Assigment 3: Image Change Captioning

This package contains the code for **image change captioning**. The image change captioning task require a model to compare two images and describe what is changed. The code is adapted from [1].

In this assignment, you'll need to

1. Implement several code snippets, including
   - InfoNCE Loss Function
   - Position Encoding in Transformer
   - Attention in Transformer
2. Run the training and evaluation scripts on your own, on a small subset of CLEVR-CHANGE dataset.

1. Environment Configuration

1. Requirement: Linux + NVIDIA GPU.
   - Not tested on Windows/MacOS.
   - Not tested on Google Colab. But you can try to upload the files and run through the notebook.
2. Make virtual environment with Python 3.8
3. Install PyTorch 1.8. Refer to Installing previous versions of PyTorch.
4. Install requirements (`pip install -r requirements.txt`)
5. Download `en_core_web_sm` english text model for spaCy, by `python3 -m spacy download en_core_web_sm`
6. Setup COCO caption eval tools (github). Or `pip install cocoevalcaps .`

2. Data Preparation

1. Download CLEVR-CHANGE data from here: google drive link, and unzip.

   ```
   tar -xzvf clevr_change.tar.gz
   ```

   Extracting this file will create `data` directory and fill it up with CLEVR-CHANGE dataset.

2. Preprocess data

   - We are providing the preprocessed data here: google drive link. You can skip the procedures explained below and just download them using the following command:

     ```
     cd data
     tar -xzvf clevr_change_features.tar.gz
     ```

   - Extract visual features using ImageNet pretrained ResNet-101:

```
# processing default images
python scripts/extract_features.py --input_image_dir ./data/images --
output_dir ./data/features --batch_size 128

# processing semantically changes images
python scripts/extract_features.py --input_image_dir ./data/sc_images --
output_dir ./data/sc_features --batch_size 128

# processing distractor images
python scripts/extract_features.py --input_image_dir ./data/nsc_images --
output_dir ./data/nsc_features --batch_size 128
```

- Build vocab and label files using caption annotations:

```
python scripts/preprocess_captions_transformer.py --input_captions_json ./
data/change_captions.json --input_neg_captions_json ./data/
no_change_captions.json --input_image_dir ./data/images --split_json ./data/
splits.json --output_vocab_json ./data/transformer_vocab.json --output_h5 ./
data/transformer_labels.h5
```

3. Code Implementation

In this assignment, you will need to implement some code snippets in the model architecture to finally train the model:

1. InfoNCE Loss. Position:
   - `models/CBR.py, Line 13`
   - `utils/utils.py, Line 292`
   - Note: the two snippets should be identical. You only need to calculate the InfoNCE loss with given unnormalized similarity matrix.
2. Position Encoding. Position:
   - `models/transformer_decoder.py, Line 31`
   - Note: you're going to implement the sinusoidal position encoding as proposed in original Transformer[2] paper. Also refer to this blog for a quick explanation.
3. Attention Mechanisms. Position:
   - Self-attention: `models/transformer_decoder.py, Line 131`
   - Cross-attention: `models/transformer_decoder.py, Line 173`
   - Attention: `models/SCORER.py, Line 48`
   - Note: these snippets are mostly similar. No residual links or LayerNorms to add, as we have already put them in the place if needed.

The places you need to modify in source files start with a comment including `== To Implement ==` . If you find it difficult to implement them, read the paper and remaining code to better comprehend how each component work together. You can also refer to public Transformer and InfoNCE codes for reference.

4. Training

To train the proposed method, run the following commands:

```
# create a directory or a symlink to save the experiments logs/snapshots etc.
mkdir experiments
# OR
ln -s $PATH_TO_DIR$ experiments

# this will start the visdom server for logging
# start the server on a tmux session since the server needs to be up during training
python -m visdom.server

# start training
python train.py --cfg configs/dynamic/transformer_quick.yaml
```

Note that we use a fractional of the whole data (~10%) for training in this assignment. If you want to try full training (takes ~2 4090 hours), replace the config file to `configs/dynamic/transformer.yaml` (no additional score!).

5. Testing/Inference

To test/run inference on the test dataset, run the following command

```
python test.py --cfg configs/dynamic/transformer.yaml  --snapshot 10000 --gpu 1
```

The command above will take the model snapshot at 10000th iteration and run inference using GPU ID 1.

6. Evaluation

- Caption evaluation

Run the following command to run evaluation:

```
# This will run evaluation on the results generated from the validation set and
print the best results
python evaluate.py --results_dir ./experiments/SCORER+CBR/eval_sents --anno ./data/
total_change_captions_reformat.json --type_file ./data/type_mapping.json
```

Once the best model is found on the validation set, you can run inference on test set for that specific model using the command exlpained in the `Testing/Inference` section and then finally evaluate on test set:

```
python evaluate.py --results_dir ./experiments/SCORER+CBR/test_output/captions --
anno ./data/total_change_captions_reformat.json --type_file ./data/type_mapping.json
```

The results are saved in `./experiments/SCORER+CBR/test_output/captions/eval_results.txt`

7. Hand-In Requirements

You are going to hand in following meterials for scoring:

1. **Python source files**.
   Only submit the four modified files: `models/CBR.py`, `models/transformer_decoder.py`, `utils/utils.py`, `models/SCORER.py`
2. **Brief report** of your own implementation, including the evaluation results on both validation and test set.

8. Reference

[1] Y. Tu, L. Li, L. Su, Z.-J. Zha, C. Yan, and Q. Huang, "Self-supervised Cross-view Representation Reconstruction for Change Captioning", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2805–2815.

[2] A. Vaswani *et al.*, "Attention is All you Need", in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, p. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf