Faster Transformer for Text Classifier

The comparison guide for Faster Transformer in BERT

Model

1. Modify the original code of bert-master:
    - Copy the code of tensorflow_bert to Bert-master path.
    - Copy the file of Faster Transformer to Bert-master path.
    - Collection bert-base model **uncased_L-12_H-768_A-12** and test classifier data set **IMDB** by processing to `.tsv` format
    - Add the code of loding the train/dev/test data set in `run_classifier.py`:

```
class ImdbProcessor(DataProcessor):
  """Processor for the IMDB data set."""

  def get_train_examples(self, data_dir):
    """See base class."""
    return self._create_examples(
        self._read_tsv(os.path.join(data_dir, "train.tsv")), "train")
  def get_dev_examples(self, data_dir):
    """See base class."""
    return self._create_examples(
        self._read_tsv(os.path.join(data_dir, "dev.tsv")), "dev")
  def get_test_examples(self, data_dir):
    """See base class."""
    return self._create_examples(
        self._read_tsv(os.path.join(data_dir, "test.tsv")), "test")
  def get_labels(self):
    """See base class."""
    return ["pos", "neg"]
  def _create_examples(self, lines, set_type):
    """Creates examples for the training and dev sets."""
    # get the examples of IMDB data
    examples = []
    for (i, line) in enumerate(lines):
      if set_type == "test":
        continue
      guid = "%s-%s" % (set_type, i)
      if set_type == "test":
        text_a = tokenization.convert_to_unicode(line[1])
```

```
        label = "0"
      else:
        text_a = tokenization.convert_to_unicode(line[1])
        label = tokenization.convert_to_unicode(line[0])
      examples.append(
          InputExample(guid=guid, text_a=text_a, text_b=None,
label=label))
    return examples
```

- Add the code of counting the time of train/evaluation/predict in `run_classifier.py`:

```
if FLAGS.do_train:
    train_file = os.path.join(FLAGS.output_dir, "train.tf_record")
    file_based_convert_examples_to_features(
        train_examples, label_list, FLAGS.max_seq_length, tokenizer,
train_file)
    tf.logging.info("***** Running training *****")
    tf.logging.info("  Num examples = %d", len(train_examples))
    tf.logging.info("  Batch size = %d", FLAGS.train_batch_size)
    tf.logging.info("  Num steps = %d", num_train_steps)
    train_input_fn = file_based_input_fn_builder(
        input_file=train_file,
        seq_length=FLAGS.max_seq_length,
        is_training=True,
        drop_remainder=True)
    #  counting the time of train
    start = time.time()
    estimator.train(input_fn=train_input_fn, max_steps=num_train_steps)
    elapsed = time.time() - start
    print("training finished, time used:{},average {} per
sample".format(elapsed, elapsed/len(train_examples)))
```

2. Environment requirements:
   - Create environment:

```
conda create -n fastertf2 python=3.6
source activate fastertf2
```

   - CMake >= 3.8
     - pip install CMake -i https://pypi.douban.com/simple
   - CUDA 10.0 && CUDNN 7.3.1 && NVIDIA RTX2080Ti
     - Install：https://blog.csdn.net/qq_39418067/article/details/87978848

        - check：`nvcc -V` && `cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A 2`
    - Tensorflow 1.13
        - pip install Tensorflow-gpu==1.13.1 -i https://pypi.douban.com/simple

3. Build with Release:

```
$ cd text_classifier/faster_transformer/fastertf_bert/
$ mkdir -p build
$ cd build
$ cmake -DSM=60 -DCMAKE_BUILD_TYPE=Release -DBUILD_TF=ON -
DTF_PATH=/home/jovenchu/anaconda3/envs/fastertf2/lib/python3.6/site-
packages/tensorflow .. # Tensorflow mode
$ make
```

    - Generate optimized **GEMM** algorithm file. For batch_size=16, sequence length=128, head_num=12, size_per_head=64，use the script below

```
$ ./bin/gemm_fp32 16 128 12 64
```

    - Inferencing

```
$ export
BERT_BASE_DIR='/home/jovenchu/text_classifier/faster_transformer/uncased_
L-12_H-768_A-12'
$ export
IMDB_DIR='/home/jovenchu/text_classifier/faster_transformer/data'
```

4. Running:
    - Training：We can't use Faster Transformer to training model because of the Faster Transformer only support for FP16 (Half precision)

```
$ python run_classifier.py   --task_name=Imdb   --do_train=true   --
data_dir=$IMDB_DIR   --vocab_file=$BERT_BASE_DIR/vocab.txt   --
bert_config_file=$BERT_BASE_DIR/bert_config.json   --max_seq_length=128
--eval_batch_size=16   --output_dir=imdb_output
```

    - Evaluation：
        - FP32 Tensorflow checkpoint files cannot be used directly for FP16 inference, we can convert its data type to FP16 in advance. The `ckpt_type_convert.py` script is provided for checkpoint data type conversion.

```
$ python ckpt_type_convert.py --
init_checkpoint=imdb_output/model.ckpt-125 --
fp16_checkpoint=imdb_output/fp16_model.ckpt
```

- Running Evaluation code:

```
$ python run_classifier.py   --task_name=Imdb   --do_eval=true   --
data_dir=$IMDB_DIR   --vocab_file=$BERT_BASE_DIR/vocab.txt   --
bert_config_file=$BERT_BASE_DIR/bert_config.json   --
init_checkpoint=imdb_output/model.ckpt-125   --max_seq_length=128   -
-eval_batch_size=16   --output_dir=imdb_output

$ python run_classifier_fastertf.py   --task_name=Imdb   --
do_eval=true   --data_dir=$IMDB_DIR   --
vocab_file=$BERT_BASE_DIR/vocab.txt   --
bert_config_file=$BERT_BASE_DIR/bert_config.json   --
init_checkpoint=imdb_output/fp16_model.ckpt   --max_seq_length=128
--eval_batch_size=16   --output_dir=imdb_output   --floatx=float16
```

Result

1. Parameter setting：
   - max_seq_length：128
   - train_batch_size：16
   - eval_batch_size：16
   - predict_batch_size：16
   - learning_rate：5e-5
   - num_train_epochs：1.0
   - save_checkpoints_steps：100
2. Result:
   - Bert_train:

     INFO:tensorflow:***** Running training *****

     INFO:tensorflow: Num examples = 2000

     INFO:tensorflow: Batch size = 16

     INFO:tensorflow: Num steps = 125

     INFO:tensorflow:Loss for final step: 0.6994.

     training finished, time used:57.629658222198486,average 0.028814829111099245 per sample
   - Bert_evaluation：

evaluation finished, time used:11.677468538284302,average 0.005838734269142151 per sample

INFO:tensorflow:***** Eval results *****

INFO:tensorflow: eval_accuracy = 0.5

INFO:tensorflow: eval_loss = 0.69381195

INFO:tensorflow: global_step = 375

INFO:tensorflow: loss = 0.69381195

- fastertf_evaluation：

  evaluation finished, time used:5.24286961555481,average 0.0026214348077774046 per sample

  INFO:tensorflow:***** Eval results *****

  INFO:tensorflow: eval_accuracy = 0.5

  INFO:tensorflow: eval_loss = 0.69376516

  INFO:tensorflow: global_step = 375

  INFO:tensorflow: loss = 0.69367576

Summary of experimental results：

Task classification	Sample	Total time	Time per sample
Bert_train	2000	57.63 s	0.029 s/sample
Bert_evaluation	2000	11.68 s	0.0058 s/sample
Faster TF_evaluation	2000	**5.25 s**	**0.0026 s/sample**

Note: The experimental configuration is 11G Nvidia RTX2080Ti, Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz, 16G RAM, 2T hard disk

3. others