XLNet: Generalized Autoregressive Pretraining for Language Understanding

The academic paper which describes XLNet in detail and provides full results on a number of tasks can be found here: https://arxiv.org/abs/1906.08237.

XLNet is a generalized autoregressive BERT-like pretraining language model that enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order. It can learn dependency beyond a fixed length without disrupting temporal coherence by using segment-level recurrence mechanism and relative positional encoding scheme introduced in Transformer-XL. XLNet outperforms BERT on 20 NLP benchmark tasks and achieves state-of-the-art results on 18 tasks including question answering, natural language inference, sentiment analysis, and document ranking.

Contents

Set up

To run XLNet on a Cloud TPU, you can first create a `tf-nightly` TPU with the ctpu tool:

```
ctpu up -name <instance name> --tf-version="nightly"
```

After SSH'ing into the VM (or if you're using an on-prem machine), setup continues as follows:

```
export PYTHONPATH="$PYTHONPATH:/path/to/models"
```

Install `tf-nightly` to get latest updates:

```
pip install tf-nightly-gpu
```

Process Datasets

Dataset processing requires a Sentence Piece model. One can be found at the publicly available GCS bucket at: `gs://cloud-tpu-checkpoints/xlnet/cased_spiece.model`.

Note that in order to train using Cloud TPUs, data must be stored on a GCS bucket.

Setup commands:

```
export SPIECE_DIR=~/cased_spiece/
export SPIECE_MODEL=${SPIECE_DIR}/cased_spiece.model
```

```
export DATASETS_DIR=gs://some_bucket/datasets
mkdir -p ${SPIECE_DIR}
gsutil cp gs://cloud-tpu-checkpoints/xlnet/cased_spiece.model ${SPIECE_DIR}
```

Pre-training

Pre-training data can be converted into TFRecords using `preprocess_pretrain_data.py`. Inputs should consist of a plain text file (or a file glob of plain text files) with one sentence per line.

To run the script, use the following command:

```
export INPUT_GLOB='path/to/wiki_cased/*.txt'

python3 preprocess_pretrain_data.py --bsz_per_host=32 --num_core_per_host=16
--seq_len=512 --reuse_len=256 --input_glob='path/to/wiki_cased/*.txt'
--save_dir=${DATASETS_DIR}/pretrain --bi_data=True --sp_path=${SPIECE_MODEL}
--mask_alpha=6 --mask_beta=1 --num_predict=85
```

Note that to make the memory mechanism work correctly, `bsz_per_host` and `num_core_per_host` are *strictly specified* when preparing TFRecords. The same TPU settings should be used when training.

Fine-tuning

- Classification

To prepare classification data TFRecords on the IMDB dataset, users can download and unpack the IMDB dataset with the following command:

```
export IMDB_DIR=~/imdb
mkdir -p ${IMDB_DIR}

cd ${IMDB_DIR}
wget http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
tar zxvf aclImdb_v1.tar.gz -C ${IMDB_DIR}
rm aclImdb_v1.tar.gz
```

Then, the dataset can be converted into TFRecords with the following command:

```
export TASK_NAME=imdb

python3 preprocess_classification_data.py --max_seq_length=512 --spiece_model_file=${SPIECE_
```

Note: To obtain SOTA on the IMDB dataset, using a sequence length of 512 is necessary.

- SQUAD

The SQuAD website contains detailed information about the SQuAD datasets and evaluation.

To download the relevant files, use the following command:

```
export SQUAD_DIR=~/squad

mkdir -p ${SQUAD_DIR} && cd ${SQUAD_DIR}
wget https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json
wget https://rajpurkar.github.io/SQuAD-explorer/dataset/dev-v2.0.json
```

Then to process the dataset into TFRecords, run the following commands:

```
python3 preprocess_squad_data.py --spiece_model_file=${SPIECE_MODEL} --train_file=${SQUAD_DI

gsutil cp ${SQUAD_DIR}/dev-v2.0.json ${DATASETS_DIR}/squad
```

Fine-tuning with XLNet

- Cloud Storage

The unzipped pre-trained model files can be found in the Google Cloud Storage folder `gs://cloud-tpu-checkpoints/xlnet/keras_xlnet`. For example:

```
export XLNET_DIR=gs:/cloud-tpu-checkpoints/xlnet/keras_xlnet
export MODEL_DIR=gs://some_bucket/my_output_dir
```

Classification task

This example code fine-tunes `XLNet` on the IMDB dataset. For this task, it takes around 11 minutes to get the first 500 steps' results, and takes around 1 hour to complete on a v3-8. It is expected to obtain an accuracy between 96.15 and 96.33.

To run on a v3-8 TPU:

```
export TPU_NAME=my-tpu

python3 run_classifier.py \
--strategy_type=tpu \
--tpu=${TPU_NAME} \
--init_checkpoint=${XLNET_DIR}/xlnet_model.ckpt \
--model_dir=${MODEL_DIR} \
--test_data_size=25024 \
--train_tfrecord_path=${DATASETS_DIR}/imdb/cased_spiece.model.len-512.train.tf_record \
--test_tfrecord_path=${DATASETS_DIR}/imdb/cased_spiece.model.len-512.dev.eval.tf_record \
--train_batch_size=32 \
--seq_len=512 \
--n_layer=24 \
--d_model=1024 \
--d_embed=1024 \
--n_head=16 \
```

```
--d_head=64 \
--d_inner=4096 \
--untie_r=true \
--n_class=2 \
--ff_activation=gelu \
--learning_rate=2e-5 \
--train_steps=4000 \
--warmup_steps=500 \
--iterations=500 \
--bi_data=false \
--summary_type=last
```

SQuAD 2.0 Task

The Stanford Question Answering Dataset (SQuAD) is a popular question answering benchmark dataset. See more in SQuAD website.

We use `XLNet-LARGE` (cased_L-24_H-1024_A-16) running on a v3-8 as an example to run this workflow. It is expected to reach a `best_f1` score of between 88.30 and 88.80. It should take around 5 minutes to read the pickle file, and then 18 minutes to get the first 1000 steps' results. It takes around 2 hours to complete.

```
export TPU_NAME=my-tpu

python3 run_squad.py \
  --strategy_type=tpu \
  --tpu=${TPU_NAME} \
  --init_checkpoint=${XLNET_DIR}/xlnet_model.ckpt \
  --model_dir=${MODEL_DIR} \
  --train_tfrecord_path=${DATASETS_DIR}/squad/squad_cased \
  --test_tfrecord_path=${DATASETS_DIR}/squad/squad_cased/12048.eval.tf_record \
  --test_feature_path=${DATASETS_DIR}/squad/spiece.model.slen-512.qlen-64.eval.features.pkl
  --predict_dir=${MODEL_DIR} \
  --predict_file=${DATASETS_DIR}/squad/dev-v2.0.json \
  --train_batch_size=48 \
  --seq_len=512 \
  --reuse_len=256 \
  --mem_len=0 \
  --n_layer=24 \
  --d_model=1024 \
  --d_embed=1024 \
  --n_head=16 \
  --d_head=64 \
  --d_inner=4096 \
  --untie_r=true \
  --ff_activation=gelu \
```

```
--learning_rate=.00003 \
--train_steps=8000 \
--warmup_steps=1000 \
--iterations=1000 \
--bi_data=false \
--query_len=64 \
--adam_epsilon=.000001 \
--lr_layer_decay_rate=0.75
```