Code for reproducing the works in GAN-LM

Please first do **pip install -r requirements.txt** before doing following steps.

BLINK **(data-augmentation-for-entity-resolution)**

We show the steps for low-resource with context case. The default setting for the size of train data after augmentation is 200 and number of train epochs in BLINK is 3 which is different from our main paper but fast to check the reproducibility. For training and evaluating the BLINK model, we used the code shared by https://github.com/facebookresearch/BLINK. If you want to see the diverse synthetic data, please use the full train set to do augmentation.

(1) Clone BLINK github:
git clone https://github.com/facebookresearch/BLINK.git

(2) Download *ZESHEL* dataset:
./BLINK/examples/zeshel/get_zeshel_data.sh

(3) Convert data to BLINK format:
python BLINK/examples/zeshel/create_BLINK_zeshel_data.py

(4) Make a train data for low-resource:
mkdir data/zeshel/low-resource
cp data/zeshel/blink_format/* data/zeshel/low-resource/
rm data/zeshel/low-resource/train.jsonl
python data-augmentation-for-entity-resolution/low-resource.py (Please use the right location for ZESHEL data. See the ccomment inside low-resource.py)

(5) Try augmentation:
- For lexical / spelling / character / gpt / opt / para / back:
./data-augmentation-for-entity-resolution/augment_conventional.sh
Change the –aug for one of lexical / spelling / character / gpt / opt / para / back.

- For BART-based augmentation:
./data-augmentation-for-entity-resolution/augment_bart.sh

- For GAN-LM:
./data-augmentation-for-entity-resolution/preprocess_gan.sh
./data-augmentation-for-entity-resolution/train_gan.sh
./data-augmentation-for-entity-resolution/augment_gan.sh

(6) Merge the original train set with synthetic data: cat train*.jsonl > train_all.jsonl → mv train_all.jsonl train.jsonl

(7) Move train_blink.sh into BLINK folder and train BLINK model:
cp data-augmentation-for-entity-resolution/train_blink.sh BLINK/→ ./train_blink.sh (inside BLINK folder)
Change APPROACH as the location of train/validation/test sets.

(8) Move pred_blink.sh into BLINK folder and measure the performance:
cp data-augmentation-for-entity-resolution/pred_blink.sh BLINK/→ ./pred_blink.sh (inside BLINK folder)
Change APPROACH as the location of train/validation/test sets.
Please change DOC_PATH in BLINK/blink/biencoder/zeshel_utils.py to the proper zeshel location

TREC **(data-augmentation-for-trec)**

We show the steps for low-resource and shared the data inside data-augmentation-for-trec/data folder as labeled.tsv. The default setting for the size of train data after augmentation is 218 which is different from our main paper but fast to check the reproducibility. For training and evaluating the BERT-Tiny model, we used the code shared by https://github.com/crux82/ganbert. If you want to see the diverse synthetic data, please use half-train set to do augmentation.

(1) Try augmentation. All the saved files will be located in data-augmentation-for-trec/aug_data:
- For lexical: python data-augmentation-for-trec/lex_aug.py

- For spelling: python data-augmentation-for-trec/spel_aug.py

- For character: python data-augmentation-for-trec/char_aug.py

- For GPT-2: python data-augmentation-for-trec/gpt_aug.py

- For BART-based: python data-augmentation-for-trec/bart_aug.py

- For OPT: python data-augmentation-for-trec/opt_aug.py

- For paraphrase: python data-augmentation-for-trec/para_aug.py

- For back-translation: python data-augmentation-for-trec/back_aug.py

- For GAN-LM: python data-augmentation-for-trec/gan_aug.py

(2) Train BERT-Tiny model and test on *TREC*. This needs the different tensorflow version and we recommend to make another environment for it:
conda create -n trec python=3.7 anaconda
conda activate trec
pip install -r data-augmentation-for-trec/ganbert/requirements.txt
pip install protobuf==3.19.0
./data-augmentation-for-trec/ganbert/run_experiment.sh

STS-B **(data-augmentation-for-sts)**

We show the steps for low-resource and shared the data inside data-augmentation-for-sts/data folder as low_data.npz. The default setting for the size of train data after augmentation is 230 which is different from our main paper but fast to check the reproducibility. For training and evaluating the SentenceTransformer, we used the code shared by https://www.sbert.net/docs/training/overview.html. If you want to see the diverse synthetic data, please use half-train set to do augmentation.

(1) Try augmentation:
- For lexical: python data-augmentation-for-sts/lex_aug.py

- For spelling: python data-augmentation-for-sts/spel_aug.py

- For character: python data-augmentation-for-sts/char_aug.py

- For GPT-2: python data-augmentation-for-sts/gpt_aug.py

- For BART-based: python data-augmentation-for-sts/bart_aug.py

- For OPT: python data-augmentation-for-sts/opt_aug.py

- For paraphrase: python data-augmentation-for-sts/para_aug.py

- For back-translation: python data-augmentation-for-sts/back_aug.py

- For GAN-LM: python data-augmentation-for-sts/gan_aug.py

(2) Train SBERT model and test on *STS-B*:
python data-augmentation-for-sts/train_test_sbert.py
Change data as the one of augmented data.

mSTS **(data-augmentation-for-msts)**

Here, we show the runbook based on jupyter notebook (Evaluation_mSTS.ipynb). The default setting for the size of train data after augmentation is 400 which is different from our main paper but fast to check the reproducibility. For training and evaluating the mean pooling of outputs for the pre-trained mBERT, we used the code shared by https://www.sbert.net/docs/training/overview.html. The summary of steps are as follows:
(A) Copy data inside data-augmentation-for-msts/data into same location as Evaluation_mSTS.ipynb. If you want to try other approaches, you should copy the original data again.
(B) Copy all py files into same location as Evaluation_mSTS.ipynb.
(C) Set the size of augmented data.
(D) Apply one of augmentation approaches.
(E) Evaluate the performance.

* For paraphrase and GAN-LM based augmentations, please read carefully the comments inside Evaluation_mSTS.ipynb to generate the augmented result properly.