DefSent+

The versions of our PyTorch and Huggingface transformers are “1.9.0+cu111” and “4.10.2” respectively for training DefSent+ encoders.

1) Released Datasets and Encoders:

The datasets (Oxford and WordNet) used for this paper are under the link below.

https://drive.google.com/drive/folders/1sZxAUKkkGnDJHRF2wmPG97khBaDXFHlf?usp=sharing

They are reformed from the datasets released by Ishiwatari et al. (2019) (wget http://www.tkl.iis.u-tokyo.ac.jp/~ishiwatari/naacl_data.zip). The datasets are in the form of “.csv”, with each row comprising “entry-name \t definition-1 <|list_more_def|>…… <|list_more_def|>definition-m”.

Ishiwatari et al. (2019) confirm data redistribution in our form, so if you would like to use our datasets, please also cite Ishiwatari et al. (2019) as well.

The 14 encoders mentioned in the paper are released under https://huggingface.co/RyuKT

How they can be used is exemplified in “**TestEncodeSentenceEmbed.py**” and “**TestSTS.py**”.

2) Training Encoders:

If you would like to train your encoders using DefSent+, two examples listed below could be helpful.

Scenario a): single step of PST (using the example of further training “SynCSE-partial-RoBERTa-base”)

Step 1): download “**config.json**” and “**pytorch_model.bin**” of SynCSE-partial-RoBERTa-base, and save them under the file named “**SynCSE-partial-RoBERTa-base**” (I assume that the file is saved under the same root directory of our code)

Step 2): create inputs and target entry ids for training your encoder using "**GenerateEncoderInput.py**" with the command below:

```
python GenerateEncoderInput.py --backbone_model_name roberta-base
                               --max_seq_length 141
                               --dataset_file ./datasets/Ox+WN.csv
                               --dataset_file_encoding utf-16
                               --save_path ./inputs
```

Noted that "available backbone model name is bert-base-uncased, bert-large-uncased, roberta-base, or roberta-large", and open a "inputs" file under the root directory.

Step 3): create entry embeddings using "**GenerateEntryEmbeddings.py**" with the command below:

```
python GenerateEntryEmbeddings.py --backbone_model_name roberta-base
                                  --model_path ./SynCSE-partial-RoBERTa-base
                                  --entry_embed_encoding ac
                                  --dataset_file ./datasets/Ox+WN.csv
                                  --dataset_file_encoding utf-16
                                  --save_path ./entry_embeddings
```

Noted that "available entry embed encoding is ac or amp", and open a "entry_embeddings" file under the root directory.

(Optional) Step 3.5): create ICA-transformed entry embeddings using "**ICA-transform.py**" with the command below:

```
python ICA-transform.py --entry_embed ./entry_embeddings/AC_entry_embed_weight.pth
                        --save_path ./entry_embeddings/
                        --max_iteration 1000
                        --seed 42
```

In the DefSent+ paper, ICA-transformed entry embeddings are effective for raw pre-trained bert-base-uncased and bert-large-uncased models at the 3rd Training of PST. In this case for further training SynCSE-partial-RoBERTa-base, this step is not needed.

Step 4): training your encoder using “**TrainEncoder.py**” with the command below:

```
python TrainEncoder.py --backbone_model_name roberta-base
                       --model_path ./SynCSE-partial-RoBERTa-base
                       --sentence_embed_encoding mean
                       --input_file_path ./inputs
                       --entry_embed ./entry_embeddings/AC_entry_embed_weight.pth
                       --save_path ./trained/
                       --batch_size 32
                       --learning_rate 1e-6
```

Noted that “available sentence embed encoding is cls or mean”, and open a “trained” file under the root directory.

Scenario b): multiple steps of PST (using the example of further training “SynCSE-partial-RoBERTa-large”)

Step 1): download “**config.json**” and “**pytorch_model.bin**” of SynCSE-partial-RoBERTa-large, and save them under the file named “**SynCSE-partial-RoBERTa-large**” (I assume that the file is saved under the same root directory of our code)

For the 1st Training of PST, from Step 2) to Step 4), there is no big difference to the single step of further training SynCSE-partial-RoBERTa-base, except:

1)) replace --backbone_model_name with “roberta-large” in Steps 2), 3) and 4)

2)) replace --model_path with “./SynCSE-partial-RoBERTa-large” in Steps 3) and 4)

3)) replace --sentence_embed_encoding with “cls” in Step 4)

3)) replace --learning_rate with “2e-5” in Step 4)

Step 5): after Step 4), cut and paste the “**config.json**” and “**pytorch_model.bin**” under ./trained to another file (e.g., 1st-Training-DefSentPlus-SynCSE-partial-RoBERTa-large)

Step 6): create entry embeddings (for 2nd Training of PST) using “**GenerateEntryEmbeddings.py**” with the command below:

```
python GenerateEntryEmbeddings.py --backbone_model_name roberta-large
                                  --model_path  ./1st-Training-DefSentPlus-SynCSE-partial-RoBERTa-large
                                  --entry_embed_encoding ac
                                  --dataset_file ./datasets/Ox+WN.csv
                                  --dataset_file_encoding utf-16
                                  --save_path ./entry_embeddings
```

Noted that there is already a “AC_entry_embed_weight.pth” used for 1st Training of PST. Thus, before execute this command, it’s better to rename it to “1st_AC_entry_embed_weight.pth”. Otherwise, it will be overwritten by the one used for 2nd Training of PST.

Step 7): training your encoder using “**TrainEncoder.py**” with the command below:

```
python TrainEncoder.py --backbone_model_name roberta-large
                       --model_path ./SynCSE-partial-RoBERTa-large
                       --sentence_embed_encoding cls
                       --input_file_path ./inputs
                       --entry_embed ./entry_embeddings/AC_entry_embed_weight.pth
                       --save_path ./trained/
                       --batch_size 32
                       --learning_rate 5e-6
```

Citation:

```
@misc{liu2024defsent,
    title={DefSent+: Improving sentence embeddings of language models by projecting definition sentences into a quasi-isotropic or isotropic vector space of unlimited dictionary entries},
    author={Xiaodong Liu},
    year={2024},
    eprint={2405.16153},
    archivePrefix={arXiv}}
```