code for our main model

Dependencies

- rouge 1.0.1
- numpy 1.21.6
- Pytorch 1.11.0

  > cuda 11.1
  >
  > ```
  > pip3 install torch --extra-index-url https://download.pytorch.org/whl/cu111
  > ```

- Python 3.7
- transformers 4.6
- text2vec 1.1.5

Data

From NLPCC 2022 shared task 6

数据集已放在代码文件夹中`/data`目录下，按照`280:20`的比例划分为训练集和测试集

```
-- data
 |---- train.jsonl
 |---- test.jsonl
```

Start

⬇️ For Training

Step 1: preprocess data for training

```
python preprocess_train.py
```

Step 2: preprocess data for scorer

```
python preprocess_scorer.py
```

Step 3: train scorer

```
python scorer_train.py
```

Step 4: fine-tune BART

```
python3 -u pipeline_train.py \
        --do_train \
        --src_lang zh_CN \
        --tgt_lang zh_CN \
        --train_filename data/processed/train_abstract.jsonl \
        --val_filename data/dev_flex_v2_deep.jsonl\
        --max_src_len 320 \
        --max_tgt_len 150 \
        --remark dta \
        --save_dir model/bart \
        --batch_size 10 \
        --num_train_epochs 10 \
        --skip_eval_epochs -1 \
        --learning_rate 2e-5
```

⬇️ For Testing

Step 5: preprocess data for test

```
python preprocess_test.py
```

Step 6: generate abstract using test data

```
python3 -u pipeline_test.py \
        --do_test \
        --src_lang zh_CN \
        --tgt_lang zh_CN \
        --test_filename data/proposal.jsonl \
        --max_src_len 320 \
        --max_tgt_len 150 \
        --remark dta \
        --batch_size 10 \
        --learning_rate 2e-5 \
        --pretrained_model_path model/bart/pre
```

Step 7: match generated results with corresponding dialogues

```
python process_test_result.py
```

Step 8: evaluate results

```
python evaluate.py
```

`Rouge-1`、`Rouge-2`、`Rouge-L` scores will be printed in the terminal

if using given preprocessed data, shall get:

Rouge-1	Rouge-2	Rouge-L
0.62515	0.29660	0.44279

Pretrained Models

1. MBART-50

Fine-tuned `MBART-50` is available here (extraction code: hasm), shall get:

```
-- pre
 |---- pytorch_model.bin
 |---- config.json
```

2. Scorer

Pre-trained `scorer` is available here (extraction code: 5iqs), shall get:

```
-- min_loss.pth
```

Architecture

the directory (for pre-trained models) should be organized as follows:

```
-- model
 |---- bart
 |    |---- pre
 |    |    |---- pytorch_model.bin
 |    |    |---- config.json
 |---- scorer
      |---- min_loss.pth
```

Preprocessed Data

Preprocessed data from `step 1`, `step 2`, `step 5`, `step 6` and `step 7`

```
-- data
 |---- processed
 |    |---- train_abstract.jsonl # data for fine-tuning bart (from step 1)
 |    |---- train_scorer.jsonl # data for training scorer (from step 2)
 |---- result
 |    |---- result.jsonl # (from step 7)
 |    |---- result.txt # (from step 6)
 |---- proposal.jsonl # (from step step 5)
 |---- train.jsonl # original data for training
 |---- test.jsonl # original data for testing
```

Date: 2022/05/31