Token Dropping for Efficient BERT Pretraining

This is the official implementation of the token dropping method Pang et al. Token Dropping for Efficient BERT Pretraining. ACL 2022.

Token dropping aims to accelerate the pretraining of transformer models such as BERT without degrading its performance on downstream tasks. In particular, we drop unimportant tokens starting from an intermediate layer in the model, to make the model focus on important tokens more efficiently with its limited computational resources. The dropped tokens are later picked up by the last layer of the model, so that the model still produces full-length sequences. We leverage the already built-in masked language modeling (MLM) loss and its dynamics to identify unimportant tokens with practically no computational overhead. In our experiments, this simple approach reduces the pretraining cost of BERT by 25% while achieving slightly better overall fine-tuning performance on standard downstream tasks.

A BERT model pretrained using this token dropping method is not different to a BERT model pretrained in the conventional way: a BERT checkpoint pretrained with token dropping can be viewed and used as a normal BERT checkpoint, for finetuning etc. Thus, this README file only illustrates how to run token dropping for pretraining.

Requirements

The starter code requires Tensorflow. If you haven't installed it yet, follow the instructions on tensorflow.org. This code has been tested with Tensorflow 2.5.0. Going forward, we will continue to target the latest released version of Tensorflow.

Please verify that you have Python 3.6+ and Tensorflow 2.5.0 or higher installed by running the following commands:

```
python --version
python -c 'import tensorflow as tf; print(tf.__version__)'
```

Refer to the instructions here for using the model in this repo. Make sure to add the models folder to your Python path.

Then, you need to generate pretraining data. See [this instruction] (https://github.com/tensorflow/models/blob/27fb855b027ead16d2616dcb59c67409a2176b7f/official/legacy/bert/
training) on how to do that.

Train using the config file.

After you generated your pretraining data, run the following command to start pretraining:

```
PARAMS="task.train_data.input_data=/path/to/train/data"
PARAMS="${PARAMS},task.validation_data.input_path=/path/to/validation/data"
```

```
PARAMS="${PARAMS},runtime.distribution_strategy=tpu"

python3 train.py \
  --experiment=token_drop_bert/pretraining \
  --config_file=wiki_books_pretrain_sequence_pack.yaml \
  --config_file=bert_en_uncased_base_token_drop.yaml \
  --params_override=${PARAMS} \
  --tpu=local \
  --model_dir=/folder/to/hold/logs/and/models/ \
  --mode=train_and_eval
```

Implementation

We implement the encoder and layers using `tf.keras` APIs in NLP modeling library:

- masked_lm.py contains the BERT pretraining task.
- experiment_configs.py registers the token dropping experiment.
- encoder.py contains the BERT encoder that supports token dropping.
- encoder_config.py contains the config and method for instantiating the token dropping BERT encoder.
- train.py is the program entry.

Reference

Please cite our paper:

```
@article{hou2022token,
  title={Token Dropping for Efficient BERT Pretraining},
  author={Pang, Richard Yuanzhe and Hou, Le and Zhou, Tianyi and Wu, Yuexin and Song, Xinyin
  journal={arXiv preprint arXiv:2203.13240},
  year={2022}
}
```