选择模型

镜像 *

mindspore2.0.0_cann6.3_notebook

数据集

MindSpore1.10.1-Cann6.0.1-Python3.7-Euler2.8-MindOCR0.2.0

mindspore_1.10.1_notebook

Pytorch_1.11-cann6.0.1-py_3.7-euler_2.8.3

TensorFlow-1.15-cann_5.1.0-python3.7-euleros2.8-aarch64

paddlepaddle_0.0.0_nightly_cann6.0.1

mindspore_2.0.0_notebook

mindspore_1.8.1_notebook

mindspore2.0.0_cann6.3_notebook

+ 选择数据集

新建任务

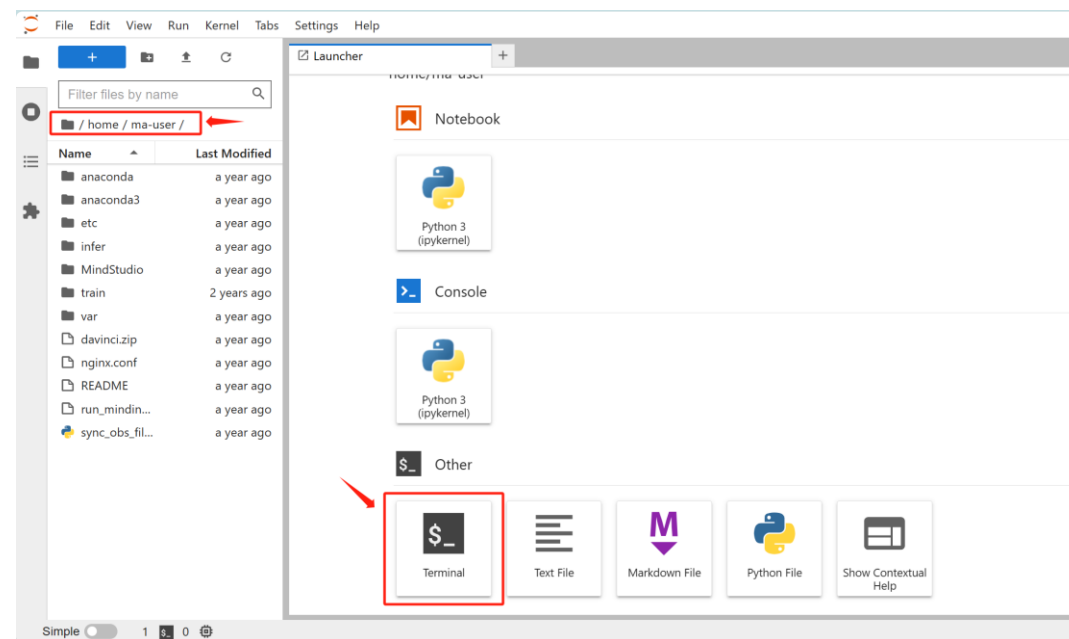
取消

启智社区使用

8. 在状态变为RUNNING后，即可点击“调试”进入调试任务



9. 进入任务后，导航至home/ma-user路径下，点击新建terminal，进行后续环境配置



环境准备

MindSpore 2.0版本, MindSpore Transformers dev版本

1. 安装MindSpore和MindSpore Transformers
 1. 参考: <https://www.mindspore.cn/install>
 2. 安装MindSpore Transformers:
 - `git clone -b dev https://gitee.com/mindspore/mindformers.git`
 - `cd mindformers`
 - `bash build.sh`
2. (如果使用启智算力) 更新opencv-python
 - `pip install opencv-python install "opencv-python-headless<4.3" -i https://pypi.tuna.tsinghua.edu.cn/simple`
3. `cd .. && git clone https://github.com/mindspore-courses/step_into_llm`
 - 使用启智社区的可以clone自己的个人仓
4. `cd step_into_llm/Season2.step_into_llm/o2.ChatGLM2/`
5. `python chatglm2_demo.py`

代码调用

可以通过如下代码调用 ChatGLM2-6B 模型来生成对话：

```
from mindformers import AutoTokenizer, AutoModel

tokenizer = AutoTokenizer.from_pretrained("glm2_6b")
model = AutoModel.from_pretrained("glm2_6b")

query = "你好"

prompted_inputs = tokenizer.build_prompt(query)
input_tokens = tokenizer([prompted_inputs])

outputs = model.generate(input_tokens["input_ids"], max_length=100)
response = tokenizer.decode(outputs)[0]
print(response)
```

问：你好

答：你好👋！我是人工智能助手 ChatGLM2-6B，很高兴见到你，欢迎问我任何问题。

ChatGLM3特性介绍

ChatGLM3-6B 是 ChatGLM3 系列中的开源模型，在保留了前两代模型对话流畅、部署门槛低等众多优秀特性的基础上，ChatGLM3-6B 引入了如下特性：

- **更强大的基础模型：** 基础模型ChatGLM3-6B-Base 采用了更多样的训练数据、更充分的训练步数和更合理的训练策略。具有在 10B 以下的基础模型中最强的性能。
- **更完整的功能支持：** **全新设计的 Prompt 格式**。原生支持**工具调用 (Function Call)**、**代码执行 (Code Interpreter)** 和 Agent 任务等复杂场景。
- **更全面的开源序列：** ChatGLM3-6B ，基础模型 ChatGLM3-6B-Base、长文本对话模型 ChatGLM3-6B-32K开源，亦允许免费商业使用。

全新Prompt格式

之前的 ChatGLM2模型中依旧使用了自然语言的prompt格式，通过用户问，模型答的方式进行对话生成。

[Round 1]

问：{用户输入}

答：{模型输出}

这样的设计会带来几个问题：

1. 进行多轮训练时操作比较复杂，不好控制loss mask，微调时效率会稍低一些
2. 无法进行原生工具调用，推理系统需要从生成的自然语言内容中解析出需要调用的工具，再进行调用

全新Prompt格式

ChatGLM3设计了四种special token，用于间隔开角色之间的对话内容。

- <|system|>：系统信息，设计上可穿插于对话中，但目前规定仅可以出现在开头
- <|user|>：用户，不会连续出现多个来自 <|user|> 的信息
- <|assistant|>：AI助手，在出现之前必须有一个来自 <|user|> 的信息
- <|observation|>：外部的返回结果，必须在 <|assistant|> 的信息之后

全新Prompt格式

special token之前遵循前不间隔后间隔的方式，即special token之前不用' \n' 隔开，后用' \n' 隔开，组合成的prompt形式如下：

<|system|>

You are ChatGLM3, a large language model trained by Zhipu.AI. Follow the user's instructions carefully.




Respond using markdown.<|user|>

Hello<|assistant|>

Hello, I'm ChatGLM3. What can I assist you today?<|user|>

今天天气怎么样？<|assistant|>

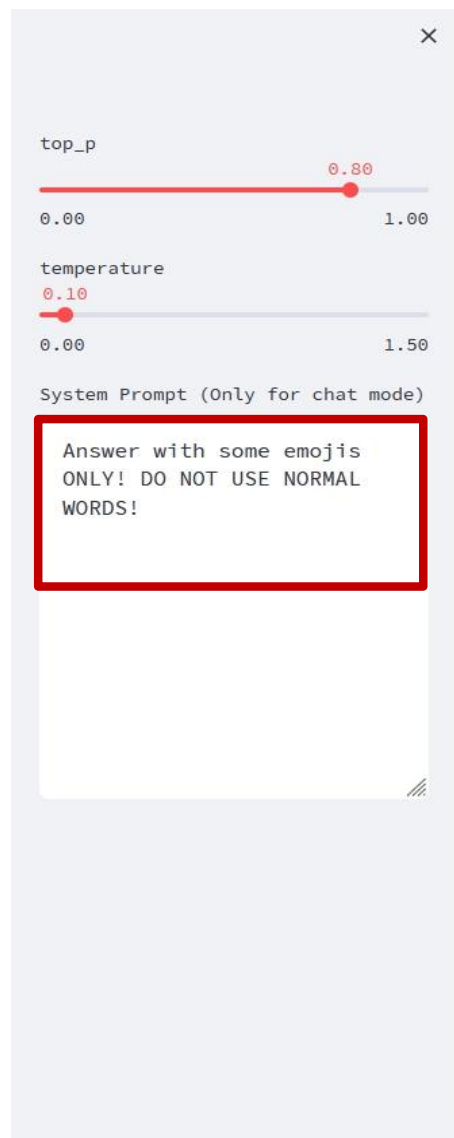
xxxxxx

-  为系统设定的角色，对应<|system|>
-  用户输入，对应<|user|>
-  模型输出，对应<|assistant|>

之前的推理中，我们往往通过<eos>或最大序列长度来识别生成的结束。在全新的prompt格式下，也可以通过识别<|user|>来定位生成句子的结束，表示AI助手对话结束，该进行用户输入了。

对话模式

对系统的角色设定,
为<|system|>后接的
内容



ChatGLM3 Demo

☒ Chat ☐ Tool ☐ Code Interpreter

你好啊

用户输入, 为<|user|>后接的内容

🏠 🌍

我现在很伤心, 嘤

🏠 😞 ❤️

鼓励鼓励我吧

🏠 😊 🍌

模型生成的输出, 为<|assistant|>
后生成的内容

谢谢你呜呜

🏠 ❤️ 🍌

Chat with ChatGLM3!

工具模式

构建system prompt

用list of dict的形式说明可调用的工具

- 输入1: location, 地理位置
- 输入2: unit, 温度单位

模型调取对应工具, 并进行参数输入

工具返回输出

```
<|system|>
Answer the following questions as best as you can. You have access to the following tools:
[
  {
    "name": "get_current_weather",
    "description": "Get the current weather in a given location",
    "parameters": {
      "type": "object",
      "properties": {
        "location": {
          "type": "string",
          "description": "The city and state, e.g. San Francisco, CA",
        },
        "unit": {"type": "string"},
      },
      "required": ["location"],
    },
  },
]

<|user|>
今天北京的天气怎么样?

<|assistant|>
好的, 让我们来查看今天的天气

<|assistant|>get_current_weather
```python
tool_call(location="beijing", unit="celsius")
```

<|observation|>
{"temperature": 22}

<|assistant|>
根据查询结果, 今天北京的气温为 22 摄氏度。
```

除了<|user|>以外, 模型也可以通过识别
<|observation|>来定位生成句子的结束, 表示AI助手对话结束, 该返回调用工具的输出了。

工具模式

可以在tool_registry.py中注册新的工具增强模型能力，具体形式为使用 @register_tool 装饰函数

```
@register_tool
def get_weather(
    city_name: Annotated[str, 'The name of the city to be queried', True],
) -> str:
    """
    Get the current weather for `city_name`
    """

    if not isinstance(city_name, str):
        raise TypeError("City name must be a string")

    key_selection = {
        "current_condition": ["temp_C", "FeelsLikeC", "humidity", "weatherDesc", "observation_time"],
    }

    import requests
    try:
        resp = requests.get(f"https://wttr.in/{city_name}?format=json")
        resp.raise_for_status()
        resp = resp.json()
        ret = {k: {_v: resp[k][_v] for _v in v} for k, v in key_selection.items()}
    except:
        import traceback
        ret = "Error encountered while fetching weather data!\n" + traceback.format_exc()

    return str(ret)
```

ChatGLM3 Demo

☐ Chat ☒ Tool ☐ Code Interpreter

☐ Manual mode ?

欧洲有哪些好玩的地方?

欧洲是一个非常有趣的地方，有许多好玩的地方。例如，可以前往法国的巴黎，欣赏埃菲尔铁塔和卢浮宫等著名景点；可以前往意大利的罗马，游览古罗马斗兽场和梵蒂冈等著名景点；可以前往德国的柏林，参观勃兰登堡门和柏林墙等著名景点。当然，欧洲还有许多其他好玩的地方，这只是其中的一部分。

查查巴黎的天气怎么样?

我需要调用天气预报工具来获取巴黎的天气信息。

Calling tool `get_weather` :

```
tool_call(city_name='巴黎')
```

Observation:

```
{'current_condition': {'temp_C': '12', 'FeelsLikeC': '10', 'humidity': '88'}}
```

根据天气预报工具返回的信息，巴黎现在的天气情况是：温度为12摄氏度，湿度为88%，天气状况为雨天。这些信息可以帮助您更好地了解巴黎的天气情况。

Chat with ChatGLM3!



代码解释器模式

拥有代码执行环境，所以可以进行绘制图表，数学运算等较为复杂的任务

这种模式下，`<|assistant|>`后的{metadata}只有一种情况：Interpreter，`<|observation|>`中通过[image]显示图片

```
<|user|>
帮我画一个爱心
<|assistant|>interpreter
```python
import numpy as np
import matplotlib.pyplot as plt

爱心形状的参数方程
def heart(t):
 x = 16 * np.sin(t) ** 3
 y = 13 * np.cos(t) - 5 * np.cos(2 * t) - 2 * np.cos(3 * t) - np.cos(4 * t)
 return x, y

t = np.linspace(0, 2 * np.pi, 1000)
x, y = heart(t)

plt.figure(figsize=(6, 6))
plt.plot(x, y, color='red')
plt.axis('equal')
plt.axis('off')
plt.show()
```
<|observation|>
```result
[Image]
```
```

ChatGLM 3 Demo

Chat Tool Code Interpreter

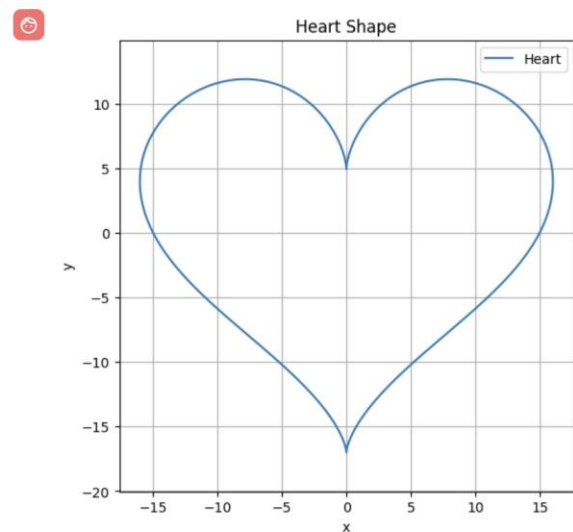
用 Python 画一个爱心

```
import matplotlib.pyplot as plt
import numpy as np

# 参数方程
t = np.linspace(0, 2 * np.pi, 1000)
x = 16 * np.sin(t) ** 3
y = 13 * np.cos(t) - 5 * np.cos(2 * t) - 2 * np.cos(3 * t) - np.cos(4 * t)

plt.figure(figsize=(6, 6))
plt.plot(x, y, label='Heart')
plt.title('Heart Shape')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.axis('equal')
plt.show()
```

Executing...



这是一个爱心形状的图形。如果您有其他要求或问题，请告诉我。

Chat with ChatGLM 3!

Thanks

课后小练习

1. 运行ChatGLM2推理部署代码，尝试和模型进行对话；
(如果出现回复不理想、内容比较奇怪的情况，也欢迎在代码仓issue中进行反馈)
2. 尝试ChatGLM3的各种模式对话

加入我们，一同学习大模型！

课程报名



课程资源获取
(课件+代码)



课程学习群

