AI 大模型基础环境搭建

搭建

- **大模型基础环境**

大模型基础环境通常会依赖以下 package：

- PyTorch：PyTorch 是一个用于深度学习的开源库，由 Facebook AI Research 开发。PyTorch 广泛用于自然语言处理、计算机视觉和语音识别等领域。
- torchvision：torchvision 是一个用于计算机视觉任务的库，提供了图像和视频处理的各种功能。torchvision 基于 PyTorch 构建，方便在 PyTorch 项目中使用。
- torchaudio：torchaudio 是一个用于音频处理的库，提供了对音频信号进行处理的函数和预训练模型。。torchaudio 也基于 PyTorch 构建，可以在 PyTorch 项目中轻松集成。
- CUDA：CUDA 是 NVIDIA 开发的一种并行计算平台和 API 模型，用于在 NVIDIA GPU 上加速计算任务。它允许开发人员编写在 GPU 上运行的代码，从而加快计算速度。PyTorch 等深度学习库通常使用 CUDA 来进行 GPU 加速。

结合所选取的大模型，参考 https://pytorch.org/get-started/previous-versions/ 选取合适的环境 package 组合。

- Bitsandbytes：提供高效的位操作和字节操作功能。它可以帮助开发人员更轻松地处理二进制数据和进行位级操作。

很多情况大模型所需要的服务器配额我们是无法满足的，这个时候需要 bitsandbytes 降低推理精度以达到降低所需显存配额的目的。

conda 安装

如果确保机器只归你一个人使用可以不用安装

- 下载并安装 anaconda，选择你需要的版本即可上机安装：https://repo.anaconda.com/archive/

  以 Tlinux 为例：下载 Anaconda3-2023.07-2-Linux-x86_64.sh 并执行，等待安装完成。

```
bash Anaconda3-2023.07-2-Linux-x86_64.sh
```

- 是否安装成功

```
# 查看 conda 版本
conda --version
# 提示没有 command，设置环境变量即可
# 获取 conda 的安装路径
# whereis conda
export PAHT=/usr/anconda/bin:$PATH
source ~/.bashrc
```

- conda 常用命令

```
# 查看当前存在的虚拟环境
conda env list
# 创建虚拟环境
conda create -n envName
# 激活虚拟环境
conda activate envName
# 退出虚拟环境
conda deactivate
# 删除虚拟环境
conda remove -n envName --all
# 虚拟环境中安装 packge，以安装 pytorch 1.13.1 版本为例
conda install pytorch==1.13.1
# 更多的 conda 命令
conda --help
```

demo 环境搭建

以 cuda11.3 为例搭建大模型运行环境。
说明，cuda11.3 较为典型，在 bitsandbytes 中没有预先编译适配 11.3 的 so。所以 11.3 的整个环境搭建具备完整环境搭建流程，适配所有 cuda 版本环境搭建。

```
# 默认情况下腾讯云服务器上镜像已经安装了 nvidia 驱动。
# 查询显卡信息，该命令将会输出当前显卡 Driver Version，CUDA Version
nvidia-sm
# print CUDA Version: 11.4，说明当前环境适配 CUDA Version <= 11.4

# 下载并安装 CUDA 11.3，需要注意的是：安装过程中会让你选择安装像，这里需要去除安装驱动。
wget https://developer.download.nvidia.com/compute/cuda/11.3.0/local_installers/cuda_11.3.0_465.19.01_linux.run
sudo sh cuda_11.3.0_465.19.01_linux.run
# 设置环境变量，通常/usr/local/cuda-1x.x/bin，/usr/local/cuda-1x.x/lib64
export PATH=/usr/local/cuda-11.3/bin:$PATH
source ~/.bashrc
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11.3/lib64
# 或者将/usr/local/cuda-11.3/lib64 添加到/etc/ld.so.conf

##### conda 和 pip 安装二选一 start ####
# 参考 https://pytorch.org/get-started/previous-versions/
# 1、conda
conda create -n starchat
conda activate starchat
conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=11.3 -c pytorch

# 2、pip
pip install torch==1.12.1+cu113 torchvision==0.13.1+cu113 torchaudio==0.12.1 --extra-index-url https://download.pytorch.org/whl/cu113
#### conda 和 pip 安装二选一 end ####

# 安装 bitsandbytes
pip install bitsandbytes
# make 对应版本的 bitsandbytes
git clone https://github.com/timdettmers/bitsandbytes.git
cd bitsandbytes
CUDA_VERSION=11.3 make cuda11x
python setup.py install
CUDA_VERSION=11.3 make cuda11x_nomatmul
python setup.py install
# 编译好的 so 通常在 build/lib/bitsandbytes 路径下，libbitsandbytes_cuda113.so, libbitsandbytes_cuda113_nocublaslt.so
# 将这两个文件放在 python/site-packages/bitsandbytes 目录下
# 根据服务器环境复制到对应的目录下
#### start ####
# 1、conda
# 如 conda 环境中使用的 python3.10
cp build/lib/bitsandbytes/libbitsandbytes_cuda113.so /root/.conda/envs/starchat/lib/python3.10/site-packages/bitsandbytes/libbitsandbytes_cuda113.so
cp build/lib/bitsandbytes/libbitsandbytes_cuda113_nocublaslt.so /root/.conda/envs/starchat/lib/python3.10/site-packages/bitsandbytes/libbitsandbytes_cuda113_nocublaslt.so
# 2、非 conda，使用的是 python3.8
cp build/lib/bitsandbytes/libbitsandbytes_cuda113.so /usr/local/lib64/python3.8/site-packages/bitsandbytes/libbitsandbytes_cuda113.so
cp build/lib/bitsandbytes/libbitsandbytes_cuda113_nocublaslt.so
```

```
/usr/local/lib64/python3.8/site-packages/bitsandbytes/libbitsandbytes_cuda113_nocublaslt.so
#### end ####
# 测试 bitsandbytes
python -m bitsandbytes
# success print：True
# 否则，这里可能存在两个缺少依赖的报错：
#    1、No module named 'scipy'。解决：pip install scipy
#    2、No module named 'triton.language'。解决：pip install triton
#                安装完成 triton 后依然报错相同的错误，那么需要修改下
/usr/local/lib64/python3.8/site-packages/bitsandbytes/triton/triton_util.py 关于引用 importlib 的方式
#       修改代码 import importlib -> import importlib.util
#       没错，我为了安装这个环境已经把 bitsandbytes 源码看完了

# 整个环境已经搭建完成
# 关于如何开启 8int 方式运行大模型可以参考
#       https://github.com/timdettmers/bitsandbytes.git
```

至此 demo 中的大模型环境已经搭建完成。
大家可以在 huggingface 或者 百度飞浆 中获取自己想要的 AI 模型来愉快的玩耍

关于该 git 项目需要注意的一些点

```
cd 大模型目录
pip install -r
requirements.txt
# 进入 python 命令行 python


from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
tokenizer = AutoTokenizer.from_pretrained(checkpoint, trust_remote_code=True)
model = AutoModelForCausalLM.from_pretrained(checkpoint, trust_remote_code=True, torch_dtype=torch.float16)
model.to(device)
inputs = tokenizer.encode('现在我是 javascript 工程师，需要用 nextjs 实现文件上传，请你给出实现方案',
return_tensors="pt").to(device)
outputs = model.generate(inputs, generation_config=generation_config)
output = tokenizer.decode(outputs[0], skip_special_tokens=False).rsplit(assistant_token, 1)[1].rstrip(end_token)
print(output)
"""
其实你会得到一个让你非常无语的答案，哈哈哈。
#1•用户登录成功后，点击头像可以进入个人中心页面。
在个人中心页面有一个上传文件的按钮，，用户点击该按钮就可以选择要上传的文件并将其显示在页面上。
import os
def upload_ file (request):
...
...
"""
```

遇到这种情况不要慌，并不是大模型的能力有问题。这个引入一个初学大模型的概念 prompt 就是所谓的工程提示，如果我们给出合理的提示功能标签，那么大模型可以更好的识别问题。

常见的 LLM 通用提示标签有

- <|system|>： 系统级提示
- <|user|>： 用户输入
- <|assistant|>：ai 回答
- <|end|>：通用结束标签

那么以上问题，通过合理的 prompt 之后是：

```
# “<|system|>你是一个 javascript 工程师<|end|>
# <|user|>请用 nextjs 实现文件上传功能。<|end|>”
```

inputs 这里应该是：

```
inputs = tokenizer.encode(
    '<|system|>你是一个 javascript 工程师<|end|><|user|>请用 nextjs 实现文件上传功能。<|end|>',
return_tensors="pt").to(device)
outputs = model.generate(inputs, generation_config=generation_config)
output = tokenizer.decode(outputs[0],
  skip_special_tokens=False).rsplit(assistant_token, 1)[1].rstrip(end_token)
print(output)
```

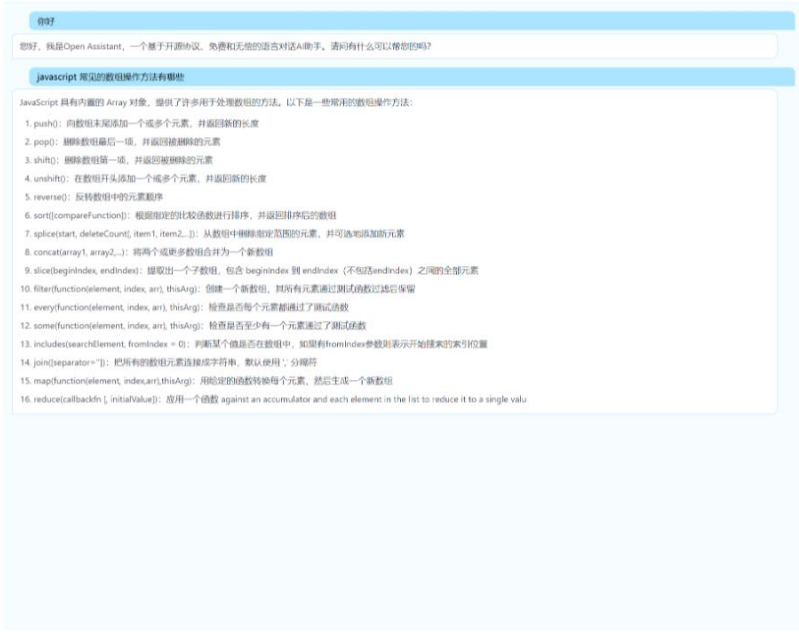
前后端封装

前后端封装相对比较简单，主要干的就是：fastapi 封装应用接口、大模型输出结果用接口返回给前端页面、前端页面收集到的问题和参数通过接口调用大模型。代码已传。

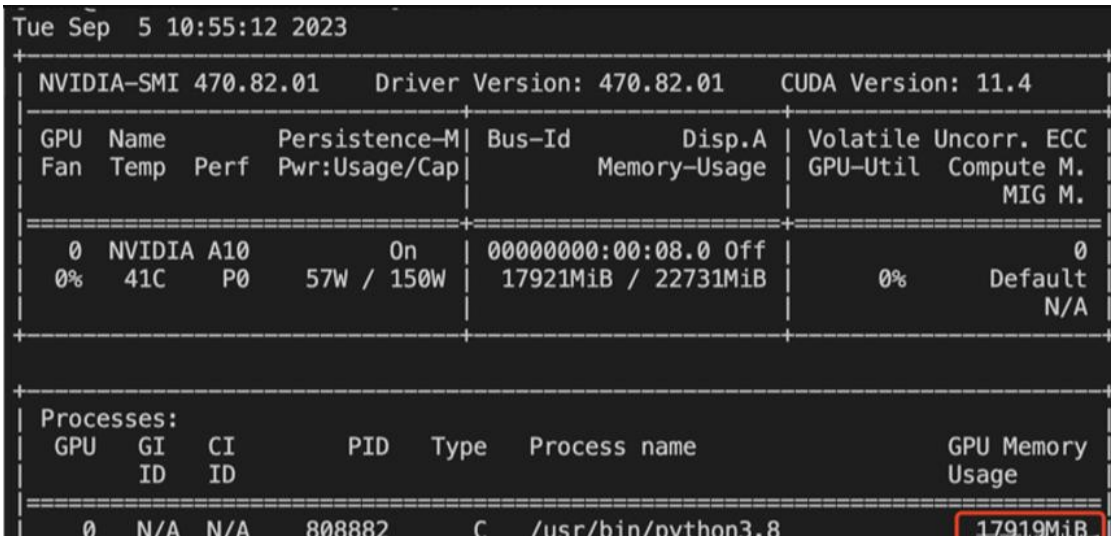
有点开发经验的同学应该能看懂，该文主要是分享记录大模型基础环境的搭建，就不赘述这部分内容啦。

实现效果截图 和 一点后续的废话：

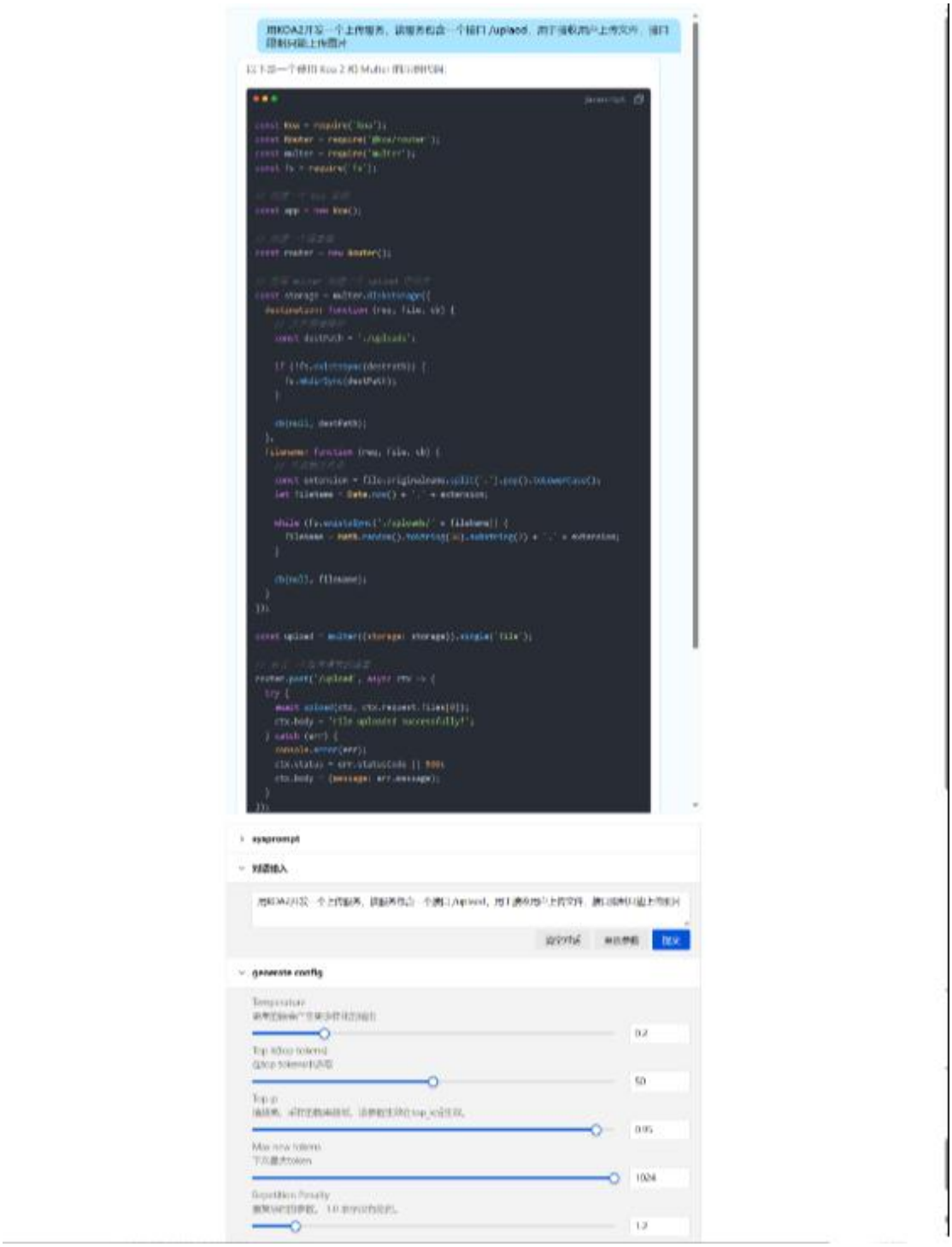
- 本人手里也没有合适跑 AI 模型的机器，是在腾讯云上薅了一个新用户的福利，几十块租了一星期练手。后面机器到期了所以没法发链接给读者亲身体验，不过一些步骤有截图记录，同时项目上传了 github 仓库，会提供 git 地址。跑了 starchat（语言模型） 和 clip（图像识别）两个模型。
- 套壳后的 startchat ，如图：（哈哈）



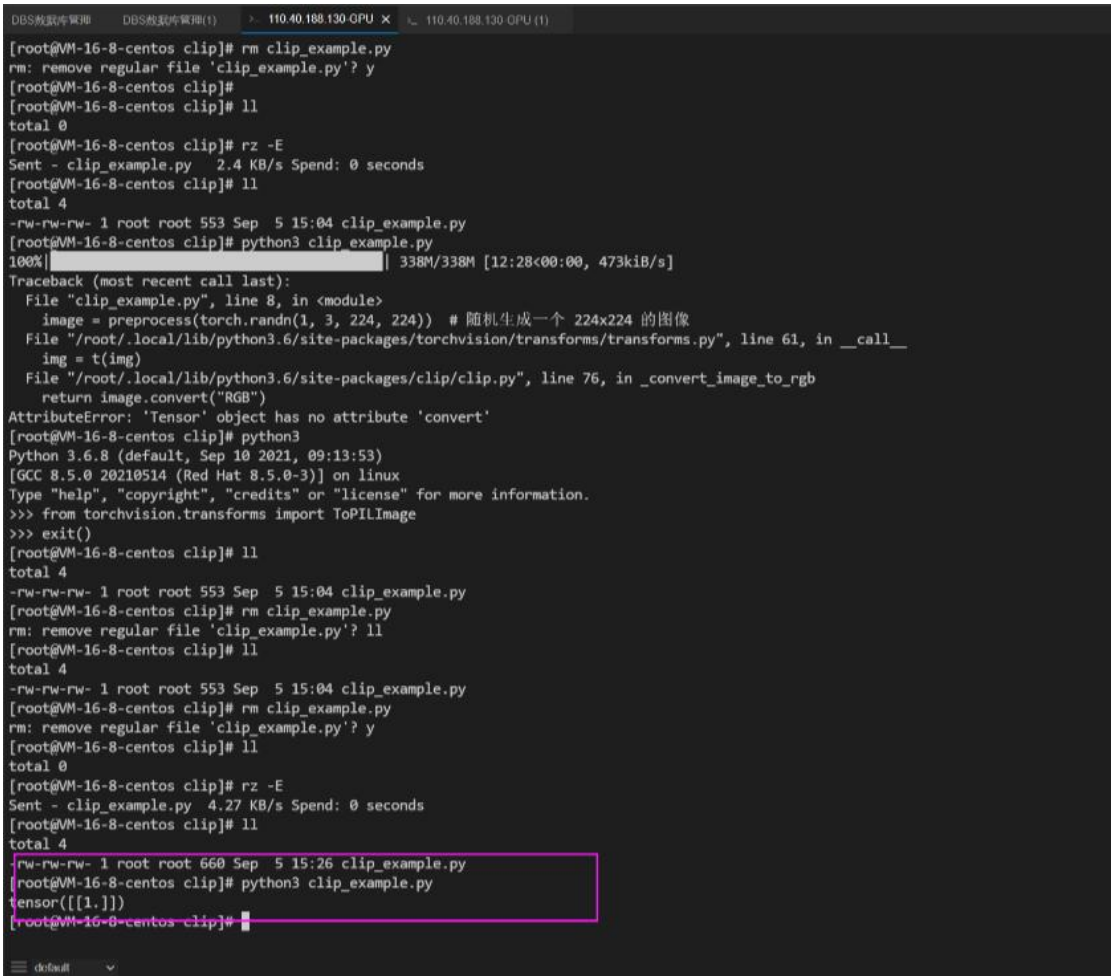
- 看一下运行时的显存（跑起来差不多用了 18 个 G）



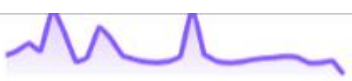
- 这个是完整页面，前端页面有一些可调参数



- clip 运行的情况 （clip 相对没 starchat 这么吃显存）


DBS故障库管理
DBS故障库管理(1)
110.40.188.130-GPU
110.40.188.130-GPU (1)
[root@VM-16-8-centos clip]# rm clip_example.py
rm: remove regular file 'clip_example.py'? y
[root@VM-16-8-centos clip]#
[root@VM-16-8-centos clip]# ll
total 0
[root@VM-16-8-centos clip]# rz -E
Sent - clip_example.py 2.4 KB/s Spend: 0 seconds
[root@VM-16-8-centos clip]# ll
total 4
-rw-rw-rw- 1 root root 553 Sep 5 15:04 clip_example.py
[root@VM-16-8-centos clip]# python3 clip_example.py
100%| | 338M/338M [12:28<00:00, 473kiB/s]
Traceback (most recent call last):
File "clip_example.py", line 8, in <module>
image = preprocess(torch.randn(1, 3, 224, 224)) # 随机生成一个 224x224 的图像
File "/root/.local/lib/python3.6/site-packages/torchvision/transforms/transforms.py", line 61, in __call__
img = t(img)
File "/root/.local/lib/python3.6/site-packages/clip/clip.py", line 76, in _convert_image_to_rgb
return image.convert("RGB")
AttributeError: 'Tensor' object has no attribute 'convert'
[root@VM-16-8-centos clip]# python3
Python 3.6.8 (default, Sep 10 2021, 09:13:53)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-3)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from torchvision.transforms import ToPILImage
>>> exit()
[root@VM-16-8-centos clip]# ll
total 4
-rw-rw-rw- 1 root root 553 Sep 5 15:04 clip_example.py
[root@VM-16-8-centos clip]# rm clip_example.py
rm: remove regular file 'clip_example.py'? ll
[root@VM-16-8-centos clip]# ll
total 4
-rw-rw-rw- 1 root root 553 Sep 5 15:04 clip_example.py
[root@VM-16-8-centos clip]# rm clip_example.py
rm: remove regular file 'clip_example.py'? y
[root@VM-16-8-centos clip]# ll
total 0
[root@VM-16-8-centos clip]# rz -E
Sent - clip_example.py 4.27 KB/s Spend: 0 seconds
[root@VM-16-8-centos clip]# ll
total 4
-rw-rw-rw- 1 root root 660 Sep 5 15:26 clip_example.py
[root@VM-16-8-centos clip]# python3 clip_example.py
tensor([[1.]])
[root@VM-16-8-centos clip]#
default

Downloads last month
13,388,343



Hosted inference API

Zero-Shot Image Classification

Examples

Possible class names (comma-separated)

playing music, playing sports,sunglasses

Compute

Computation time on Intel Xeon 3rd Gen Scalable cpu: cached

墨镜 sunglasses 0.823

奏乐 playing music 0.110

运动 playing sports 0.067

JSON Output

Maximize