Tysean Bucknor
Language Technology HW2

PART 2

Python code used to prepare data:

```
train_data1 = pd.read_csv('ice_train.tsv', sep='\t', header=None, usecols=[0])
train_data1 = list(train_data1[0])
f1 = open('train.ice.g', 'x')
for word in train_data1:
    f1.write(' '.join(word) + '\n')
f1.close()

train_data2 = pd.read_csv('ice_train.tsv', sep='\t', header=None, usecols=[1])
train_data2 = list(train_data2[1])
f2 = open('train.ice.p', 'x')
for word in train_data2:
    f2.write(word + '\n')
f2.close()

dev_data1 = pd.read_csv('ice_dev.tsv', sep='\t', header=None, usecols=[0])
dev_data1 = list(dev_data1[0])
f3 = open('dev.ice.g', 'x')
for word in dev_data1:
    f3.write(' '.join(word) + '\n')
f3.close()
dev_data2 = pd.read_csv('ice_dev.tsv', sep='\t', header=None, usecols=[1])
dev_data2 = list(dev_data2[1])
f4 = open('dev.ice.p', 'x')
for word in dev_data2:
    f4.write(word + '\n')
f4.close()

test_data1 = pd.read_csv('ice_test.tsv', sep='\t', header=None, usecols=[0])
test_data1 = list(test_data1[0])
f5 = open('test.ice.g', 'x')
for word in test_data1:
    f5.write(' '.join(word) + '\n')
f5.close()

test_data2 = pd.read_csv('ice_test.tsv', sep='\t', header=None, usecols=[1])
test_data2 = list(test_data2[1])
f6 = open('test.ice.p', 'x')
for word in test_data2:
    f6.write(word + '\n')
f6.close()
```

Logging output:

```
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.g] Dictionary: 40 types
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.g] train.ice.g: 800 sents,
5242 tokens, 0.0191% replaced by <unk>
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.g] Dictionary: 40 types
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.g] dev.ice.g: 100 sents,
634 tokens, 0.0% replaced by <unk>
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.g] Dictionary: 40 types
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.g] test.ice.g: 100 sents,
667 tokens, 0.0% replaced by <unk>
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.p] Dictionary: 64 types
```

```
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.p] train.ice.p: 800 sents,
5376 tokens, 0.0558% replaced by <unk>
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.p] Dictionary: 64 types
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.p] dev.ice.p: 100 sents,
652 tokens, 0.153% replaced by <unk>
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.p] Dictionary: 64 types
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | [ice.p] test.ice.p: 100 sents,
685 tokens, 0.292% replaced by <unk>
2021-10-22 13:58:33 | INFO | fairseq_cli.preprocess | Wrote preprocessed data to data-
bin
```

PART 3

Fairseq-train command:

```
fairseq-train data-bin --source-lang ice.g --target-lang ice.p --seed 5 --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

Python script for computing wer

```
import argparse
import pandas as pd
import regex as re
import numpy
from jiwer import wer

def remove(list):
    pattern = r'[A-Z\-\d\\\t(\-\d\.\d\\\t)?]'
    list = [re.sub(pattern, '', n) for n in list]
    return list

def main(args: argparse.Namespace) -> None:
    with open('predictions.txt', 'r') as file:
        data = [line.strip() for line in file]
    Tlist = [string for string in data if string.startswith('T-')]
    file2 = open('target.txt', 'a+')
    for string in remove(Tlist):
        file2.write(string.rstrip() + '\n')
    gold = file2.readlines()

    Hlist = [string for string in data if string.startswith('H-')]
    file3 = open('hypothesis.txt', 'a+')
    for string in remove(Hlist):
        file3.write(string.rstrip() + '\n')
    pred = file3.readlines()

    wer_score = wer(gold, pred, standardize=True)

print(round(wer_score * 100))
```

72 ← The WER

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='calculate WER')
    parser.add_argument("predictions", help="path to input predictions file")
    parser.add_argument("target", help="path to output target file", type=str)
    parser.add_argument("hypothesis", help="path to output hypothesis file", type=str)
    main(parser.parse_args())
```

PART 5

Reflection:

In Part 1, when I first attempted to install fairseq version 0.10.2 in the command line, it returned an error unrelated to my Python version. So, instead, I just ran the command: `pip install fairseq`. However, this solution only installed an earlier version (I believe 0.10.0) which is what I suspect posed a problem in Part 3.

Part 2 was relatively straightforward. However, I would've preferred to write a more DRY script with a loop to create the .g and .p files instead of writing six blocks of code but did not want to waste time.

Part 3 began with me struggling to find a handful of flags that were buried in the `fairseq` page. After finally discovering the stray flags is when I realized having an older version of `fairseq` caused command line issues, specifically with the `--batch-size` flag. No matter which number I completed the flag with, `fairseq-train` would return the error: `fairseq-train: error: argument --batch-size: invalid typing.Optional[int] value: 'x'.` I could not figure out how to debug since I was certain I wrote the flags properly. As a last resort, I uninstalled `fairseq` then reinstalled the latest version using a more roundabout method that I don't quite remember now. After that, `fairseq-train` ran as normal.

In Part 4, the `fairseq-generate` command ran without trouble. Afterwards, part 4 became relatively tedious. I came up with many ideas about how to extract the necessary data from the predictions.txt file, but ultimately settled on defining a function comprising regex. I then wrote the extracted target and hypothesis data to two respective files. I calculated the WER by using the `jiwer` module and ended up with a WER of 77.