

How to run LLM models

This tutorial shows, how to run LLM on UNIL clusters

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/llama-3-instruct.jinja
```

Then, you should create the following python file:

```
from transformers import AutoModelForCausalLM
from transformers import AutoTokenizer

model_hf='/reference/LLMs/Llama3/llama3_1/Meta-Llama-3.1-8B-Instruct-hf/'

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/LLMs/Llama3/llama3_1/Meta-Llama-3.1-8B-Instruct-hf/'

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

Révision #9

Créé 30 janvier 2025 10:22:56 par Cristian Ruiz

Mis à jour 20 mars 2025 13:12:46 par Cristian Ruiz