SGLang Integration for Open R1

Integrating SGLang as an alternative inference backend for the Open R1 project (Hugging Face Series).

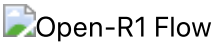
Table of Contents

Overview

The purpose of this integration is to introduce **SGLang** as a new inference backend within the Open R1 pipeline. Currently, Open R1 relies on **vLLM** for *GRPO training*, *data generation*, and *evaluation*. This documentation outlines the approach to seamlessly support SGLang while preserving the option to use vLLM when needed.

Integration Overview

The Open-R1 Repo is built around a modular pipeline that consists of three main steps:



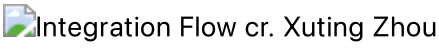
- **Training:** The `grpo.py` script employs the `GRPOTrainer` module from the TRL repository which currently uses vLLM for fast inference.

```
from datasets import load_dataset
from trl import GRPOConfig, GRPOTrainer

dataset = load_dataset("trl-lib/tldr", split="train")

# Dummy reward function: rewards completions that are close to 20
characters
def reward_len(completions, **kwargs):
    return [abs(20 - len(completion)) for completion in completions]

training_args = GRPOConfig(output_dir="Qwen2-0.5B-GRPO", logging_steps=10)
trainer = GRPOTrainer(
    model="Qwen/Qwen2-0.5B-Instruct",
    reward_funcs=reward_len,
    args=training_args,
```

```
    train_dataset=dataset,
)
trainer.train()
```

- **Data Generation:** The `generate.py` script utilizes components from the Distilabel project (e.g., OpenAILLM) to create synthetic data. These components are also dependent on vLLM. Xuting(@Xuting Zhou) and Jin(@Jin Pan) will explore this first.
- **Evaluation:** The evaluation pipeline (`evaluate.py` & `pipeline.py`) leverages the Lighteval repository where vLLM is used for backend acceleration, with custom tasks defined in *src/open_r1/evaluate.py*. Detailed Doc can be found in Use VLLM as backend . Currently Qiujiang(@Qiujiang Chen) is responsible for this part

The integration effort with SGLang involves creating adapters and configuration toggles that allow users to select SGLang as an alternative backend without disrupting the current workflow shown as the picture below:


Integration Flow cr. Xuting Zhou

Integration Tutorial (In Progress)

Installation

To use SGLang in the Open R1 project, install SGLang alongside the other dependencies. Below are example commands for setting up your environment:

1. **Set up your Python environment** (using your preferred virtual environment tool):

```
python -m venv openr1-env && source openr1-env/bin/activate
pip install --upgrade pip
```

2. **Install vLLM** (if you plan to keep it as an option):

```
pip install vllm>=0.7.0
```

3. **Install SGLang** as the inference backend:

```
pip install sgl-kernel --force-reinstall --no-deps
pip install "sglang[all]" --find-links
https://flashinfer.ai/whl/cu124/torch2.4/flashinfer/
```

For details on the correct versions and CUDA compatibility, refer to the FlashInfer installation documentation.

After installing the inference backends, install the remaining project dependencies:

```
pip install -e ".[dev]"
```