Yuanzhan Tang (Mateo)

Homework #2

Part 1

Installation of FairSeq went very smoothly. No error occurred.

Part 2

The approach for finishing this part was to read each line of the original .tsv file as a list, and take the first element and insert spaces between characters and output the result into the grapheme file, and take the second element and directly output it into the phoneme file.

The logging output is as follows (also in data-bin/preprocess.log):

```
Namespace(align_suffix=None, alignfile=None, all_gather_list_size=16384,
bf16=False, bpe=None, checkpoint_shard_count=1, checkpoint_suffix='',
cpu=False, criterion='cross_entropy', dataset_impl='mmap', destdir='data-
bin', empty_cache_freq=0, fp16=False, fp16_init_scale=128,
fp16_no_flatten_grads=False, fp16_scale_tolerance=0.0,
fp16_scale_window=None, joined_dictionary=False, log_format=None,
log_interval=100, lr_scheduler='fixed', memory_efficient_bf16=False,
memory_efficient_fp16=False, min_loss_scale=0.0001,
model_parallel_size=1, no_progress_bar=False, nwordssrc=-1, nwordstgt=-1,
only_source=False, optimizer=None, padding_factor=8, profile=False,
quantization_config_path=None, scoring='bleu', seed=1,
source_lang='ice.g', srcdict=None, target_lang='ice.p',
task='translation', tensorboard_logdir=None, testpref='test',
tgtdict=None, threshold_loss_scale=None, thresholdsrc=2, thresholdtgt=2,
tokenizer='space', tpu=False, trainpref='train', user_dir=None,
validpref='dev', workers=1)
[ice.g] Dictionary: 40 types
[ice.g] train.ice.g: 800 sents, 5242 tokens, 0.0191% replaced by <unk>
[ice.g] Dictionary: 40 types
[ice.g] dev.ice.g: 100 sents, 634 tokens, 0.0% replaced by <unk>
[ice.g] Dictionary: 40 types
[ice.g] test.ice.g: 100 sents, 667 tokens, 0.0% replaced by <unk>
[ice.p] Dictionary: 64 types
[ice.p] train.ice.p: 800 sents, 5376 tokens, 0.0558% replaced by <unk>
[ice.p] Dictionary: 64 types
[ice.p] dev.ice.p: 100 sents, 652 tokens, 0.153% replaced by <unk>
[ice.p] Dictionary: 64 types
[ice.p] test.ice.p: 100 sents, 685 tokens, 0.292% replaced by <unk>
Wrote preprocessed data to data-bin
```

Code used for Part 2 is named preprocessing.py.

Part 3

I actually had the biggest problem finishing this part. I found the official documentation for for fairseq-train was a bit lacking, as it didn't mention all the arguments for the parameters needed for the model, such as bidirectional encoder, encoder/decoder embedding dimensionality. etc. However, I did find a lot more information on its Github page for LSTM model, and by reading the code, I got to be sure of the right arguments to use:

```
fairseq-train \
    data-bin \
    --source-lang ice.g \
    --target-lang ice.p \
    --seed 21 \
    --arch lstm \
    --encoder-bidirectional \
    --dropout 0.2 \
    --encoder-embed-dim 128 \
    --decoder-embed-dim 128 \
    --decoder-out-embed-dim 128 \
    --encoder-hidden-size 512 \
    --decoder-hidden-size 512 \
    --optimizer adam \
    --lr 0.001 \
    --criterion label_smoothed_cross_entropy \
    --label-smoothing 0.1 \
    --clip-norm 1 \
    --batch-size 50 \
    --max-update 800 \
    --no-epoch-checkpoints \
```

Then when I ran the command line, an error message kept popping up:

```
Error #15: Initializing libiomp5.dylib, but found libomp.dylib already
initialized.
```

A bit Google search revealed that it had to do with MKL, so I tried to debug it by running `conda install nomkl`, as recommended by many in online discussions, but the error persisted. I then tried to update conda, update the packages, uninstall and reinstall numpy, etc., and ultimately the problem went away – though I still don't know exactly how.

Part 4

The big challenge for me in the beginning was to process the predictions.txt file, as there was a big chunk of information irrelevant to the task at hand. I used the old trick and read each line as a list, and found that all those irrelevant information were just one string in their respective lists, and all the useful rest had a length that's greater than 1. So I used this to filter out the data that were actually needed. The data also had a pattern where each word predictions accounted for exactly 5 rows, and I used this pattern for iteration.

The final WER for the model was 18.

Code used for Part 4 is named evaluation.py.