Model Garden NLP Common Training Driver

train.py is the common training driver that supports multiple NLP tasks (e.g., pre-training, GLUE and SQuAD fine-tuning etc) and multiple models (e.g., BERT, ALBERT, MobileBERT etc).

Experiment Configuration

train.py is driven by configs defined by the ExperimentConfig including configurations for `task` , `trainer` and `runtime` . The pre-defined NLP related ExperimentConfig can be found in configs/experiment_configs.py.

Experiment Registry

We use an experiment registry to build a mapping between experiment type to experiment configuration instance. For example, configs/finetuning_experiments.py registers `bert/sentence_prediction` and `bert/squad` experiments. User can use `--experiment` FLAG to invoke a registered experiment configuration, e.g., `--experiment=bert/sentence_prediction` .

Overriding Configuration via Yaml and FLAGS

The registered experiment configuration can be overridden by one or multiple Yaml files provided by `--config_file` FLAG. For example:

```
--config_file=configs/experiments/glue_mnli_matched.yaml \
--config_file=configs/models/bert_en_uncased_base.yaml
```

In addition, experiment configuration can be further overriden by `params_override` FLAG. For example:

```
 --
params_override=task.train_data.input_path=/some/path,task.hub_module_url=/some/tfhub
```

Run locally on GPUs

An example command for training a model on local GPUs is below. This command trains a BERT-base model on GLUE/MNLI-matched which is a sentence prediction task.

```
PARAMS=runtime.distribution_strategy=mirrored  # Train no GPU
PARAMS=${PARAMS},task.train_data.input_path=/path-to-your-training-data/

python3 train.py \
  --experiment=bert/sentence_prediction \
  --mode=train \
  --model_dir=/a-folder-to-hold-checkpoints-and-logs/ \
  --config_file=configs/models/bert_en_uncased_base.yaml \
  --config_file=configs/experiments/glue_mnli_matched.yaml \
  --params_override=${PARAMS}
```

Note that you can specify any detailed configuration by appending to the `PARAMS` variable. For example, if you want to load from a pretrained checkpoint as initialization (instead of random initialization):

```
PARAMS=${PARAMS},task.hub_module_url=https://tfhub.dev/tensorflow/bert_en_uncased_L-
12_H-768_A-12/4
```

The configuration entry `task.hub_module_url` uses a URL to a TF-Hub model which is officially pretrained. See List of Pretrained Models for the complete list of pretrained models on TF-Hub. When initializing from a pretrained model, the encoder architecture of the pretrained model will be used and the encoder architecture you set in the config (`configs/models/bert_en_uncased_base.yaml` in this case) will be ignored.

You can change `--mode=train` to `--mode=train_and_eval` if you want to see evaluation results. But you need to specify the path to the evaluation data by setting `task.validation_data.input_path` in `PARAMS` .

Run on Cloud TPUs

Next, we will describe how to run the train.py on Cloud TPUs.

Setup

First, you need to create a `tf-nightly` TPU with ctpu tool:

```
export TPU_NAME=YOUR_TPU_NAME
ctpu up -name $TPU_NAME --tf-version=nightly --tpu-size=YOUR_TPU_SIZE --
project=YOUR_PROJECT
```

and then install Model Garden and required dependencies:

```
git clone https://github.com/tensorflow/models.git
export PYTHONPATH=$PYTHONPATH:/path/to/models
pip3 install --user -r official/requirements.txt
```

Fine-tuning Sentence Classification with BERT from TF-Hub

▶ Details

Fine-tuning SQuAD with a pre-trained BERT checkpoint

▶ Details

Note: More examples about pre-training will come soon.