Distil*

Author: @VictorSanh

This folder contains the original code used to train Distil* as well as examples showcasing how to use DistilBERT, DistilRoBERTa and DistilGPT2.

January 20, 2020 - Bug fixing We have recently discovered and fixed a bug in the evaluation of our `run_*.py` scripts that caused the reported metrics to be over-estimated on average. We have updated all the metrics with the latest runs.

December 6, 2019 - Update We release **DistilmBERT**: 92% of `bert-base-multilingual-cased` on XNLI. The model supports 104 different languages listed here.

November 19, 2019 - Update We release German **DistilBERT**: 98.8% of `bert-base-german-dbmdz-cased` on NER tasks.

October 23, 2019 - Update We release **DistilRoBERTa**: 95% of `RoBERTa-base`'s performance on GLUE, twice as fast as RoBERTa while being 35% smaller.

October 3, 2019 - Update We release our NeurIPS workshop paper explaining our approach on **DistilBERT**. It includes updated results and further experiments. We applied the same method to GPT2 and release the weights of **DistilGPT2**. DistilGPT2 is two times faster and 33% smaller than GPT2. **The paper supersedes our previous blogpost with a different distillation loss and better performances. Please use the paper as a reference when comparing/reporting results on DistilBERT.**

September 19, 2019 - Update: We fixed bugs in the code and released an updated version of the weights trained with a modification of the distillation loss. DistilBERT now reaches 99% of `BERT-base`'s performance on GLUE, and 86.9 F1 score on SQuAD v1.1 dev set (compared to 88.5 for `BERT-base`). We will publish a formal write-up of our approach in the near future!

What is Distil*

Distil* is a class of compressed models that started with DistilBERT. DistilBERT stands for Distilled-BERT. DistilBERT is a small, fast, cheap and light Transformer model based on Bert architecture. It has 40% less parameters than `bert-base-uncased`, runs 60% faster while preserving 97% of BERT's performances as measured on the GLUE language understanding benchmark. DistilBERT is trained using knowledge distillation, a technique to compress a large model called the teacher into a smaller model called the student. By distillating Bert, we obtain a smaller Transformer model that bears a lot of similarities with the original BERT model while being lighter, smaller and faster to run. DistilBERT is thus an interesting option to put large-scaled trained Transformer model into production.

We have applied the same method to other Transformer architectures and released the weights: - GPT2: on the WikiText-103 benchmark, GPT2 reaches a perplexity on the test set of 16.3 compared to 21.1 for **DistilGPT2** (after fine-tuning on the train set). - RoBERTa: **DistilRoBERTa** reaches 95% of `RoBERTa-base`'s performance on GLUE while being twice faster and 35% smaller. - German BERT: **German DistilBERT** reaches 99% of `bert-base-german-dbmdz-cased`'s performance on German NER (CoNLL-2003). - Multilingual BERT: **DistilmBERT** reaches 92% of Multilingual BERT's performance on XNLI while being twice faster and 25% smaller. The model supports 104 languages listed here.

For more information on DistilBERT, please refer to our NeurIPS workshop paper.

Here are the results on the dev sets of GLUE:

Model	Macro-score	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
BERT-base-uncased	**79.5**	56.3	84.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT-base-uncased	**77.0**	51.3	82.1	87.5	89.2	88.5	59.9	91.3	86.9	56.3
BERT-base-cased	**78.2**	58.2	83.9	87.8	91.0	89.2	66.1	91.7	89.2	46.5
DistilBERT-base-cased	**75.9**	47.2	81.5	85.6	88.2	87.8	60.6	90.4	85.5	56.3
—	—	—	—	—	—	—	—	—	—	—
RoBERTa-base (reported)	**83.2/86.6**[2]	63.6	87.6	90.2	92.8	91.9	78.7	94.8	91.2	57.73
DistilRoBERTa[1]	**79.0/84.0**[3]	59.3	84.0	86.6	90.8	89.4	67.9	92.5	88.3	52.1

1 We did not use the MNLI checkpoint for fine-tuning but directly perform transfer learning on the pre-trained DistilRoBERTa.

2 Macro-score computed without WNLI.

3 We compute this score ourselves for completeness.

Here are the results on the *test* sets for 6 of the languages available in XNLI. The results are computed in the zero shot setting (trained on the English portion and evaluated on the target language portion):

Model	English	Spanish	Chinese	German	Arabic	Urdu
mBERT base cased (computed)	82.1	74.6	69.1	72.3	66.4	58.5
mBERT base uncased (reported)	81.4	74.3	63.8	70.5	62.1	58.3
DistilmBERT	78.2	69.1	64.0	66.3	59.1	54.7

Setup

This part of the library has only be tested with Python3.6+. There are few specific dependencies to install before launching a distillation, you can install them with the command `pip install -r requirements.txt`.

Important note: The training scripts have been updated to support PyTorch v1.2.0 (there are breaking changes compared to v1.1.0).

How to use DistilBERT

Transformers includes five pre-trained Distil* models, currently only provided for English and German (we are investigating the possibility to train and release a multilingual version of DistilBERT):

- `distilbert-base-uncased`: DistilBERT English language model pretrained on the same data used to pretrain Bert (concatenation of the Toronto Book Corpus and full English Wikipedia) using distillation with the supervision of the `bert-base-uncased` version of Bert. The model has 6 layers, 768 dimension and 12 heads, totalizing 66M parameters.
- `distilbert-base-uncased-distilled-squad`: A finetuned version of `distilbert-base-uncased` finetuned using (a second step of) knowledge distillation on SQuAD 1.0. This model reaches a F1 score of 86.9 on the dev set (for comparison, Bert `bert-base-uncased` version reaches a 88.5 F1 score).
- `distilbert-base-cased`: DistilBERT English language model pretrained on the same data used to pretrain Bert (concatenation of the Toronto Book Corpus and full English Wikipedia) using distillation with the supervision of the `bert-base-cased` version of Bert. The model has 6 layers, 768 dimension and 12 heads, totalizing 65M parameters.
- `distilbert-base-cased-distilled-squad`: A finetuned version of `distilbert-base-cased` finetuned using (a second step of) knowledge distillation on SQuAD 1.0. This model reaches a F1 score of 87.1 on the

dev set (for comparison, Bert `bert-base-cased` version reaches a 88.7 F1 score).

- `distilbert-base-german-cased`: DistilBERT German language model pretrained on 1/2 of the data used to pretrain Bert using distillation with the supervision of the `bert-base-german-dbmdz-cased` version of German DBMDZ Bert. For NER tasks the model reaches a F1 score of 83.49 on the CoNLL-2003 test set (for comparison, `bert-base-german-dbmdz-cased` reaches a 84.52 F1 score), and a F1 score of 85.23 on the GermEval 2014 test set (`bert-base-german-dbmdz-cased` reaches a 86.89 F1 score).
- `distilgpt2`: DistilGPT2 English language model pretrained with the supervision of `gpt2` (the smallest version of GPT2) on OpenWebTextCorpus, a reproduction of OpenAI's WebText dataset. The model has 6 layers, 768 dimension and 12 heads, totalizing 82M parameters (compared to 124M parameters for GPT2). On average, DistilGPT2 is two times faster than GPT2.
- `distilroberta-base`: DistilRoBERTa English language model pretrained with the supervision of `roberta-base` solely on OpenWebTextCorpus, a reproduction of OpenAI's WebText dataset (it is ~4 times less training data than the teacher RoBERTa). The model has 6 layers, 768 dimension and 12 heads, totalizing 82M parameters (compared to 125M parameters for RoBERTa-base). On average DistilRoBERTa is twice as fast as Roberta-base.
- `distilbert-base-multilingual-cased`: DistilmBERT multilingual model pretrained with the supervision of `bert-base-multilingual-cased` on the concatenation of Wikipedia in 104 different languages. The model supports the 104 languages listed here. The model has 6 layers, 768 dimension and 12 heads, totalizing 134M parameters (compared to 177M parameters for mBERT-base). On average DistilmBERT is twice as fast as mBERT-base.

Using DistilBERT is very similar to using BERT. DistilBERT share the same tokenizer as BERT's `bert-base-uncased` even though we provide a link to this tokenizer under the `DistilBertTokenizer` name to have a consistent naming between the library models.

```
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-cased')
model = DistilBertModel.from_pretrained('distilbert-base-cased')

input_ids = torch.tensor(tokenizer.encode("Hello, my dog is cute")).unsqueeze(0)
outputs = model(input_ids)
last_hidden_states = outputs[0]  # The last hidden-state is the first element of the output
```

Similarly, using the other Distil* models simply consists in calling the base classes with a different pretrained checkpoint: - DistilBERT uncased: `model = DistilBertModel.from_pretrained('distilbert-base-uncased')` - DistilGPT2: `model = GPT2Model.from_pretrained('distilgpt2')` - DistilRoBERTa: `model = RobertaModel.from_pretrained('distilroberta-base')`

- DistilmBERT: `model = DistilBertModel.from_pretrained('distilbert-base-multilingual-cased')`

How to train Distil*

In the following, we will explain how you can train DistilBERT.

A. Preparing the data

The weights we release are trained using a concatenation of Toronto Book Corpus and English Wikipedia (same training data as the English version of BERT).

To avoid processing the data several time, we do it once and for all before the training. From now on, will suppose that you have a text file `dump.txt` which contains one sequence per line (a sequence being composed of one of several coherent sentences).

First, we will binarize the data, i.e. tokenize the data and convert each token in an index in our model's vocabulary.

```
python scripts/binarized_data.py \
    --file_path data/dump.txt \
    --tokenizer_type bert \
    --tokenizer_name bert-base-uncased \
    --dump_file data/binarized_text
```

Our implementation of masked language modeling loss follows XLM's one and smooths the probability of masking with a factor that put more emphasis on rare words. Thus we count the occurrences of each tokens in the data:

```
python scripts/token_counts.py \
    --data_file data/binarized_text.bert-base-uncased.pickle \
    --token_counts_dump data/token_counts.bert-base-uncased.pickle \
    --vocab_size 30522
```

B. Training

Training with distillation is really simple once you have pre-processed the data:

```
python train.py \
    --student_type distilbert \
    --student_config training_configs/distilbert-base-uncased.json \
    --teacher_type bert \
    --teacher_name bert-base-uncased \
    --alpha_ce 5.0 --alpha_mlm 2.0 --alpha_cos 1.0 --alpha_clm 0.0 --mlm \
    --freeze_pos_embs \
    --dump_path serialization_dir/my_first_training \
    --data_file data/binarized_text.bert-base-uncased.pickle \
    --token_counts data/token_counts.bert-base-uncased.pickle \
    --force # overwrites the `dump_path` if it already exists.
```

By default, this will launch a training on a single GPU (even if more are available on the cluster). Other parameters are available in the command line, please look in `train.py` or run `python train.py --help` to list them.

We highly encourage you to use distributed training for training DistilBERT as the training corpus is quite large. Here's an example that runs a distributed training on a single node having 4 GPUs:

```
export NODE_RANK=0
export N_NODES=1

export N_GPU_NODE=4
export WORLD_SIZE=4
export MASTER_PORT=<AN_OPEN_PORT>
export MASTER_ADDR=<I.P.>

pkill -f 'python -u train.py'

python -m torch.distributed.launch \
    --nproc_per_node=$N_GPU_NODE \
    --nnodes=$N_NODES \
    --node_rank $NODE_RANK \
    --master_addr $MASTER_ADDR \
    --master_port $MASTER_PORT \
    train.py \
        --force \
        --n_gpu $WORLD_SIZE \
        --student_type distilbert \
        --student_config training_configs/distilbert-base-uncased.json \
        --teacher_type bert \
        --teacher_name bert-base-uncased \
        --alpha_ce 0.33 --alpha_mlm 0.33 --alpha_cos 0.33 --alpha_clm 0.0 --mlm \
        --freeze_pos_embs \
        --dump_path serialization_dir/my_first_training \
        --data_file data/binarized_text.bert-base-uncased.pickle \
        --token_counts data/token_counts.bert-base-uncased.pickle
```

Tips: Starting distilled training with good initialization of the model weights is crucial to reach decent performance. In our experiments, we initialized our model from a few layers of the teacher (Bert) itself! Please refer to `scripts/extract.py` and `scripts/extract_distilbert.py` to create a valid initialization checkpoint and use `--student_pretrained_weights` argument to use this initialization for the distilled training!

Happy distillation!

Citation

If you find the resource useful, you should cite the following paper:

```
@inproceedings{sanh2019distilbert,
  title={DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter},
  author={Sanh, Victor and Debut, Lysandre and Chaumond, Julien and Wolf, Thomas},
  booktitle={NeurIPS EMC^2 Workshop},
  year={2019}
}
```