NeMo P-tuning

NVIDIA Professional Services & Quantiphi

1. Introduction

This document consists of instructions to execute p-tuning for two downstream tasks: Extractive Question Answering and Sentiment Analysis. The pre-trained GPT model used for this demonstration is GPT-3 5 billion parameter model.

This demonstration is intended to showcase the capability of Megatron Nemo to p tune Large language models like GPT on downstream tasks with support for multi-GPU training. Another special feature that p - tuning demonstrates is the capability of the same prompt model to be p tuned on multiple tasks.

In this experiment we have shown that the virtual prompts that were first trained on SQUAD dataset can be tuned on sentiment analysis tasks and in the end inference for both kinds of tasks is possible.

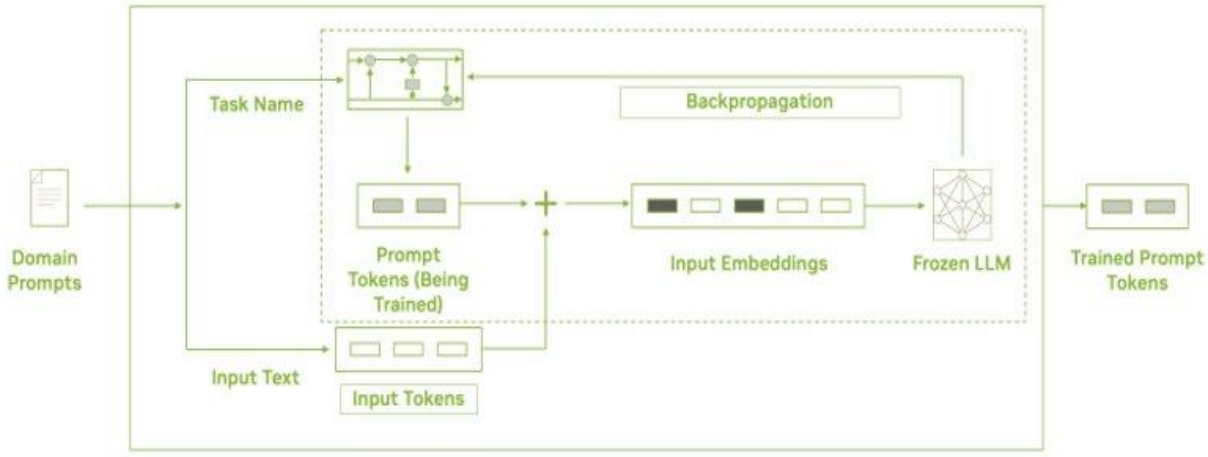

Fig 1.1 Prompt/P tuning

2. Prerequisites

Please follow the below instructions to pull the Megatron Nemo image and download the GPT 5B model.

- Pull docker image nvcr.io/nvidia/nemo:22.12,
  `sudo docker pull nvcr.io/nvidia/nemo:22.12`
- Download the model GPT5 B model using the below link https://huggingface.co/nvidia/nemo-megatron-gpt-5B/resolve/main/nemo_gpt5B_bf16_tp2.nemo

3. Working Directory

3.1 Content of the shared zipped folder

Below are the contents of the p_tuning demo folder

- p_tuning
  - megatron_gpt_prompt_learning.py
  - megatron_gpt_prompt_learning_sentiment.py
  - megatron_gpt_prompt_learning_eval.py
  - preprocess_sentiment_data.py
  - p_tuning_demo_squad.sh
  - p_tuning_demo_sentiment.sh
  - p_tuning_demo_inference.sh
  - GPT5B_model/
    - nemo_gpt5B_bf16_tp2.nemo
  - conf/
    - megatron_gpt_prompt_learning_config.yaml
    - megatron_gpt_prompt_learning_config_sentiment.yaml

        - megatron_gpt_prompt_learning_inference.yaml
- dataset/
        - train-v1.1.json
        - dev-v1.1.json
        - mini_squad_test_ground_truth.jsonl
        - mini_squad_test.jsonl
- sentiment_dataset/
        - train.csv
        - test.csv
        - train_sentiment.jsonl
        - test_sentiment.jsonl
        - test_sentiment_ground_truth.jsonl
        - mini_sentiment_test.jsonl
        - mini_sentiment_test_ground_truth.jsonl
- models/
- predictions/

3.2 Details of Subdirectories

- **GPT5B_model**: This folder should contain the **nemo_gpt5B_bf16_tp2.nemo** f file.
- **dataset** : Contains the squad dataset (train-v1.1.json, dev-v1.1.json, mini_squad_test_ground_truth.jsonl, mini_squad_test.jsonl)
- **sentiment_dataset** : Contains the sentiment analysis dataset (train_sentiment.jsonl, test_sentiment.jsonl, test_sentiment_ground_truth.jsonl,mini_sentiment_test.jsonl, mini_sentiment_test_ground_truth.jsonl)
- **conf** : Contains `.yaml` files that define configuration for p tuning (e.g. path to the GPT 5B model, path to the training dataset, hyperparameters etc.)
- **models** : Directory where the p -tuned model will be saved. It is empty when we mount it.

- **predictions**: Directory where the output file (.txt) will be written after the inference is completed.

4. Training and Inference

4.1 Details of files

- megatron_gpt_prompt_learning.py : This file is responsible for triggering the p tuning on SQUAD dataset. All the configurable parameters are read from the configuration file present under **conf** folder (conf/megatron_gpt_prompt_learning_config.yaml).
  Please refer to this link to access the file in the Nemo repository.

- megatron_gpt_prompt_learning_sentiment.py : This file is responsible for triggering p-tuning on sentiment analysis dataset. All the configurable parameters are read from the configuration file present under **conf** (conf/megatron_gpt_prompt_learning_config_sentiment.yaml) folder.
  Please refer to this link to access the file in the Nemo repository.

- megatron_gpt_prompt_learning_eval.py : This file is responsible for running inference on the dataset files mentioned in the inference configuration file (conf/megatron_gpt_prompt_learning_inference.yaml) and storing the output under predictions folder.
  Please refer to this link to access the file in the Nemo repository.

- p_tuning_demo_squad.sh : Bash script to trigger p tuning on SQUAD dataset.
- p_tuning_demo_sentiment.sh : Bash script to trigger p tuning on Sentiment analysis dataset.
- p_tuning_demo_inference.sh : Bash script to run inference on a subset of SQUAD test and Sentiment test datasets.

4.2 Contents of bash script

- The bash files `**p_tuning_demo_squad.sh**`, `**p_tuning_demo_sentiment.sh**` and `**p_tuning_demo_inference.sh**` contain docker run command to create a container from the image (nvcr.io/nvidia/nemo:22.12), mount the necessary folders and run training and inference scripts.

```
docker run --gpus all \
            -it \
            --rm \
            --shm-size=5gb \
            -v ${pwd}/dataset:/workspace/dataset \
            -v ${pwd}/conf:/workspace/conf \
            -v ${pwd}/megatron_gpt_prompt_learning.py:/workspace/megatron_gpt_prompt_learning.py \
            -v ${pwd}/megatron_gpt_prompt_learning_eval.py:/workspace/megatron_gpt_prompt_learning_eval.py \
            -v ${pwd}/GPT5B_model:/workspace/GPT5B_model \
            -v ${pwd}/models:/workspace/models \
            -v ${pwd}/predictions:/workspace/predictions \
            nvcr.io/nvidia/nemo:22.12 \
            bash -c 'cd .. && \
            python3 nemo/scripts/dataset_processing/nlp/squad/prompt_learning_squad_preprocessing.py --data-dir dataset && \
            python3 megatron_gpt_prompt_learning.py'
```

Fig 1.2 snapshot of p_tuning_demo_squad.sh

It consists of a docker run command, with the below flags.

- –gpus : allocate all the GPUs to the container that will be spun.
- -it: Interactive mode
- –rm: remove the container once training is completed and model is saved in the models directory.
- –shm-size : shared memory size
- -v : flag to mount a volume from the base machine onto the docker container.
- nvcr.io/nvidia/nemo:22.12 : The image that was pulled in the prerequisites section.
- bash -c : the commands following `bash -c` will be executed once the container is created.
    - `cd ..` : go one directory back.
    - `python3 ….squad_preprocessing.py …. dataset` : run preprocessing on the dataset.
    - `python3 megatron_gpt_prompt_learning.py` : run p tuning

4.3 Instructions to execute training and inference

- `**p_tuning/GPT5B_model**` directory will be empty in the shared zipped folder.

- The downloaded GPT 5B models should be placed under the `**p_tuning/GPT5B_model**` folder.

- Copy the entire p_tuning folder to the GPU VM.

- To execute the bash scripts to trigger training and inference follow the following sequence of commands. Make sure to let the execution of one bash script complete before triggering the next one.

Go to the p_tuning folder : `cd p_tuning`
P tuning on SQUAD dataset : `sudo bash p_tuning_demo_squad.sh`
P tuning on sentiment dataset : `sudo bash p_tuning_demo_sentiment.sh`
Inference on **subset** of test dataset of both squad and sentiment : `sudo bash p_tuning_demo_inference.sh`

- If the **docker run** command from the bash scripts is to be executed from a yaml config file in a **kubernetes** environment setup, the mount paths of folders should be changed if required.

5. Reference Configuration

The shared zipped folder contains configuration files with all the required changes, hence no changes are required. Below snapshots have been shared for reference.

5.1 Configuration file for p-tuning on SQUAD dataset

Config file parameters for training (megatron_gpt_prompt_learning_config.yaml)

```
model:
  seed: 1234
  nemo_path: /workspace/models/${name}.nemo # .nemo filename/absolute path to where the virtual prompt model parameters will be saved
  virtual_prompt_style: 'p-tuning' # one of 'prompt-tuning', 'p-tuning', or 'inference'
  tensor_model_parallel_size: 2 # intra-layer model parallelism
  pipeline_model_parallel_size: 1 # inter-layer model parallelism
  global_batch_size: 8
  micro_batch_size: 4

  restore_path: null # Path to an existing p-tuned/prompt tuned .nemo model you wish to add new tasks to or run inference with
  language_model_path: /workspace/GPT5B_model/nemo_gpt5B_bf16_tp2.nemo # Path to the GPT language model .nemo file, always required
  save_nemo_on_validation_end: True # Saves an inference ready .nemo file every time a checkpoint is saved during training.
  existing_tasks: [] # List of tasks the model has already been p-tuned/prompt-tuned for, needed when a restore path is given
  new_tasks: ['squad'] # List of new tasknames to be prompt-tuned
```

In the above snapshot **model** section in the configuration file is shown. It has parameters relevant to the kind of model we want to train.
In the above snapshot, the useful parameters are

- virtual_prompt_style : It could be either ‘p_tuning’ or ‘prompt_tuning’.
- Tensor_model_parallel_size : It should be set to 2 since it is a model variant with Tensor Parallelism (TP) of 2 and Pipeline Parallelism (PP) of 1 on two GPUs.
- nemo_path : The path when the p tuned model will be saved.
- language_model_path : path to the pretrained GPT 5 billion parameters model.
- restore_path : should be null because we are p tuning for the first time.
- exisiting_tasks : Should be an empty list because the virtual prompts were not tuned to any downstream tasks.
- new_tasks : List of downstream tasks.

```
trainer:
  devices: 2
  accelerator: gpu
  num_nodes: 1
  precision: 16
  logger: False # logger provided by exp_manager
  enable_checkpointing: False
  replace_sampler_ddp: False
  max_epochs: 1 # min 25 recommended
  max_steps: -1 # consumed_samples = global_step * micro_batch_size * data_parallel_size * accumulate_grad_batches
  log_every_n_steps: 10 # frequency with which training steps are logged
  val_check_interval: 1.0 # If is an int n > 1, will run val every n training steps, if a float 0.0 - 1.0 will run val every epoch fraction, e.g. 0.25 will run val every quarter
  gradient_clip_val: 1.0
  resume_from_checkpoint: null # The path to a checkpoint file to continue the training, restores the whole state including the epoch, step, LR schedulers, apex, etc.
  benchmark: False
```

In the above snapshot **trainer** section in the configuration file is shown. It has parameters relevant to the kind of model we want to train.

In the above snapshot, the useful parameters are

- devices : Number of GPUs we want to utilize in training.
- num_nodes : number of GPU compute instances. It should be set to 1 since we only have 1 A100 instance with 2 GPUs.
- max_epochs : It is set to 1 for this demonstration purpose.

```
data:
  train_ds: [/workspace/dataset/squad_train.jsonl,]
  validation_ds: [/workspace/dataset/squad_val.jsonl,]
  add_eos: True
  shuffle: True
  num_workers: 8
  pin_memory: True
  train_cache_data_path: null  # the path to the train cache data
  validation_cache_data_path: null  # the path to the validation cache data
  test_cache_data_path: null  # the path to the test cache data
  load_cache: False  # whether to load from the cache data
```

In the above snapshot **trainer** section in the configuration file is shown. It has parameters relevant to the kind of model we want to train.

In the above snapshot, the useful parameters are

- train_ds : path to training (.jsonl) file.
- validation_ds : path to validation (.jsonl) file.

5.2 Configuration file for p-tuning on sentiment analysis dataset

Config file parameters for training (megatron_gpt_prompt_learning_config_inference.yaml)

```
model:
  seed: 1234
  nemo_path: /workspace/models/${name}.nemo # .nemo filename/absolute path to where the virtual prompt model parameters will be saved
  virtual_prompt_style: 'p-tuning' # one of 'prompt-tuning', 'p-tuning', or 'inference'
  tensor_model_parallel_size: 2 # intra-layer model parallelism
  pipeline_model_parallel_size: 1 # inter-layer model parallelism
  global_batch_size: 8
  micro_batch_size: 4

  restore_path: /workspace/models/${name}.nemo # Path to an existing p-tuned/prompt tuned .nemo model you wish to add new tasks to or run inference with
  language_model_path: /workspace/GPT5B_model/nemo_gpt5B_bf16_tp2.nemo # Path to the GPT language model .nemo file, always required
  save_nemo_on_validation_end: True # Saves an inference ready .nemo file every time a checkpoint is saved during training.
  existing_tasks: ['squad'] # List of tasks the model has already been p-tuned/prompt-tuned for, needed when a restore path is given
  new_tasks: ['sentiment'] # List of new tasknames to be prompt-tuned
```

In the above snapshot **model** section in the configuration file is shown. It has parameters relevant to the kind of model we want to train.

In the above snapshot, the useful parameters are

- virtual_prompt_style : It could be either ‘p_tuning’ or ‘prompt_tuning’.
- Tensor_model_parallel_size : It should be set to 2 since it is a model variant with Tensor Parallelism (TP) of 2 and Pipeline Parallelism (PP) of 1 on two GPUs.
- nemo_path : The path when the p-tuned model will be saved.
- language_model_path : path to the pre-trained GPT 5 billion parameters model.
- restore_path : should be set to the path of virtual prompts stored in the models directory after the SQUAD p tuning was completed.
- exisiting_tasks : should contain ‘squad’ keyword in the list.
- new_tasks : should contain ‘sentiment’ keyword in the list.

```
trainer:
  devices: 2
  accelerator: gpu
  num_nodes: 1
  precision: 16
  logger: False # logger provided by exp_manager
  enable_checkpointing: False
  replace_sampler_ddp: False
  max_epochs: 1 # min 25 recommended
  max_steps: -1 # consumed_samples = global_step * micro_batch_size * data_parallel_size * accumulate_grad_batches
  log_every_n_steps: 10 # frequency with which training steps are logged
  val_check_interval: 1.0 # If is an int n > 1, will run val every n training steps, if a float 0.0 - 1.0 will run val every epoch fraction, e.g. 0.25 will run val every quarter
  gradient_clip_val: 1.0
  resume_from_checkpoint: null # The path to a checkpoint file to continue the training, restores the whole state including the epoch, step, LR schedulers, apex, etc.
  benchmark: False
```

In the above snapshot **trainer** section in the configuration file is shown.

In the above snapshot, the useful parameters are

- devices : Number of GPUs we want to utilize in training.
- num_nodes : number of GPU compute instances. It should be set to 1 since we only have 1 A100 instance with 2 GPUs.
- max_epochs : It is set to 1 for this demonstration purpose.

```
data:
  train_ds: [/workspace/sentiment_dataset/train_sentiment.jsonl,]
  validation_ds: null
  add_eos: True
  shuffle: True
  num_workers: 8
  pin_memory: True
  train_cache_data_path: null  # the path to the train cache data
  validation_cache_data_path: null  # the path to the validation cache data
  test_cache_data_path: null  # the path to the test cache data
  load_cache: False  # whether to load from the cache data
```

In the above snapshot **data** section in the configuration file is shown. It has parameters relevant to the data we will be providing for training.
In the above snapshot, the useful parameters are

- train_ds : path to training (.jsonl) file.
- validation_ds : path to validation (.jsonl) file if available.

6. Resources

1. https://github.com/NVIDIA/NeMo/blob/main/examples/nlp/language_modeling/megatron_gpt_prompt_learning.py
2. https://github.com/NVIDIA/NeMo/blob/main/examples/nlp/language_modeling/megatron_gpt_prompt_learning_eval.py
3. https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset?resource=download[Sentiment analysis dataset]
4. https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v1.1.json [SQUAD training dataset]
5. https://rajpurkar.github.io/SQuAD-explorer/dataset/dev-v1.1.json [SQUAD validation and test dataset]
6. https://huggingface.co/nvidia/nemo-megatron-gpt-5B/resolve/main/nemo_gpt5B_bf16_tp2.nemo [GPT 5 Billion parameter model]