Language model training

Fine-tuning (or training from scratch) the library models for language modeling on a text dataset for GPT, GPT-2, ALBERT, BERT, DistilBERT, RoBERTa, XLNet... GPT and GPT-2 are trained or fine-tuned using a causal language modeling (CLM) loss while ALBERT, BERT, DistilBERT and RoBERTa are trained or fine-tuned using a masked language modeling (MLM) loss. XLNet uses permutation language modeling (PLM), you can find more information about the differences between those objectives in our model summary.

There are two sets of scripts provided. The first set leverages the Trainer API. The second set with `no_trainer` in the suffix uses a custom training loop and leverages the 🤗 Accelerate library . Both sets use the 🤗 Datasets library. You can easily customize them to your needs if you need extra processing on your datasets.

Note: The old script `run_language_modeling.py` is still available here.

The following examples, will run on datasets hosted on our hub or with your own text files for training and validation. We give examples of both below.

GPT-2/GPT and causal language modeling

The following example fine-tunes GPT-2 on WikiText-2. We're using the raw WikiText-2 (no tokens were replaced before the tokenization). The loss here is that of causal language modeling.

```
python run_clm.py \
    --model_name_or_path gpt2 \
    --dataset_name wikitext \
    --dataset_config_name wikitext-2-raw-v1 \
    --per_device_train_batch_size 8 \
    --per_device_eval_batch_size 8 \
    --do_train \
    --do_eval \
    --output_dir /tmp/test-clm
```

This takes about half an hour to train on a single K80 GPU and about one minute for the evaluation to run. It reaches a score of ~20 perplexity once fine-tuned on the dataset.

To run on your own training and validation files, use the following command:

```
python run_clm.py \
    --model_name_or_path gpt2 \
    --train_file path_to_train_file \
    --validation_file path_to_validation_file \
    --per_device_train_batch_size 8 \
    --per_device_eval_batch_size 8 \
    --do_train \
    --do_eval \
    --output_dir /tmp/test-clm
```

This uses the built in HuggingFace `Trainer` for training. If you want to use a custom training loop, you can utilize or adapt the `run_clm_no_trainer.py` script. Take a look at the script for a list of supported arguments. An example is shown below:

```
python run_clm_no_trainer.py \
    --dataset_name wikitext \
    --dataset_config_name wikitext-2-raw-v1 \
    --model_name_or_path gpt2 \
    --output_dir /tmp/test-clm
```

RoBERTa/BERT/DistilBERT and masked language modeling

The following example fine-tunes RoBERTa on WikiText-2. Here too, we're using the raw WikiText-2. The loss is different as BERT/RoBERTa have a bidirectional mechanism; we're therefore using the same loss that was used during their pre-training: masked language modeling.

In accordance to the RoBERTa paper, we use dynamic masking rather than static masking. The model may, therefore, converge slightly slower (over-fitting takes more epochs).

```
python run_mlm.py \
    --model_name_or_path roberta-base \
    --dataset_name wikitext \
    --dataset_config_name wikitext-2-raw-v1 \
    --per_device_train_batch_size 8 \
    --per_device_eval_batch_size 8 \
    --do_train \
    --do_eval \
    --output_dir /tmp/test-mlm
```

To run on your own training and validation files, use the following command:

```
python run_mlm.py \
    --model_name_or_path roberta-base \
    --train_file path_to_train_file \
    --validation_file path_to_validation_file \
    --per_device_train_batch_size 8 \
    --per_device_eval_batch_size 8 \
    --do_train \
    --do_eval \
    --output_dir /tmp/test-mlm
```

If your dataset is organized with one sample per line, you can use the `--line_by_line` flag (otherwise the script concatenates all texts and then splits them in blocks of the same length).

This uses the built in HuggingFace `Trainer` for training. If you want to use a custom training loop, you can utilize or adapt the `run_mlm_no_trainer.py` script. Take a look at the script for a list of supported arguments. An example is shown below:

```
python run_mlm_no_trainer.py \
    --dataset_name wikitext \
    --dataset_config_name wikitext-2-raw-v1 \
    --model_name_or_path roberta-base \
    --output_dir /tmp/test-mlm
```

Note: On TPU, you should use the flag `--pad_to_max_length` in conjunction with the `--line_by_line` flag to make sure all your batches have the same length.

Whole word masking

This part was moved to `examples/research_projects/mlm_wwm`.

XLNet and permutation language modeling

XLNet uses a different training objective, which is permutation language modeling. It is an autoregressive method to learn bidirectional contexts by maximizing the expected likelihood over all permutations of the input sequence factorization order.

We use the `--plm_probability` flag to define the ratio of length of a span of masked tokens to surrounding context length for permutation language modeling.

The `--max_span_length` flag may also be used to limit the length of a span of masked tokens used for permutation language modeling.

Here is how to fine-tune XLNet on wikitext-2:

```
python run_plm.py \
    --model_name_or_path=xlnet-base-cased \
    --dataset_name wikitext \
    --dataset_config_name wikitext-2-raw-v1 \
    --per_device_train_batch_size 8 \
    --per_device_eval_batch_size 8 \
    --do_train \
    --do_eval \
    --output_dir /tmp/test-plm
```

To fine-tune it on your own training and validation file, run:

```
python run_plm.py \
    --model_name_or_path=xlnet-base-cased \
    --train_file path_to_train_file \
    --validation_file path_to_validation_file \
    --per_device_train_batch_size 8 \
    --per_device_eval_batch_size 8 \
    --do_train \
    --do_eval \
    --output_dir /tmp/test-plm
```

If your dataset is organized with one sample per line, you can use the `--line_by_line` flag (otherwise the script concatenates all texts and then splits them in blocks of the same length).

Note: On TPU, you should use the flag `--pad_to_max_length` in conjunction with the `--line_by_line` flag to make sure all your batches have the same length.

Creating a model on the fly

When training a model from scratch, configuration values may be overridden with the help of `--config_overrides`:

```
python run_clm.py --model_type gpt2 --tokenizer_name gpt2 \ --
config_overrides="n_embd=1024,n_head=16,n_layer=48,n_positions=102" \
[...]
```

This feature is only available in `run_clm.py`, `run_plm.py` and `run_mlm.py`.