Multi-doc News Headline Generation Model: NHNet

This repository contains TensorFlow 2.x implementation for NHNet [1] as well as instructions for producing the data we described in the paper.

Introduction

NHNet is a multi-doc news headline generation model. It extends a standard Transformer-based encoder-decoder model to multi-doc setting and relies on an article-level attention layer to capture information common to most (if not all) input news articles in a news cluster or story, and provide robustness against potential outliers in the input due to clustering quality.

Our academic paper [1] which describes NHNet in detail can be found here: https://arxiv.org/abs/2001.09386.

Dataset

Raw Data: One can download our multi-doc headline dataset which contains 369,940 news stories and 932,571 unique URLs. We split these stories into train (359,940 stories), validation (5,000 stories) and test set (5,000 stories) by timestamp.

More information, please checkout: https://github.com/google-research-datasets/NewSHead

Crawling

Unfortunately, we will not be able to release the pre-processed dataset that is exactly used in the paper. Users need to crawl the URLs and the recommended pre-processing is using an open-sourced library to download and parse the news content including title and leading paragraphs. For ease of this process, we provide a config of news-please that will crawl and extract news articles on a local machine.

First, install the `news-please` CLI (requires python 3.x)

```
$ pip3 install news-please==1.4.26
```

Next, run the crawler with our provided config and URL list

```
# Sets to path of the downloaded data folder.
$ DATA_FOLDER=/path/to/downloaded_dataset

# Uses CLI interface to crawl. We assume news_please subfolder contains the
# decompressed config.cfg and sitelist.hjson.
$ news-please -c $DATA_FOLDER/news_please
```

By default, it will store crawled articles under `/tmp/nhnet/`. To terminate the process press `CTRL+C`.

The crawling may take some days (48 hours in our test) and it depends on the network environment and #threads set in the config. As the crawling tool won't stop automatically, it is not straightforward to check the progress. We suggest to terminate the job if there are no new articles crawled in a short time period (e.g., 10 minutes) by running

```
$ find /tmp/nhnet -type f | wc -l
```

Please note that it is expected that some URLs are no longer available on the web as time goes by.

Data Processing

Given the crawled articles under `/tmp/nhnet/`, we would like to transform these textual articles into a set of `TFRecord` files containing serialized tensorflow.Example protocol buffers, with feature keys following the BERT [2] tradition but is extended for multiple text segments. We will later use these processed TFRecords for training and evaluation.

To do this, please first download a BERT pretrained checkpoint (`BERT-Base,Uncased` preferred for efficiency) and decompress the `tar.gz` file. We need the vocabulary file and later use the checkpoint for NHNet initialization.

Next, we can run the following data preprocess script which may take a few hours to read files and tokenize article content.

```
# Recall that we use DATA_FOLDER=/path/to/downloaded_dataset.
$ python3 raw_data_process.py \
    -crawled_articles=/tmp/nhnet \
    -vocab=/path/to/bert_checkpoint/vocab.txt \
    -do_lower_case=True \
    -len_title=15 \
    -len_passage=200 \
    -max_num_articles=5 \
    -data_folder=$DATA_FOLDER
```

This python script will export processed train/valid/eval files under `$DATA_FOLDER/processed/`.

Training

Please first install TensorFlow 2 and Tensorflow Model Garden following the requirments section.

CPU/GPU

```
$ python3 trainer.py \
    --mode=train_and_eval \
    --init_checkpoint=/path/to/bert_checkpoint/bert_model.ckpt \
    --params_override='init_from_bert2bert=false' \
    --train_file_pattern=$DATA_FOLDER/processed/train.tfrecord* \
    --model_dir=/path/to/output/model \
    --len_title=15 \
    --len_passage=200 \
    --num_nhnet_articles=5 \
    --model_type=nhnet \
    --train_batch_size=16 \
    --train_steps=10000 \
    --steps_per_loop=1 \
    --checkpoint_interval=100
```

TPU

```
$ python3 trainer.py \
    --mode=train_and_eval \
    --init_checkpoint=/path/to/bert_checkpoint/bert_model.ckpt \
    --params_override='init_from_bert2bert=false' \
    --train_file_pattern=$DATA_FOLDER/processed/train.tfrecord* \
    --model_dir=/path/to/output/model \
    --len_title=15 \
    --len_passage=200 \
    --num_nhnet_articles=5 \
    --model_type=nhnet \
    --train_batch_size=1024 \
    --train_steps=10000 \
    --steps_per_loop=1000 \
    --checkpoint_interval=1000 \
    --distribution_strategy=tpu \
    --tpu=grpc://${TPU_IP_ADDRESS}:8470
```

In the paper, we train more than 10k steps with batch size set as 1024 with TPU-v3-64.

Note that, `trainer.py` also supports `train` mode and continuous `eval` mode. For large scale TPU training, we recommend the have a process running the `train` mode and another process running the continuous `eval` mode which can runs on GPUs. This is the setting we commonly used for large-scale experiments, because `eval` will be non-blocking to the expensive training load.

Metrics

Note: the metrics reported by `evaluation.py` are approximated on word-piece level rather than the real string tokens. Some metrics like BLEU scores can be off.

We will release a colab to evaluate results on string-level soon.

References

[1] Xiaotao Gu, Yuning Mao, Jiawei Han, Jialu Liu, You Wu, Cong Yu, Daniel Finnie, Hongkun Yu, Jiaqi Zhai and Nicholas Zukoski "Generating Representative Headlines for News Stories": https://arxiv.org/abs/2001.09386. World Wide Web Conf. (WWW'2020).

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding": https://arxiv.org/abs/1810.04805.