Question answering

This folder contains several scripts that showcase how to fine-tune a 🤗 Transformers model on a question answering dataset, like SQuAD.

Trainer-based scripts

The `run_qa.py`, `run_qa_beam_search.py` and `run_seq2seq_qa.py` leverage the 🤗 Trainer for fine-tuning.

Fine-tuning BERT on SQuAD1.0

The `run_qa.py` script allows to fine-tune any model from our hub (as long as its architecture has a `ForQuestionAnswering` version in the library) on a question-answering dataset (such as SQuAD, or any other QA dataset available in the `datasets` library, or your own csv/jsonlines files) as long as they are structured the same way as SQuAD. You might need to tweak the data processing inside the script if your data is structured differently.

Note: This script only works with models that have a fast tokenizer (backed by the 🤗 Tokenizers library) as it uses special features of those tokenizers. You can check if your favorite model has a fast tokenizer in this table, if it doesn't you can still use the old version of the script which can be found here.

Note that if your dataset contains samples with no possible answers (like SQuAD version 2), you need to pass along the flag `--version_2_with_negative`.

This example code fine-tunes BERT on the SQuAD1.0 dataset. It runs in 24 min (with BERT-base) or 68 min (with BERT-large) on a single tesla V100 16GB.

```
python run_qa.py \
  --model_name_or_path bert-base-uncased \
  --dataset_name squad \
  --do_train \
  --do_eval \
  --per_device_train_batch_size 12 \
  --learning_rate 3e-5 \
  --num_train_epochs 2 \
  --max_seq_length 384 \
  --doc_stride 128 \
  --output_dir /tmp/debug_squad/
```

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

```
f1 = 88.52
exact_match = 81.22
```

Fine-tuning XLNet with beam search on SQuAD

The `run_qa_beam_search.py` script is only meant to fine-tune XLNet, which is a special encoder-only Transformer model. The example code below fine-tunes XLNet on the SQuAD1.0 and SQuAD2.0 datasets.

Command for SQuAD1.0:

```
python run_qa_beam_search.py \
    --model_name_or_path xlnet-large-cased \
    --dataset_name squad \
    --do_train \
    --do_eval \
    --learning_rate 3e-5 \
    --num_train_epochs 2 \
    --max_seq_length 384 \
    --doc_stride 128 \
    --output_dir ./wwm_cased_finetuned_squad/ \
    --per_device_eval_batch_size=4  \
    --per_device_train_batch_size=4   \
    --save_steps 5000
```

Command for SQuAD2.0:

```
export SQUAD_DIR=/path/to/SQUAD

python run_qa_beam_search.py \
    --model_name_or_path xlnet-large-cased \
    --dataset_name squad_v2 \
    --do_train \
    --do_eval \
    --version_2_with_negative \
    --learning_rate 3e-5 \
    --num_train_epochs 4 \
    --max_seq_length 384 \
    --doc_stride 128 \
    --output_dir ./wwm_cased_finetuned_squad/ \
    --per_device_eval_batch_size=2  \
    --per_device_train_batch_size=2   \
    --save_steps 5000
```

Fine-tuning T5 on SQuAD2.0

The `run_seq2seq_qa.py` script is meant for encoder-decoder (also called seq2seq) Transformer models, such as T5 or BART. These models are generative, rather than discriminative. This means that they learn to generate the correct answer, rather than predicting the start and end position of the tokens of the answer.

This example code fine-tunes T5 on the SQuAD2.0 dataset.

```
python run_seq2seq_qa.py \
  --model_name_or_path t5-small \
  --dataset_name squad_v2 \
  --context_column context \
  --question_column question \
  --answer_column answer \
  --do_train \
  --do_eval \
  --per_device_train_batch_size 12 \
```

```
  --learning_rate 3e-5 \
  --num_train_epochs 2 \
  --max_seq_length 384 \
  --doc_stride 128 \
  --output_dir /tmp/debug_seq2seq_squad/
```

Accelerate-based scripts

Based on the scripts `run_qa_no_trainer.py` and `run_qa_beam_search_no_trainer.py`.

Like `run_qa.py` and `run_qa_beam_search.py`, these scripts allow you to fine-tune any of the models supported on a SQuAD or a similar dataset, the main difference is that this script exposes the bare training loop, to allow you to quickly experiment and add any customization you would like. It offers less options than the script with `Trainer` (for instance you can easily change the options for the optimizer or the dataloaders directly in the script), but still run in a distributed setup, on TPU and supports mixed precision by leveraging the 🤗 `Accelerate` library.

You can use the script normally after installing it:

```
pip install accelerate
```

then

```
python run_qa_no_trainer.py \
  --model_name_or_path bert-base-uncased \
  --dataset_name squad \
  --max_seq_length 384 \
  --doc_stride 128 \
  --output_dir ~/tmp/debug_squad
```

You can then use your usual launchers to run in it in a distributed environment, but the easiest way is to run

```
accelerate config
```

and reply to the questions asked. Then

```
accelerate test
```

that will check everything is ready for training. Finally, you can launch training with

```
accelerate launch run_qa_no_trainer.py \
  --model_name_or_path bert-base-uncased \
  --dataset_name squad \
  --max_seq_length 384 \
  --doc_stride 128 \
  --output_dir ~/tmp/debug_squad
```

This command is the same and will work for:

- a CPU-only setup

- a setup with one GPU
- a distributed training with several GPUs (single or multi node)
- a training on TPUs

Note that this library is in alpha release so your feedback is more than welcome if you encounter any problem using it.