Language model training examples in streaming mode

The following examples showcase how to train a language model from scratch using the JAX/Flax backend.

JAX/Flax allows you to trace pure functions and compile them into efficient, fused accelerator code on both GPU and TPU. Models written in JAX/Flax are **immutable** and updated in a purely functional way which enables simple and efficient model parallelism.

All of the following examples make use of dataset streaming, therefore allowing to train models on massive datasets without ever having to download the full dataset.

Masked language modeling

In the following, we demonstrate how to train a bi-directional transformer model using masked language modeling objective as introduced in BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. More specifically, we demonstrate how JAX/Flax and dataset streaming can be leveraged to pre-train `roberta-base` in English on a single TPUv3-8 pod for 10000 update steps.

The example script uses the 🤗 Datasets library. You can easily customize them to your needs if you need extra processing on your datasets.

Let's start by creating a model repository to save the trained model and logs. Here we call the model `"english-roberta-base-dummy"`, but you can change the model name as you like.

You can do this either directly on huggingface.co (assuming that you are logged in) or via the command line:

```
huggingface-cli repo create english-roberta-base-dummy
```

Next we clone the model repository to add the tokenizer and model files.

```
git clone https://huggingface.co/<your-username>/english-roberta-base-dummy
```

To ensure that all tensorboard traces will be uploaded correctly, we need to track them. You can run the following command inside your model repo to do so.

```
cd english-roberta-base-dummy
git lfs track "*tfevents*"
```

Great, we have set up our model repository. During training, we will automatically push the training logs and model weights to the repo.

Next, let's add a symbolic link to the `run_mlm_flax.py`.

```
export MODEL_DIR="./english-roberta-base-dummy"
ln -s ~/transformers/examples/research_projects/jax-projects/dataset-
streaming/run_mlm_flax_stream.py ./
```

Copy config and tokenizer of existing model

In this example, we will simply copy an existing config and tokenizer in English. You can run the following code in a Python shell to do so.

```python
from transformers import RobertaTokenizerFast, RobertaConfig

model_dir = "./english-roberta-base-dummy"

tokenizer = RobertaTokenizerFast.from_pretrained("roberta-base")
config = RobertaConfig.from_pretrained("roberta-base")

tokenizer.save_pretrained(model_dir)
config.save_pretrained(model_dir)
```

Train model

Next we can run the example script to pretrain the model. Compared to the default `run_mlm_flax`, we introduced 4 new training settings:

- `num_train_steps` - how many update steps should be run.
- `num_eval_samples` - how many training samples should be taken for evaluation.
- `logging_steps` - at what rate should the training loss be logged.
- `eval_steps` - at what rate should evaluation be run. 10K update steps

```
./run_mlm_flax_stream.py \
    --output_dir="${MODEL_DIR}" \
    --model_type="roberta" \
    --config_name="${MODEL_DIR}" \
    --tokenizer_name="${MODEL_DIR}" \
    --dataset_name="oscar" \
    --dataset_config_name="unshuffled_deduplicated_en" \
    --max_seq_length="128" \
    --per_device_train_batch_size="128" \
    --per_device_eval_batch_size="128" \
    --learning_rate="3e-4" \
    --warmup_steps="1000" \
    --overwrite_output_dir \
    --adam_beta1="0.9" \
    --adam_beta2="0.98" \
    --num_train_steps="10000" \
    --num_eval_samples="5000" \
    --logging_steps="250" \
    --eval_steps="1000" \
    --push_to_hub
```