How to run LLM models

This tutorial shows, how to run LLM on UNIL clusters

Available models

You are free to download and use any LLM model you like in your `/work` space, but the process can sometimes be a bit tricky. In addition, several users may want to use the same models.

To simplify this, we now provide a selection of LLM models for the whole community. These models are obtained directly from the Hugging Face Hub.

Location: `/reference/LLM/`

Naming convention

Folder names follow Hugging Face terminology. For example, the folder `meta-llama/Llama-3.1-8B-Instruct` corresponds to the model available at: https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

If the LLM you need is not available, you can request its installation by sending an email to `helpdesk@unil.ch` with the subject line: DCSR Request LLM installation.

Simple test

set up

For this simple test, we are going to use `transformers` library from hugging face. So you should type the following commands to setup a proper python environment:

```
module load python
python -m venv venv
source venv/bin/activate
pip install transformers accelerate torch
```

If you plan to use a instruct model, you will need a chat template file which you can download from https://github.com/chujiezheng/chat_templates . For this example, we are going to use the llama template

```
wget
https://raw.githubusercontent.com/chujiezheng/chat_templates/refs/heads/main/chat_templates/ll
ama-3-instruct.jinja
```

Then, you should create the following python file:

```
from transformers import AutoModelForCausalLM
from transformers import AutoTokenizer

model_hf='/reference/LLM/meta-llama/Llama-3.1-8B-Instruct/'

model = AutoModelForCausalLM.from_pretrained(model_hf, device_map="auto")
tokenizer = AutoTokenizer.from_pretrained(model_hf)

with open('llama-3-instruct.jinja', "r") as f:
    chat_template = f.read()

tokenizer.chat_template = chat_template

with open('prompt.txt') as f:
    prompt=f.read()

prompts = [
    [{'role': 'user', 'content': prompt}]
]

model_inputs = tokenizer.apply_chat_template(
    prompts,
    return_tensors="pt",
    tokenize=True,
    add_generation_prompt=True #This is for adding prompt, useful in chat mode
).to("cuda")

generated_ids = model.generate(
    model_inputs,
    max_new_tokens=400,
```

```
)

for i, answer in enumerate(tokenizer.batch_decode(generated_ids, skip_special_tokens=True)):
    print(answer)
```

This python code reads a prompt from a text file called prompt.tx and uses the model 8B de LLama to perform the inference.

To turn it in the cluster, we can use the following job script:

```
#!/bin/bash

#SBATCH -p gpu
#SBATCH --mem 20G
#SBATCH --gres gpu:1
#SBATCH -c 2

source venv/bin/activate
python run_inference.py
```

You should run the previous command sbatch:

```
sbatch job.sh
```

The result of the inference will be written in the SLURM file `slurm-xxxx.out`

Using VLLM

If you need to run big models, you can use VLLM library which uses less GPU memory. To install it:

```
pip install vllm
```

Then, you can use it with the following simple code:

```
import os
import time
from vllm import LLM, SamplingParams

num_gpus =
len(os.environ['CUDA_VISIBLE_DEVICES'].split(","))
```

```
model_hf='/reference/LLM/meta-llama/Llama-3.1-8B-Instruct/'

with open('llama-3-instruct.jinja', "r") as f:
    chat_template = f.read()

with open('prompt.txt') as f:
    prompt=f.read()

prompts = [
    [{'role': 'user', 'content': prompt}]
]

model = LLM(model=model_hf,tensor_parallel_size=num_gpus)

sampling = SamplingParams(
    n=1,
    temperature=0,
    max_tokens=400,
    skip_special_tokens=True,
    stop=["<|eot_id|>"]
)

output = model.chat(
    prompts, sampling, chat_template=chat_template,
)



results = []
for i, out in enumerate(output):
    answer = out.outputs[0].text
    print(answer)
```

If you need to use several GPUs, do not forget to put `#SBATCH --gres gpu:2` in your job description

---

Créé 2025-01-30 10:22:56 UTC par Cristian Ruiz
Mis à jour 2026-03-20 18:16:48 UTC par Philippe Jacquet