BigBird: Transformers for Longer Sequences

BigBird is a sparse attention mechanism that reduces this quadratic dependency to linear. BigBird is a universal approximator of sequence functions and is Turing complete, thereby preserving these properties of the quadratic, full attention model. Along the way, our theoretical analysis reveals some of the benefits of having O(1) global tokens (such as CLS), that attend to the entire sequence as part of the sparse attention mechanism.

Requirements

The starter code requires Tensorflow. If you haven't installed it yet, follow the instructions on tensorflow.org. This code has been tested with Tensorflow 2.5.0. Going forward, we will continue to target the latest released version of Tensorflow.

Please verify that you have Python 3.6+ and Tensorflow 2.5.0 or higher installed by running the following commands:

```
python --version
python -c 'import tensorflow as tf; print(tf.__version__)'
```

Refer to the instructions here for using the model in this repo. Make sure to add the models folder to your Python path.

Network Implementations

We implement the encoder and layers using `tf.keras` APIs in NLP modeling library:

- bigbird_attention.py contains the BigBird sparse attention implementation.
- encoders.py contains the integration of BigBird attention to the `EncoderScaffold`. Note that, currently the gradient checkpointing is implemented in bigbird/encoder.py.

Train using the config file.

Create a YAML file for specifying the parameters to be overridden. Working examples can be found in `bigbird/experiments` directory.

The code can be run in different modes: `train / train_and_eval / eval`. Run `official/nlp/train.py` and specify which mode you wish to execute.

Data processing

The script to process training data is the same as the BERT. Please check out the instructions.

The sentence piece vocabulary file can be downloaded here.

GLUE

The following commands will train and evaluate a model on GLUE datasets on TPUs. If you are using GPUs, just remove the `--tpu` flag and set `runtime.distribution_strategy` to `mirrored` to use the `tf.distribute.MirroredStrategy`.

```
INIT_CKPT=???
TRAIN_FILE=???
EVAL_FILE=???

python3 official/nlp/train.py \
   --experiment_type=bigbird/glue \
   --config_file=experiments/glue_mnli_matched.yaml \
   --params_override=task.init_checkpoint=${INIT_CKPT} \
   --params_override=runtime.distribution_strategy=tpu \
   --params_override=task.train_data.input_path=${TRAIN_FILE},task.validation_data.input_pat
   --tpu=??? \
   --mode=train_and_eval
```

SQuAD

The following commands will train and evaluate a model on SQuAD datasets.

```
VOCAB_FILE=???
TRAIN_FILE=???
EVAL_FILE=???
python3 official/nlp/train.py \
   --experiment_type=bigbird/squad \
   --config_file=third_party/tensorflow_models/official/nlp/projects/bigbird/experiments/squ
   --params_override=task.init_checkpoint=${INIT_CKPT} \
   --params_override=task.train_data.input_path=${TRAIN_FILE},task.validation_data.input_pat
   --params_override=runtime.distribution_strategy=tpu \
   --tpu=??? \
   --mode=train_and_eval
```

Checkpoints

Model	Configuration	Training Data	Checkpoint	Metrics
BigBird base	12 layer, 1024<= sequence length <= 4096	Wiki + Books + CC-News + Stories (part of Common Crawl)	bigbird_base	Squad v1 F1 91.3, Triv-i-aQA F1 79.8