Token classification

PyTorch version, no Trainer

Fine-tuning (m)LUKE for token classification task such as Named Entity Recognition (NER), Parts-of-speech tagging (POS) or phrase extraction (CHUNKS). You can easily customize it to your needs if you need extra processing on your datasets.

It will either run on a datasets hosted on our hub or with your own text files for training and validation, you might just need to add some tweaks in the data preprocessing.

The script can be run in a distributed setup, on TPU and supports mixed precision by the mean of the 🤗 `Accelerate` library. You can use the script normally after installing it:

```
pip install accelerate
```

then to train English LUKE on CoNLL2003:

```
export TASK_NAME=ner

python run_luke_ner_no_trainer.py \
  --model_name_or_path studio-ousia/luke-base \
  --dataset_name conll2003 \
  --task_name $TASK_NAME \
  --max_length 128 \
  --per_device_train_batch_size 32 \
  --learning_rate 2e-5 \
  --num_train_epochs 3 \
  --output_dir /tmp/$TASK_NAME/
```

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

```
export TASK_NAME=ner

accelerate launch run_ner_no_trainer.py \
  --model_name_or_path studio-ousia/luke-base \
  --dataset_name conll2003 \
  --task_name $TASK_NAME \
  --max_length 128 \
  --per_device_train_batch_size 32 \
  --learning_rate 2e-5 \
```

```
    --num_train_epochs 3 \
    --output_dir /tmp/$TASK_NAME/
```

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.