Maria Karamihaylova
LING83600: Language Technology
Fall 2021
HW 2

Part 2 logging output:

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

Part 3 training command:

```
fairseq-train \
    data-bin \
--source-lang ice.g \
--target-lang ice.p \
--seed 11216 \
--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 \
--criterion label_smoothed_cross_entropy \
--label-smoothing 0.1 \
--optimizer adam \
--lr 0.001 \
--clip-norm 1 \
--batch-size 50 \
--max-update 800 \
--no-epoch-checkpoints
```

Part 4 WER:

WER = 24

Part 5 Reflection:

Overall, I found this assignment to be just the right amount of challenging. I did not experience any difficulty installing FairSeq. I decided to set up a data frame using pandas in order to generate files with the separated columns. It took some trial and error to figure out how to add spaces between the characters for the .g files. I initially generated empty files due to the positioning of my for loops.

I found part 3 a bit more difficult, though it was a fun puzzle to get all the parameters just right. Initially I focused only on the linked fairseq-train documentation but not all of the parameters I needed were in that document. I made several errors while building my parameters. I initially struggled with setting up a bidirectional encoder parameter but after some Googling figured out I needed a simple "encoder-bidirectional" command. The first time I ran my training command to train the model, the program ran for over two hours. I decided to terminate it and see if I could change some of the parameters to speed up the process. After a few failed attempts, an error message in the terminal revealed a list of possible parameters (not all of which had been in the documentation). I realized that I was not setting up the encoder/decoder hidden layer sizes correctly, which was resulting in a much longer run time. Once I fixed that, the model trained for approximately 25 minutes and was complete. However, the checkpoints directory was empty. I realized that I should have used the "no-epoch-checkpoints" parameter instead of "no-save." After I corrected this, I was able to generate the two models in the checkpoints directory and was ready to move on to the next step.

It took me some time to figure out how to calculate the word error rate. It was also tough to isolate the target and hypothesized words in the predictions.txt file. The formatting of the file was confusing. I stripped the file of newlines but struggled to deal with the "\t" in the printed output. First, I tried to use a regex to strip each line of the "\t" and everything that was to the left of those characters; of course, that did not work. Ultimately, my approach was to isolate the lines beginning with "T-" and "H-," split the lines by tabs, and add the final tabular element of each line to lists of the target and hypothesized words. I added up all of the word pairs that were not the same, divided that by the total number of target words, and multiplied that by 100 to obtain a word error rate of 24.