Fine-tuning BERT on SQuAD1.0 with relative position embeddings

The following examples show how to fine-tune BERT models with different relative position embeddings. The BERT model `bert-base-uncased` was pretrained with default absolute position embeddings. We provide the following pretrained models which were pre-trained on the same training data (BooksCorpus and English Wikipedia) as in the BERT model training, but with different relative position embeddings.

- `zhiheng-huang/bert-base-uncased-embedding-relative-key`, trained from scratch with relative embedding proposed by Shaw et al., Self-Attention with Relative Position Representations
- `zhiheng-huang/bert-base-uncased-embedding-relative-key-query`, trained from scratch with relative embedding method 4 in Huang et al. Improve Transformer Models with Better Relative Position Embeddings
- `zhiheng-huang/bert-large-uncased-whole-word-masking-embedding-relative-key-query`, fine-tuned from model `bert-large-uncased-whole-word-masking` with 3 additional epochs with relative embedding method 4 in Huang et al. Improve Transformer Models with Better Relative Position Embeddings

Base models fine-tuning

```
export CUDA_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
python -m torch.distributed.launch --nproc_per_node=8 ./examples/question-
answering/run_squad.py \
    --model_name_or_path zhiheng-huang/bert-base-uncased-embedding-relative-key-
query \
    --dataset_name squad \
    --do_train \
    --do_eval \
    --learning_rate 3e-5 \
    --num_train_epochs 2 \
    --max_seq_length 512 \
    --doc_stride 128 \
    --output_dir relative_squad \
    --per_device_eval_batch_size=60 \
    --per_device_train_batch_size=6
```

Training with the above command leads to the following results. It boosts the BERT default from f1 score of 88.52 to 90.54.

```
'exact': 83.6802270577105, 'f1': 90.54772098174814
```

The change of `max_seq_length` from 512 to 384 in the above command leads to the f1 score of 90.34. Replacing the above model `zhiheng-huang/bert-base-uncased-embedding-relative-key-query` with `zhiheng-huang/bert-base-uncased-embedding-relative-key` leads to the f1 score of 89.51. The changing of 8 gpus to one gpu training leads to the f1 score of 90.71.

Large models fine-tuning

```
export CUDA_VISIBLE_DEVICES=0,1,2,3,4,5,6,7
python -m torch.distributed.launch --nproc_per_node=8 ./examples/question-
answering/run_squad.py \
    --model_name_or_path zhiheng-huang/bert-large-uncased-whole-word-masking-
```

```
embedding-relative-key-query \
    --dataset_name squad \
    --do_train \
    --do_eval \
    --learning_rate 3e-5 \
    --num_train_epochs 2 \
    --max_seq_length 512 \
    --doc_stride 128 \
    --output_dir relative_squad \
    --per_gpu_eval_batch_size=6 \
    --per_gpu_train_batch_size=2 \
    --gradient_accumulation_steps 3
```

Training with the above command leads to the f1 score of 93.52, which is slightly better than the f1 score of 93.15 for `bert-large-uncased-whole-word-masking`.

Distributed training

Here is an example using distributed training on 8 V100 GPUs and Bert Whole Word Masking uncased model to reach a F1 > 93 on SQuAD1.1:

```
python -m torch.distributed.launch --nproc_per_node=8 ./examples/question-
answering/run_squad.py \
    --model_name_or_path bert-large-uncased-whole-word-masking \
    --dataset_name squad \
    --do_train \
    --do_eval \
    --learning_rate 3e-5 \
    --num_train_epochs 2 \
    --max_seq_length 384 \
    --doc_stride 128 \
    --output_dir ./examples/models/wwm_uncased_finetuned_squad/ \
    --per_device_eval_batch_size=3   \
    --per_device_train_batch_size=3   \
```

Training with the previously defined hyper-parameters yields the following results:

```
f1 = 93.15
exact_match = 86.91
```

This fine-tuned model is available as a checkpoint under the reference `bert-large-uncased-whole-word-masking-finetuned-squad`.

Results

Larger batch size may improve the performance while costing more memory.

Results for SQuAD1.0 with the previously defined hyper-parameters:

```
{
"exact": 85.45884578997162,
"f1": 92.5974600601065,
"total": 10570,
"HasAns_exact": 85.45884578997162,
"HasAns_f1": 92.59746006010651,
"HasAns_total": 10570
}
```

Results for SQuAD2.0 with the previously defined hyper-parameters:

```
{
"exact": 80.4177545691906,
"f1": 84.07154997729623,
"total": 11873,
"HasAns_exact": 76.73751686909581,
"HasAns_f1": 84.05558584352873,
"HasAns_total": 5928,
"NoAns_exact": 84.0874684608915,
"NoAns_f1": 84.0874684608915,
"NoAns_total": 5945
}
```