目录

[MindSpore Serving服务接口说明（dev） 4](#_Toc157516484)

[1 配置文件 4](#_Toc157516485)

[1.1 model\_path 部分，输入serving启动所需要的所有文件路径 4](#_Toc157516486)

[1.1.1 prefill\_model: 4](#_Toc157516487)

[1.1.2 decode\_model: 4](#_Toc157516488)

[1.1.3 argmax\_model: 4](#_Toc157516489)

[1.1.4 topk\_model： 5](#_Toc157516490)

[1.1.5 prefill\_ini: 5](#_Toc157516491)

[1.1.6 decode\_ini: 5](#_Toc157516492)

[1.1.7 post\_model\_ini: 5](#_Toc157516493)

[1.2 model\_config部分，模型配置参数 6](#_Toc157516494)

[1.2.1 model\_name 6](#_Toc157516495)

[1.2.2 max\_generate\_length 6](#_Toc157516496)

[1.2.3 end\_token 6](#_Toc157516497)

[1.2.4 seq\_length 6](#_Toc157516498)

[1.2.5 vocab\_size 7](#_Toc157516499)

[1.2.6 prefill\_batch\_size 7](#_Toc157516500)

[1.2.7 decode\_batch\_size 7](#_Toc157516501)

[1.2.8 zactivate\_len 7](#_Toc157516502)

[1.2.9 model\_type 7](#_Toc157516503)

[1.2.10 page\_attention 8](#_Toc157516504)

[1.2.11 batching\_strategy 8](#_Toc157516505)

[1.2.12 current\_index 8](#_Toc157516506)

[1.3 serving\_config部分: 8](#_Toc157516507)

[1.3.1 agent\_ports 9](#_Toc157516508)

[1.3.2 start\_device\_id 9](#_Toc157516509)

[1.3.3 server\_ip 9](#_Toc157516510)

[1.3.4 server\_port 9](#_Toc157516511)

[1.4 pa\_config部分 9](#_Toc157516512)

[1.4.1 num\_blocks 9](#_Toc157516513)

[1.4.2 block\_size 10](#_Toc157516514)

[1.4.3 decode\_seq\_length 10](#_Toc157516515)

[1.5 tokenizer部分，自定义分词器 10](#_Toc157516516)

[1.6 模型入参部分 10](#_Toc157516517)

[2 自定义模块 11](#_Toc157516518)

[2.1自定义tokenizer 11](#_Toc157516519)

[2.2 自定义inputs 11](#_Toc157516520)

[3 client请求入参配置 12](#_Toc157516521)

[3.1 流式推理 12](#_Toc157516522)

[3.2 非流式推理 13](#_Toc157516523)

[3.3 使能Sample推理 15](#_Toc157516524)

[3.4 关闭Sample推理 15](#_Toc157516525)

[4 Serving推理API接口 16](#_Toc157516526)

[4.1 generate\_answer(request\_id, \*\*params) 16](#_Toc157516527)

[5 Serving状态获取接口 17](#_Toc157516528)

[5.1 获取当前时刻的batch\_size大小： 17](#_Toc157516529)

[5.2 获取当前时刻请求队列里还在等待推理的请求数 17](#_Toc157516530)

[6 错误码 17](#_Toc157516531)

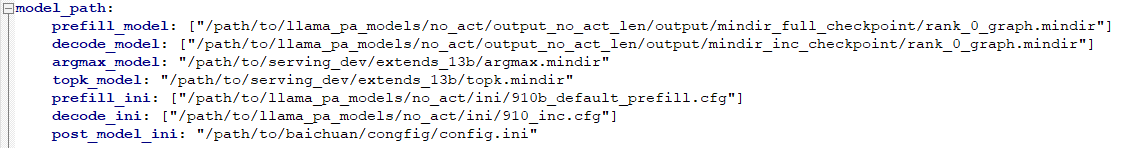
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 修订时间 | 修订人 | 修订类型 | 修订章节 | 修订内容 |
| A0.1 | 2024-01-29 | 石子洋 | A | 123456 | serving支持yaml配置，基本接口内容 |

MindSpore Serving服务接口说明（dev）

# 配置文件

配置文件放在serving-gitee/serving/configs目录下，当前提供双动态的llama2-70b和internLM模型的yaml，新增模型yaml的话需要按照以下说明进行填写。这里以llma2\_13b\_pa\_no\_act.yam为例

## model\_path 部分，输入serving启动所需要的所有文件路径



## prefill\_model:

* 数据类型：List[str]
* 说明：全量模型的权重mindir路径，List长度为模型的切分数；
* 类型：必要参数，用户必须传入

## decode\_model:

* 数据类型：List[str]
* 说明：增量模型的权重mindir路径，List长度为模型的切分数；
* 类型：必要参数，用户必须传入

## argmax\_model:

* 数据类型：str
* 说明：argmax模型的权重mindir路径；
* 类型：必要参数，用户需通过post\_sampling\_model.py脚本导出

## topk\_model：

* 数据类型：str
* 说明：topk模型的权重mindir路径；
* 类型：必要参数，用户需通过post\_sampling\_model.py脚本导出

## prefill\_ini:

* 数据类型：List[str]
* 说明：全量模型的配置文件路径，具体ini配置参考lite模型的配置说明, 多个档位的配置用 ’ , ’分开；
* 类型：必要参数，用户必须传入；

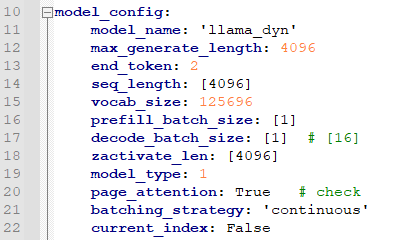
## decode\_ini:

* 数据类型：List[str]
* 说明：增量模型的配置文件路径，具体ini配置参考lite模型的配置说明, 多个档位的配置用 ’ , ’分开；
* 类型：必要参数，用户必须传入；

## post\_model\_ini:

* 数据类型：str
* 说明：后处理模型的配置文件路径，具体ini配置参考lite模型的配置说明；
* 类型：必要参数，用户必须传入；

## model\_config部分，模型配置参数



## model\_name

* 数据类型：str
* 说明：模型名称, e.g. internlm\_\*, llama\_\*；
* 类型：必要参数，用户必须传入

## max\_generate\_length

* 数据类型：int
* 说明：如果开启PagedAttention，则设置该参数为seq\_len（导出模型的最大seq length） / block\_size
* 类型：必要参数，用户必须传入

## end\_token

* 数据类型：int
* 说明：分词器中eos token（结束符）的id；
* 类型：必要参数，用户自定义，根据实际分词器词表情况填写；

## seq\_length

* 数据类型：List[int]
* 说明：适配prefill seq length分档， list为空时表示纯动态，设置一个档位时为静态seq，设置多个档位，表示动态seq分档
* 类型：配合seq\_type字段使用

## vocab\_size

* 数据类型：int
* 说明：分词器的词表长度；
* 类型：必要参数，用户自定义，根据实际分词器词表情况填写；

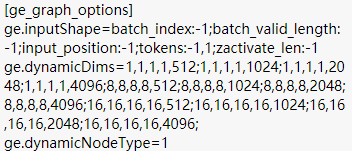
## prefill\_batch\_size

* 数据类型：List[int]
* 说明：适配prefill动态分档，设置一个值时为静态batch模型；
* 类型：必要参数，用户必须传入, 根据实际模型的配置情况填写

## decode\_batch\_size

* 数据类型：List[int]
* 说明：适配decode动态分档，设置一个值时为静态batch模型；
* 类型：必要参数，用户必须传入, 根据实际模型的配置情况填写

## zactivate\_len

* 数据类型：List[int]
* 说明：适配act len优化
* 类型：必要参数，用户自定义，根据实际部署的模型决定；
* 使用示例： [512, 1024, 2048, 4096] 配合ini使用，decode.ini增加

## model\_type

* 数据类型：str
* 说明：模型类型，用于构造输入使用，针对动态seq, 静态seq输入设置为”static”，动态设置为”dyn”；
* 类型：必要参数，用户自定义，根据实际部署的模型决定；

## page\_attention

* 数据类型：bool
* 说明：当启动PagedAttention算法时，将参数设置为True
* 类型：可选参数，默认为False；

## batching\_strategy

* 数据类型：str
* 说明：组batch的策略，当前支持static batch和continuous batching两种模式，static batch该参数设置为’static’， continuous batching设置为’continuous’
* 类型：可选参数，默认为’continuous’；

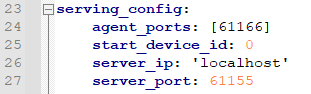
## current\_index

* 数据类型：bool
* 说明：ft分支旧模型的输入参数，dev分支没有此参数
* 类型：可选参数，默认为’ False’；

## pad\_token\_id

* 数据类型: int
* 说明：模型做输入如果要做padding，padding的token\_id，根据模型的分词器配置；
* 类型：必须参数，默认为0（llama的pad\_token\_id）；

## serving\_config部分:



## agent\_ports

* 数据类型：List[int]
* 说明：模型并行下，每份模型对应一个socket server进行管理，需要提供socket server的port。size需要和prefill\_model 一致
* 类型：必要参数

## start\_device\_id

* 数据类型：int
* 说明：使用的NPU的起始device id
* 类型：可选，默认为0卡

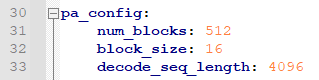
## server\_ip

* 数据类型：str
* 说明：serving server侧FastAPI服务的ip
* 类型：可选，默认为’localhost’

## server\_port

* 数据类型：int
* 说明：serving server侧FastAPI服务的port
* 类型：必要参数

## pa\_config部分



当model\_config中的page\_attention为True时，需要根据模型设置此项

## num\_blocks

* 数据类型：int
* 说明： PA预申请内存块总量
* 类型：必要参数

## block\_size

* 数据类型：int
* 说明：PA单内存块容纳slot（token）数量
* 类型：必要参数

## decode\_seq\_length

* 数据类型：int
* 说明：用于计算每个请求的最大内存块申请量，公式：max\_num\_block\_per\_seq = decode\_seq\_length /block\_size
* 类型：必要参数

## tokenizer部分，自定义分词器

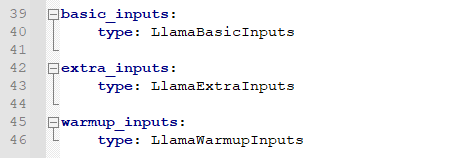


type: 自定义分词器名称

vocab\_file: 模型对应的tokenizer.model

自定义分词器参考[自定义tokenizer](#_自定义tokenizer)

## 模型入参部分



basic\_inputs 包括[input\_ids, current\_index, init\_reset, valid\_length, decode\_batch\_index]

extra\_inputs 包括所有除了上述参数以外的模型入参，如act\_len

warmup\_inputs 模型warmup所需要的入参

输入自定义部分参考[自定义inputs](#_自定义inputs)

# 2 自定义模块

## 2.1自定义tokenizer

在mindspore\_serving/models/tokenizer目录创建xxx\_tokenizer.py

* 定义class XxxTokenizer, 类的初始化参数第一个必须为vocab\_file,然后[调用@Registers.TOKENIZER.register()](mailto:调用@Registers.TOKENIZER.register())进行注册

## 自定义inputs

在mindspore\_serving/models/model\_inputs目录创建xxx\_inputs.py

* 定义 class XxxBasicInputs(BaseBasicInputs)，入参为(input\_ids, current\_index, init\_reset, batch\_valid\_length, \*args)，[用@Registers.BASIC\_INPUTS.register()](mailto:用@Registers.BASIC_INPUTS.register())进行注册
* 定义 class XxxExtraInputs(BaseExtraInputs)，入参为(input\_ids, current\_index, init\_reset, is\_prefill, valid\_length, \*\*kwargs)。注意extra inputs必须得通过入参计算而来，比如act\_len就是通过valid\_length计算来的，[用@Registers.EXTRA\_INPUTS.register()](mailto:用@Registers.EXTRA_INPUTS.register())进行注册
* 定义 class XxxWarmupInputs(BaseWarmupInputs)，入参为(seq\_length, batch\_size, full\_model, valid\_length=None, \*\*kwargs)，[用@Registers.WARMUP\_INPUTS.register()](mailto:用@Registers.WARMUP_INPUTS.register())进行注册

# 3 client请求入参配置

## 流式推理

每次推理请求结果按照单个单词依次返回，最后支持返回所有生成单词。

URI

* URI格式

POST /models/{mode\_type}/generate\_stream

* 参数描述

| 参数 | 是否必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| inputs | 是 | String | 用户输入提示 |
| parameters | 是 | Json | 生成参数 |

请求

* 请求样例

POST https://endpoint/models/{mode\_type}/generate\_stream

{

"inputs":"what is Monetary Policy?",

"parameters":{

"max\_new\_tokens":20,

"temperature":1,

"top\_k":1,

"top\_p":1,

"do\_sample":"False",

"repetition\_penalty":1,

"return\_full\_text":"True"

}

* 参数说明

| 参数 | 是否必选 | 类型 | 描述 | 取值 |
| --- | --- | --- | --- | --- |
| do\_sample | 否 | bool | 后处理是否采样 | 默认值为False |
| max\_new\_tokens | 否 | int | 最大生成数量 | 默认值为300 |
| repetition\_penalty | 否 | float | 重复单子惩罚参数 | 默认值为1.0 |
| temperature | 否 | float | 后处理随机性参数 | 默认值为1.0 |
| top\_k | 否 | int | 后处理top\_k参数 | 默认值为3，如果请求给的top\_k < 0，调整为0 |
| top\_p | 否 | float | 后处理top\_p参数 | 默认值为1.0，如果请求给的top\_p < 0.01，调整为0.01,top\_p > 1.0 调整为1.0 |
| return\_full\_text | 否 | bool | 最后是否返回所有生成 | 默认值为True |
| return\_protocol | 否 | str | 默认使用sse协议返回,其他情况均使用fastapi的流式返回（StreamingResponse） | 默认值为’sse’ |

响应

* 流式中间响应样例

data:{

"token": {"text": "xx"}

}

1. 流式最后响应样例

data:{

"generated\_text": "xxxxxxxxxxxxxxxxxxxxxxxx"

}

## 非流式推理

功能介绍

每次推理请求结果一次返回所有生成单词。

URI

* URI格式

POST /models/{mode\_type}/generate

* 参数描述

| 参数 | 是否必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| inputs | 是 | String | 用户输入提示 |
| parameters | 是 | Json | 生成参数 |

* Parameters字段描述

请求

* 请求样例

POST https://endpoint/models/{mode\_type}/generate

{

"inputs":"what is Monetary Policy?",

"parameters":{

"max\_new\_tokens":20,

"temperature":1,

"top\_k":1,

"top\_p":1,

"do\_sample":"False",

"repetition\_penalty":1

}

* 参数说明

| 参数 | 是否必选 | 类型 | 描述 | 取值（同流式请求） |
| --- | --- | --- | --- | --- |
| do\_sample | 否 | bool | 后处理是否采样 | - |
| max\_new\_tokens | 否 | int | 最大生成数量 | - |
| repetition\_penalty | 否 | float | 重复单子惩罚参数 | - |
| temperature | 否 | float | 后处理随机性参数 | - |
| top\_k | 否 | int | 后处理top\_k参数 | - |
| top\_p | 否 | float | 后处理top\_p参数 | - |

响应

* 响应样例

{

"generated\_text": "xxxxxxxxxxxxxxxxxxxxxxxx"

}

## 使能Sample推理

参数设置

“do\_sample”设置为“True”

请求

* 请求样例

POST https://endpoint/models/{mode\_type}/generate

{

"inputs":"what is Monetary Policy?",

"parameters":{

"max\_new\_tokens":20,

"temperature":1,

"top\_k":1,

"top\_p":1,

"do\_sample":"True",

"repetition\_penalty":1

}

## 关闭Sample推理

参数设置

“do\_sample”设置为“False”

请求

* 请求样例

POST https://endpoint/models/{mode\_type}/generate

{

"inputs":"what is Monetary Policy?",

"parameters":{

"max\_new\_tokens":20,

"do\_sample":"False",

}

# Serving推理API接口

## generate\_answer(request\_id, \*\*params)

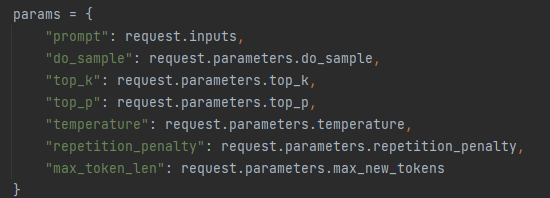
使用方法：

1、实例化llm\_server = LLMServer()

2、准备好请求request\_id及入参，params;

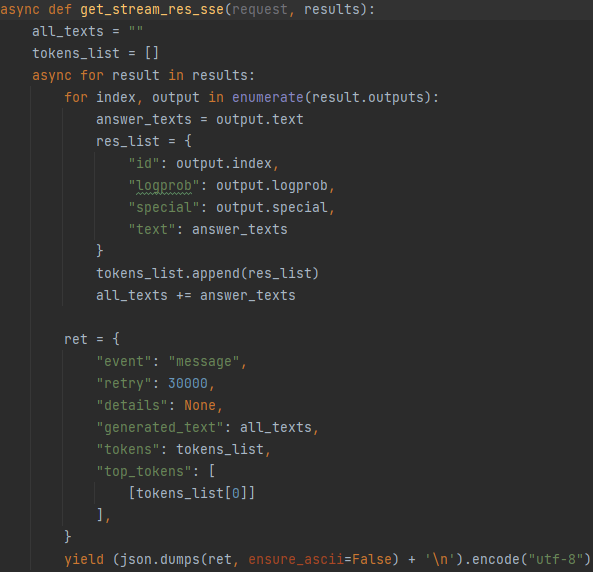
3、results = llm\_server.generate\_answer(request\_id, \*\*params)

4、params如下：



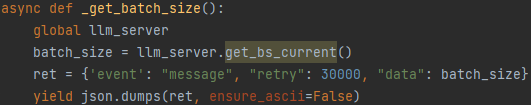
5、返回值为async\_generator对象；

6、使用API方式调用的，可以将async\_generator对象解析成流式的或非流式的结果，参照代码clinet/server\_app\_post.py, 以下是以sse协议封装的用例。



# Serving状态获取接口

## 获取当前时刻的batch\_size大小：



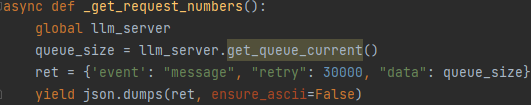
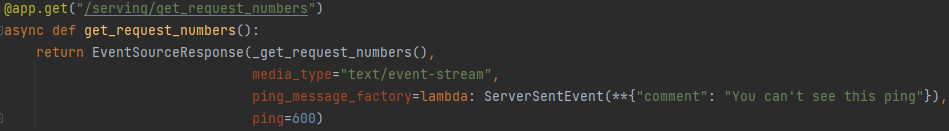


以sse的形式提供，同流式和非流式的接收方法：



## 获取当前时刻请求队列里还在等待推理的请求数

同5.1

以sse的形式提供，同流式和非流式的接收方法：



# 错误码

目前包含的错误码总结如下表：

|  |  |
| --- | --- |
| 错误码返回值 | 错误原因 |
| 202 | 分词长度校验，超过模型输入 |
| 203 | 解分词超过模型支持范围 |
| 204 | 入参超过边界值，返回警告和默认值 |
| 301 | 请求队列超过了设定阈值，返回警告 |