登临 OpenAI-compatible API Server 发布包使用说明

DL-DG/SW-057A-03

2025-01-13

商标和许可

和其它苏州登临科技有限公司的其它登临科技的图标为苏州登临科技有限公司的商标。本手册中提及的所有其他商标均为其各自所有者的财产。

通知

所购买的产品、服务和特性由苏州登临科技有限公司与客户签订的合同规定。本文件中描述的所有或部分产品、服务和特性可能不在采购范围或使用范围内。除非合同中另有规定，本文件中的所有声明、信息和建议均按“原样”提供，无任何明示或暗示的保证或陈述。
本手册中的信息如有更改，恕不另行通知。本文件在编制过程中已尽一切努力确保内容的准确性，本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

苏州登临科技有限公司

苏州工业园区扬富路11号南岸新地一期商务楼栋5号1101室，江苏，中国

http://www.denglin.ai

email：support@denglin.ai

更新历史

版本	更新描述
03	添加Driver SDK编译进Triton Server支持。
02	添加Rerank使用说明。 添加启动参数。
01	第一次发布。

目录

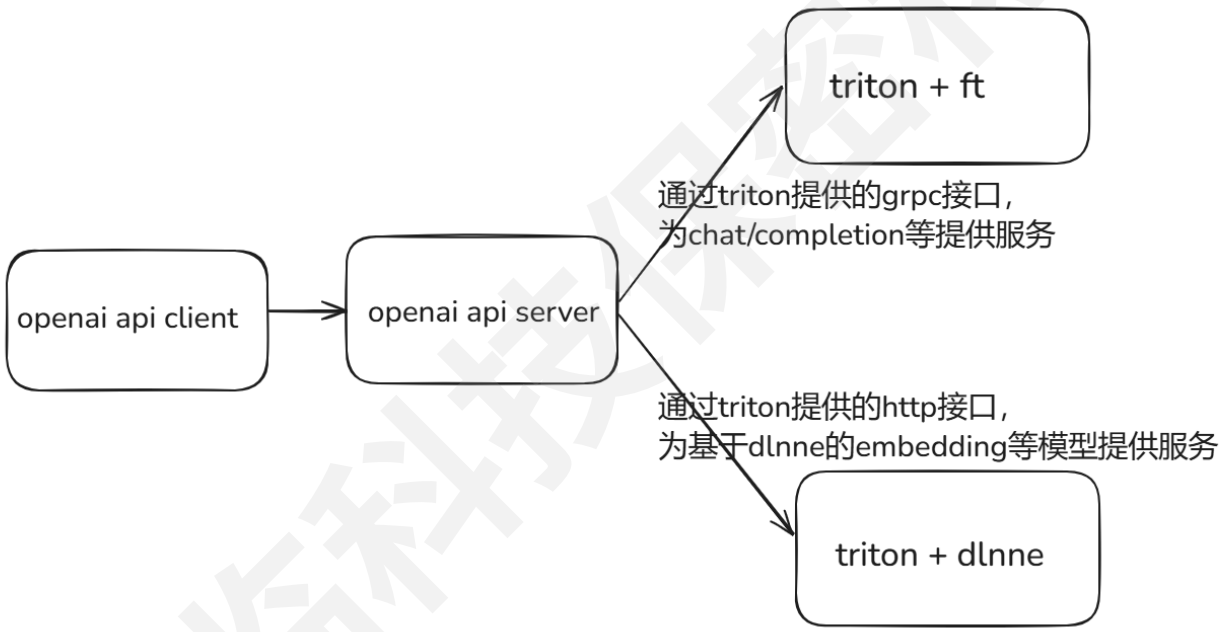
1 简介

OpenAPI Specification定义了一套与语言无关的HTTP APIs，为了方便用户使用这套API，登临提供openai_api项目作为HTTP server用来接收请求。openai_api后端连接登临Triton作为推理引擎。
目前openai_api支持的API请求如下：

- /v1/models
- /v1/completions
- /v1/chat/completions
- /v1/embeddings
- /v1/rerank

对于openai_api，基于Triton的fastertransformer(ft) backend提供了聊天对话相关能力，基于Triton的dlNNE backend提供了embedding相关能力。

登临openai_api整体架构如下图：



2 安装

2.1 安装登临 Triton

因openai_api后端依赖登临Triton，所以首先要搭建登临Triton的运行环境，具体安装方法请参考《登临 Hamming V2 Triton 发布包使用说明》。

2.2 安装 Python 依赖

登临openai_api项目使用Python语言开发，在使用前还需要安装Python依赖包（请基于登临Triton提供的镜像来运行该程序）。 `openai_api/requirements.txt` 中记录所有的依赖包，可以直接使用Pip安装，命令如下：

```
pip install -r requirements.txt
```

2.3 启动镜像

登临提供openai_api的镜像，在该镜像中已经安装了Python依赖，项目源代码位于 `/opt/openai_api` 目录。

可以使用下面的命令来启动镜像（相关参数可以参考章节3 启动）。

```
docker run -it --network=host --rm -v /LocalRun:/LocalRun <name:version>
/opt/openai_api/openai_api_cli.py --host 0.0.0.0 --port 5100
```

对于config文件以及config中的配置文件，需要挂载到容器中，并且路径配置为该文件在容器中的路径。

3 启动

openai_api可直接启动（默认参数），也可自行设置参数。

使用默认参数启动openai_api，命令如下：

```
python ${YOUR_PATH}/openai_api/openai_api_cli.py
```

3.1 参数

openai_api提供了可选参数，用户可以在启动时设置参数的值，例如设置端口号为8000：

```
python ${YOUR_PATH}/openai_api/openai_api_cli.py --port 8000
```

查询所有参数信息：

```
python ${YOUR_PATH}/openai_api/openai_api_cli.py --help
```

openai_api参数如下表所示：

参数	用途
--help	帮助信息
--host	主机地址，默认loalhost
--port	端口号
--verbose	输出更多后端关于LLM和Triton的信息
--uvicorn-log-level	设置模块uvicorn的日志级别，包括{debug,info,warning,error,critical,trace}
--config-file	描述模型信息的JSON配置文件路径

3.2 日志

openai_api除了显示模块uvicorn的日志信息，还可以显示自身的日志。

通过环境变量LOG_LEVEL设置openai_api日志级别，可以设置的级别包括：

- CRITICAL
- FATAL
- ERROR
- WARNING
- INFO
- DEBUG
- NOTSET

不做任何设置的话，默认值为INFO。

关于更多LOG_LEVEL的信息，请参考Python的logging模块 https://docs.python.org/3/library/logging.html 。

3.3 配置文件

通过JSON配置文件提供所有模型信息，并在启动时通过 `--config-file` 参数指定配置文件。发送请求时只能使用配置文件中的模型。

默认配置文件 `${YOUR_PATH}/openai_api/openai_api/model_config.json` 。

配置文件信息如下：

类别	说明
model_infos	模型参数有关信息，例如start_id
LLM模型信息	name：模型的名称
	url：登临Triton server监听的gRPC地址
	path：huggingface模型所在的文件夹
	chat_template：模板文件
embedding模型信息	name：模型的名称
	url：登临Triton server监听的HTTP地址
	path：huggingface模型所在的文件夹
	max_tokens：生成的token数量最大值
rerank模型信息	name：模型的名称
	url：登临Triton server监听的HTTP地址
	path：huggingface模型所在的文件夹
	max_tokens：生成的token数量最大值
	input_config：输入配置，定义了模型所需的输入字段及其对应的键名
	output_config：输出配置，定义了模型输出的字段及其处理方式

配置文件举例：

```
{
  "model_infos": [
    {
      "name": "chatglm2-6b",
      "start_id": -1,
      "end_id": -2,
      "max_len": 2048,
```

```
      "bad_words": [
        [7768, 3908],
        [1, 2]
      ],
      "stop_words": [[2], [2]],
      "temperature": 1,
      "repetition_penalty": 1,
      "mask_positions": 1
    }
  ],
  "llm": [
    {
      "name": "chatglm2-6b",
      "url": "127.0.0.1:8001",
      "path": "/LocalRun/void_main_ft_with_llama/chatglm2-6b",
      "chat_template": "chatglm2.jinja"
    }
  ],
  "embedding": [
    {
      "name": "bge_large_zh_v1_5",
      "url": "127.0.0.1:8000",
      "path": "/LocalRun/model_zoo/bge-large-zh-v1.5",
      "max_tokens": 512
    }
  ],
  "rerank": [
    {
      "name": "bge-reranker",
      "url": "127.0.0.1:8000",
      "path": "/LocalRun/model_zoo/bge-reranker",
      "max_tokens": 512,
      "input_config": {
        "input_ids": "input_ids",
        "attention_mask": "attention_mask"
      },
      "output_config": {
        "name": "logits",
        "process": "normalize"
      }
    }
  ]
}
```

4 发起请求

可以使用下面的命令发起一个API请求。

```
curl http://localhost:5100/v1/chat/completions
    -H "Content-Type: application/json"
    -d '{
    "model": "llama2-7b", "stream":false,
    "messages": [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"}
        ]
    }'
```

此请求查询llama2-7b模型，会收到类似以下内容的响应：

```
data: {
        "id": "cmpl-d7f644b0de8a4cdca81b36d1c67cfd4d",
        "object": "chat.completion.chunk",
        "created": 1712479703,
        "model": "llama2-7b",
        "choices": [{
                "index": 0,
                "delta": {
                        "role": "user"
                },
                "logprobs": null,
                "finish_reason": null
        }]
}

data: {
        "id": "cmpl-d7f644b0de8a4cdca81b36d1c67cfd4d",
        "object": "chat.completion.chunk",
        "created": 1712479703,
        "model": "llama2-7b",
        "choices": [{
                "index": 0,
                "delta": {
                        "content": ""
                },
                "finish_reason": "stop",
                "stop_reason": "stop"
        }],
        "usage": {
                "prompt_tokens": 44,
                "total_tokens": 44,
                "completion_tokens": 0
        }
}
```

```
data: [DONE]
```

可以看到 `finish_reason` 返回值是 `stop`，这表明API返回了模型生成的完整Completion。

4.1 Models 命令

Models命令用于列出当前可用的模型，并提供有关每个模型的基本信息（例如所有者和可用性）。

```
GET http://localhost:5100/v1/models
```

发送请求：

```
curl http://localhost:5100/v1/models
```

响应：

```
{
        "object": "list",
        "data": [{
                "id": "baichuan2-13b",
                "object": "model",
                "created": 1712043451,
                "owned_by": "openai_api",
                "root": "baichuan2-13b",
                "parent": null,
                "permission": [{
                        "id": "modelperm-b1b1dbfa238c4bc08c7e189c0163a5a1",
                        "object": "model_permission",
                        "created": 1712043451,
                        "allow_create_engine": false,
                        "allow_sampling": true,
                        "allow_logprobs": true,
                        "allow_search_indices": false,
                        "allow_view": true,
                        "allow_fine_tuning": false,
                        "organization": "*",
                        "group": null,
                        "is_blocking": false
                }]
        }]
}
```

4.2 Completions 命令

创建Completions：

```
POST http://localhost:5100/v1/completions
```

发送请求：

```
curl http://localhost:5100/v1/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "baichuan2-13b",
    "stream": "true",
    "prompt": "Once upon a time",
    "max_tokens": 41,
    "temperature": 0.5
  }'
```

响应：

```
......

data: {
        "id": "cmpl-ae36dfac590a4466b59e7faa0d47df3f",
        "object": "text_completion",
        "created": 1712117714,
        "model": "baichuan2-13b",
        "choices": [{
                "index": 0,
                "text": ", there was a little girl named Sarah. Sarah loved to read books,
and she especially loved fairy tales. One day, she came across a book called \"The
Enchanted Forest.\" The book was",
                "logprobs": null,
                "finish_reason": "stop",
                "stop_reason": null
        }],
        "usage": null
}

data: {
        "id": "cmpl-ae36dfac590a4466b59e7faa0d47df3f",
        "object": "text_completion",
        "created": 1712117714,
        "model": "baichuan2-13b",
        "choices": [{
                "index": 0,
                "text": "",
                "logprobs": null,
                "finish_reason": "stop",
                "stop_reason": null
        }],
        "usage": {
```

```
                "prompt_tokens": 4,
                "total_tokens": 45,
                "completion_tokens": 41
        }
}

data: [DONE]
```

4.3 Chat 命令

为给定的聊天对话创建模型响应。

```
POST http://localhost:5100/v1/chat/completions
```

发送请求：

```
curl http://localhost:5100/v1/chat/completions
    -H "Content-Type: application/json"
    -d '{
    "model": "baichuan2-13b", "stream":false,
    "messages": [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"}
        ]
    }'
```

响应：

```
{
        "id": "cmpl-176257166c76458d8c7ad6bf98900157",
        "object": "chat.completion",
        "created": 1712120769,
        "model": "baichuan2-13b",
        "choices": [{
                "index": 0,
                "message": {
                        "role": "user",
                        "content": "The 2020 World Series was won by the Los Angeles
Dodgers, who defeated the Tampa Bay Rays in a six-game series. This was the Dodgers' first
World Series victory since 1988."
                },
                "logprobs": null,
                "finish_reason": "stop",
                "stop_reason": "stop"
        }],
        "usage": {
                "prompt_tokens": 20,
                "total_tokens": 65,
                "completion_tokens": 45
```

```
        }
}
```

4.4 Embeddings 命令

Embeddings命令用于将文本转换为向量表示（即嵌入），这些向量可以用于各种自然语言处理任务，如文本相似度计算、聚类、分类等。

```
POST http://localhost:5100/v1/embeddings
```

发送请求：

```
curl http://localhost:5100/v1/embeddings \
  -H "Content-Type: application/json" \
  -d '{
    "input": "The food was delicious and the waiter...",
    "model": "bge_large_zh_v1_5",
    "encoding_format": "float"
  }'
```

响应：

```
{
        "object": "list",
        "data": [{
                "object": "embedding",
                "embedding": [0.05523403361439705, 0.01236140076071024,
                 0.005706621799618006,-0.031625982373952866,......],
                "index": 0
        }],
        "model": "bge_large_zh_v1_5"
}
```

4.5 Rerank 命令

Rerank命令用于排序。对给定的一段文本输入查询语句，经过推理后输出重新排序的文件对象内容列表，按概率从大到小的结果排序（relevance_score分越高，概率越大）。

- model：string类型，Rerank模型名称。当前支持bge-reranker。
- query：string类型，搜索查询语句。
- documents：Array of string类型。需要重新排序的文件对象内容列表。
- top_n：string或者int64，可选，要返回的最相关文件内容的数量，默认为需要重新排序的文件对象内容列表`documents`的长度。
- return_documents：bool类型，可选，表示是否返回文档内容。默认为`false`。

值为 `false` 表示不返回文档内容 {index， relevance_score} 。值为 `true` 表示返回文档内容 {index, text, relevance_score} 。

```
POST http://localhost:5100/v1/rerank
```

发送请求：

```
curl http://localhost:5100/v1/rerank \
-H "Content-Type: application/json" \
-d '{
    "query": "What principles does the management of finished oil circulation in Jiangsu Province follow?",
    "documents": [
        "Article 1 In order to strengthen the management of finished oil circulation, standardize finished oil business operations, maintain market order for finished oil, protect the legitimate rights and interests of finished oil operators and consumers, according to relevant laws, regulations, and State Council decisions, combined with the actual conditions of this province, this law is formulated.",
        "Article 12 The market supervision and administration department shall push the information of enterprises with new registered business scopes involving finished oil wholesale and storage through the government shared platform to the commerce department. The commerce department shall promptly push information found during industry regulation that is not within its regulatory responsibilities to the competent authorities concerned.",
        "Article 3 The management of finished oil circulation follows the principles of legal management, ensuring supply, safe development, and fair competition.",
        "Article 7 The finished oil industry association shall establish and improve industry operating self-discipline norms, self-regulation conventions, and professional ethics standards, regulate member behavior, and play a role in rights protection, dispute resolution, and industry self-regulation."
    ],
    "model": "bge-reranker",
    "encoding_format": "float",
    "return_documents": true,
    "top_n": 3
}'
```

响应：

```
{
  "object": "list",
  "data": [
    {
      "object": "rerank",
      "document": {
        "text": "Article 3 The management of finished oil circulation follows the principles of legal management, ensuring supply, safe development, and fair competition."
      },
      "index": 2,
      "relevance_score": 0.9997203946113586
    },
    {
```

```
      "object": "rerank",
      "document": {
        "text": "Article 1 In order to strengthen the management of finished oil
circulation, standardize finished oil business operations, maintain market order for
finished oil, protect the legitimate rights and interests of finished oil operators and
consumers, according to relevant laws, regulations, and State Council decisions, combined
with the actual conditions of this province, this law is formulated."
      },
      "index": 0,
      "relevance_score": 0.956419050693512
    },
    {
      "object": "rerank",
      "document": {
        "text": "Article 7 The finished oil industry association shall establish and
improve industry operating self-discipline norms, self-regulation conventions, and
professional ethics standards, regulate member behavior, and play a role in rights
protection, dispute resolution, and industry self-regulation."
      },
      "index": 3,
      "relevance_score": 0.07036340236663818
    }
  ],
  "model": "bge-reranker",
  "usage": {
    "prompt_tokens": 316,
    "total_tokens": 316,
    "completion_tokens": 0
  }
}
```

5 添加新模型

添加新的模型需要分别在登临Triton和openai_api中添加配置信息。

首先，要在登临Triton中完成新模型的部署，确保登临Triton能使用新的模型进行推理。具体配置方法请参考《登临Triton发布包使用说明》。

其次，在openai_api中添加模型配置信息，具体步骤如下。

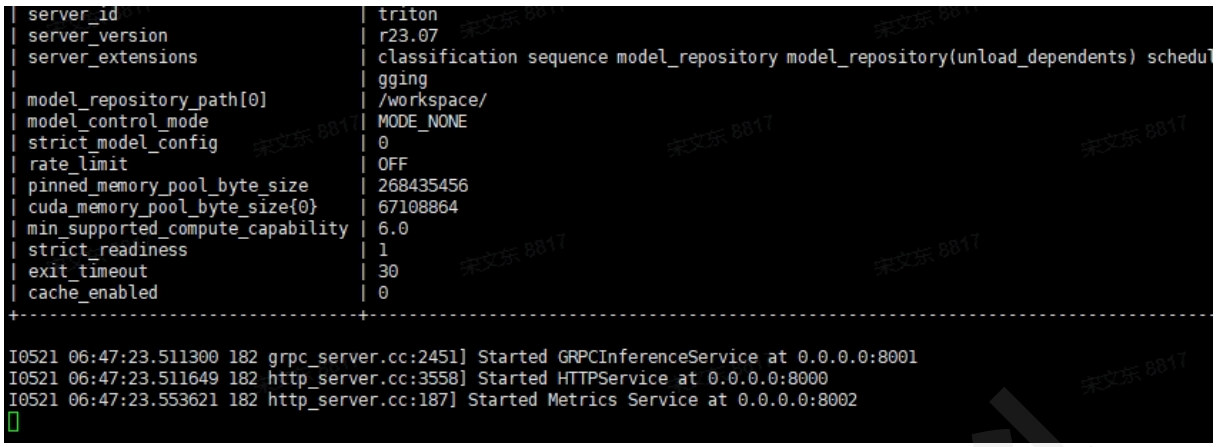
1. 将模型具体信息添加到JSON配置文件中。配置选项信息可以参考openai_api提供的默认配置文件 `${YOUR_PATH}/openai_api/openai_api/model_config.json`
   配置文件中，三个主要部分如下：

   ```
   {
   "model_infos": [...],
   "llm": [...],
   "embedding": [...]
   }
   ```

   model_infos中填入模型本身具体的信息，llm和embedding对应两种模型类别。
   llm和embedding 配置项说明：

   ```
   "llm": [
    {
      "name": "chatglm2-6b",
      "url": "10.28.31.48:8001",
      "path": "/LocalRun/void_main_ft_with_llama/chatglm2-6b",
      "chat_template": "chatglm2.jinja"
    }
   ],
   "embedding": [
    {
      "name": "bge_large_zh_v1_5",
      "url": "10.28.31.48:8000",
      "path": "/LocalRun/model_zoo/bge-large-zh-v1.5",
      "max_tokens": 512
    }
   ]
   ```

   - name：模型名字。
   - url：登临Triton所监听的地址，要与登临Triton一致。登临Triton启动成功后会打印相关信息如下：



- path：openai_api 访问的模型所在路径。
- 如果模型属于LLM，需要提供chat template。
  - 由于各个聊天模型使用的训练数据格式不同，如果在推理时使用的格式与模型训练时使用的格式不同，可能会引起性能下降等问题，因此匹配训练期间使用的格式非常重要。
  - 使用chat template保存模型训练时使用的聊天格式。使用Jinja模板语言描述这个chat template，并在JSON配置文件中配置该项.jinja文件路径。关于chat template和.jinja文件，详细内容请参考https://huggingface.co/docs/transformers/en/chat_templating。
- 如果模型属于embedding，可以设置`max_tokens`。

最后，启动openai_api，通过`--config-file`参数指向JSON配置文件。