TAS-BERT

Code and data for our paper Target-Aspect-Sentiment Joint Detection for Aspect-Based Sentiment Analysis" (AAAI 2020)

Our code is based on Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence (NAACL 2019)

Requirements

- pytorch: 1.0.1
- python: 3.6.8
- tensorflow: 1.13.1 (only for creating BERT-pytorch-model)
- pytorch-crf: 0.7.2
- numpy: 1.16.4
- nltk: 3.4.4
- sklearn: 0.21.2

Data Preprocessing

- Download uncased BERT-Based model, unpack, place the folder in the root directory and run `convert_tf_checkpoint_to_pytorch.py` to create BERT-pytorch-model.
- run following commands to get preprocessed data.

```
cd data
python data_preprocessing_for_TAS.py --dataset semeval2015
python data_preprocessing_for_TAS.py --dataset semeval2016
cd ../
```

The preprocessed data is in folders **semeval2015/three_joint/BIO**, **semeval2015/three_joint/TO**, **semeval2016/three_joint/BIO** and **semeval2016/three_joint/TO**. BIO and TO are the two tagging schemes mentioned in our paper.

The preprocessed data structure is as follows:

Key	Description
sentence_id	Id of the sentence.
yes_no	whether the sentence has corresponding sentiment in the corresponding aspect. The corresponding sentiment and aspect are given in *aspect_sentiment*.
aspect_sentiment	<aspect, sentiment> pair of this line, such as "food quality positive".
sentence	Content of the sentence.
ner_tags	label sequence for targets that have corresponding sentiment in the corresponding aspect. The corresponding sentiment and aspect are given in *aspect_sentiment*.

Code Structure

- `TAS_BERT_joint.py`: Program Runner.
- `modeling.py`: Program Models.
- `optimization.py`: Optimization for model.
- `processor.py`: Data Processor.
- `tokenization.py`: Tokenization, including three unknown-word-solutions.
- `evaluation_for_TSD_ASD_TASD.py`: evaluation for ASD, TASD and TSD tasks.
- `evaluation_for_AD_TD_TAD/`: The official evaluation tool for AD, TD and TAD tasks.
- `TAS_BERT_separate.py` and `evaluation_for_loss_separate.py`: Separate detection for Ablation Study.

Training & Testing

If you want to train and test a joint detection model, you can use the following command:

```
CUDA_VISIBLE_DEVICES=0 python TAS_BERT_joint.py \
--data_dir data/semeval2016/three_joint/BIO/ \
--output_dir results/semeval2016/three_joint/BIO/my_result \
--vocab_file uncased_L-12_H-768_A-12/vocab.txt \
--bert_config_file uncased_L-12_H-768_A-12/bert_config.json \
--init_checkpoint uncased_L-12_H-768_A-12/pytorch_model.bin \
--tokenize_method word_split \
--use_crf \
--eval_test \
--do_lower_case \
--max_seq_length 128 \
--train_batch_size 24 \
--eval_batch_size 8 \
--learning_rate 2e-5 \
--num_train_epochs 30.0
```

The test results for each epoch will be stored in test_ep_*.txt in the output folder.

Evaluation

If you want to evaluate the test result for each epoch, you can use the following commands:

Note: We chose the epoch which performed best on the TASD task, and evaluate the result on all the subtasks.

- If you want to evaluate on the TASD task, ASD task, TSD task ignoring implicit targets, and TSD task considering implicit targets, you can use the following command:

```
python evaluation_for_TSD_ASD_TASD.py \
--output_dir results/semeval2016/three_joint/BIO/my_result \
--num_epochs 30 \
--tag_schema BIO
```

☆The *tag_schema* bust be consistent with the contents in the *output_dir*, otherwise you will get error results.

"All tuples" correspond to "C1" in Table 3 of our paper.

"Only NULL tuples" correspond to "C2" in Table 3 of our paper.

"NO and pure O tag sequence" correspond to "C3" in Table 3 of our paper.

As for the TD, AD and TAD tasks, we use the evaluation tool provided by the SemEval2015 competition. The tool requires a Java environment.

- First, we should convert our test results into XML file format. You can use the following command:

```
cd evaluation_for_AD_TD_TAD
python change_pre_to_xml.py \
--gold_path ../data/semeval2016/three_joint/BIO/test_TAS.tsv \
--pre_path ../results/semeval2016/three_joint/BIO/my_result/test_ep_23.txt \
--gold_xml_file ABSA16_Restaurants_Test.xml \
--pre_xml_file pred_file_2016.xml \
--tag_schema BIO
```

  Note: the "test_ep_*.txt" is the best epoch on the TASD task.

  We will get a predication file in XML format: pred_file_2016.xml.

- If you want to evaluate on the AD task:

```
java -cp ./A.jar absa15.Do Eval ./pred_file_2016.xml
./ABSA16_Restaurants_Test.xml 1 0
```

- If you want to evaluate on the TD task:

```
java -cp ./A.jar absa15.Do Eval ./pred_file_2016.xml
./ABSA16_Restaurants_Test.xml 2 0
```

- If you want to evaluate on the TAD task:

```
java -cp ./A.jar absa15.Do Eval ./pred_file_2016.xml
./ABSA16_Restaurants_Test.xml 3 0
```

Ablation Study

If you want to try the separate models, please use the following commands:

```
CUDA_VISIBLE_DEVICES=0 python TAS_BERT_separate.py \
--data_dir data/semeval2016/three_joint/BIO/ \
--output_dir results/semeval2016/three_joint/BIO/my_result_AS \
--vocab_file uncased_L-12_H-768_A-12/vocab.txt \
--bert_config_file uncased_L-12_H-768_A-12/bert_config.json \
--init_checkpoint uncased_L-12_H-768_A-12/pytorch_model.bin \
--tokenize_method word_split \
--use_crf \
--subtask AS \
--eval_test \
--do_lower_case \
--max_seq_length 128 \
--train_batch_size 24 \
--eval_batch_size 8 \
--learning_rate 2e-5 \
--num_train_epochs 30.0
```

```
CUDA_VISIBLE_DEVICES=0 python TAS_BERT_separate.py \
--data_dir data/semeval2016/three_joint/BIO/ \
--output_dir results/semeval2016/three_joint/BIO/my_result_T \
--vocab_file uncased_L-12_H-768_A-12/vocab.txt \
--bert_config_file uncased_L-12_H-768_A-12/bert_config.json \
--init_checkpoint uncased_L-12_H-768_A-12/pytorch_model.bin \
--tokenize_method word_split \
--use_crf \
--subtask T \
--eval_test \
--do_lower_case \
--max_seq_length 128 \
--train_batch_size 24 \
--eval_batch_size 8 \
--learning_rate 2e-5 \
--num_train_epochs 30.0
```

```
python evaluation_for_loss_separate.py \
--output_dir_AS results/semeval2016/three_joint/BIO/my_result_AS \
--output_dir_T results/semeval2016/three_joint/BIO/my_result_T \
--num_epochs 30 \
--tag_schema BIO
```