PortLLM: Personalizing Evolving Large Language Models with Training-Free and Portable Model Patches

Overview

As large language models (LLMs) increasingly shape the AI landscape, fine-tuning pretrained models has become more popular than in the pre-LLM era for achieving optimal performance in domain-specific tasks. However, pretrained LLMs such as ChatGPT are periodically evolved (i.e., model parameters are frequently updated) making it challenging for downstream users with limited resources to keep up with fine-tuning the newest LLMs for their domain application. Even though fine-tuning costs have nowadays been reduced thanks to the innovations of parameter-efficient fine-tuning such as LoRA, not all downstream users have adequate computing for frequent personalization. Moreover, access to fine-tuning datasets, particularly in sensitive domains such as healthcare, could be time-restrictive, making it crucial to retain the knowledge encoded in earlier fine-tuned rounds for future adaptation. In this paper, we present PORTLLM, a training-free framework that (i) creates an initial lightweight model update patch to capture domain-specific knowledge, and (ii) allows a subsequent seamless plugging for the continual personalization of evolved LLM at minimal cost. Our extensive experiments cover a series of seven representative datasets, from easier question-answering tasks {BoolQ, SST2} to harder reasoning tasks {WinoGrande, GSM8K}, and models including {Mistral-7B, Llama2, Llama3.1, and Gemma2}, validating the portability of our designed model patches and showcasing the effectiveness of our proposed framework. For instance, PORTLLM achieves comparable performance to LoRA fine-tuning with reductions of upto 12.21× in GPU Memory. Finally, we provide theoretical justifications to understand the portability of our model update patches, which offers new insights into the theoretical dimension of large language models’ personalization.

Setup

```
conda create -n portllm python=3.10 -y && conda activate portllm
pip3 install -r requirements.txt
git clone --depth 1 https://github.com/EleutherAI/lm-evaluation-harness
cd lm-evaluation-harness
pip install -e .
cd ..
```

Usage

Note : To fine-tune and use any of the models used in the paper, please make an account on huggingface, and sign in with your token using the huggingface CLI. If you would like to work with local models, this is not necessary.

Fine-tuning on Downstream tasks

Replace {dataset} with downstream task you would like to fine-tune on.

```
accelerate launch scripts/run_{dataset}.py --max_length=256 --batch_size=4
--num_workers=8 --model_name_or_path="mistralai/Mistral-7B-v0.1" --rank=8 -
-lora_alpha=16 --lora_dropout=0.1 --lr=1e-4 --num_epochs=10 --seed=1234 --
output_dir={OUTPUT_DIR}
```

For example if you would like to reproduce the results on BoolQ for Mistral-7B, you can fine-tune it with the following command

```
accelerate launch scripts/run_boolq.py --max_length=256 --batch_size=4 --
num_workers=8 --model_name_or_path="mistralai/Mistral-7B-v0.1" --rank=8 --
lora_alpha=16 --lora_dropout=0.1 --lr=1e-4 --num_epochs=10 --seed=1234 --
output_dir={OUTPUT_DIR}
```

where OUTPUT_DIR is where you would like to store the model adapter. To see all the possible datasets you can fine-tune on please check the scripts folder.

Continued Pretraining

For Continued Pretraining we utilize the Axolotl framework. The config files are under configs folder for Axolotl.

Evaluation

To evaluate a model without its adapter on any dataset simply run the following command

```
accelerate launch -m lm_eval --model hf --model_args "pretrained={MODEL}" -
-tasks "{TASKS}" --num_fewshot 0 --batch_size 32
```

by replacing the {MODEL} with the model of your choice and {TASKS} by the task you would like to evaluate on. For example if you would like to evaluate Mistral-7B on GSM8K simply do the following

```
accelerate launch -m lm_eval --model hf --model_args
"pretrained=mistralai/Mistral-7B-v0.1" --tasks "gsm8k" --num_fewshot 0 --
batch_size 32
```

If you would like to also evaluate the adapter performance run the following command

```
accelerate launch -m lm_eval --model hf --model_args
"pretrained=mistralai/Mistral-7B-v0.1,peft={PEFT}" --tasks "gsm8k" --
num_fewshot 0 --batch_size 32
```

where you can replace {PEFT} with the adapter weights.

Merge Adapters

To merge adapters with the model itself please run the following command

```
python3 merge_adapters.py --base_model_name_or_path {BASE_MODEL} \
    --peft_model_path {PEFT_MODEL} \
    --output_dir {OUTPUT_DIR} \
    --push_to_hub
```

where you can replace the {BASE_MODEL}, {PEFT_MODEL}, and {OUTPUT_DIR} with your choice.