Sal D'Arienzo
LING83800 HW2
8 March 2021

1. Challenge 1: Writing the `.tag` files.

Using `$ wc -l train.tag dev.tag test.tag` yielded the following: `1 train.tag 1 dev.tag 1 test.tag 3 total`. I opened `dev.tag` to check out what the file looked like, and that showed a list of lists of lists, which makes sense because initially all my `write_tags` function did was write exactly what it was given to the file:

```
def write_tags(path: str, corpus: list):
    with open(path, "w") as sink:
        print(corpus, file=sink)
```

I realized the output files should look similar to the input file, which presented the challenge to maintain sentence boundaries with whitespace. I went with:

```
def write_tags(path: str, corpus: list):
    with open(path, "w") as sink:
        for sentence in corpus:
            for word in sentence:
                print(" ".join(word) if sentence.index(word) != len(sentence)-1 else " ".join(word)+"\n", file=sink)
```

Ran `wc -l train.tag dev.tag test.tag` again, and this time it looked good: `215721 train.tag 27184 dev.tag 27048 test.tag 269953 total`. But I had a bad feeling about that `print` line, so I checked `wc -l conll2000.tag`, which showed `270052 conll2000.tag` —a loss of 99 lines.

I modified `split.py` to output everything in a single file, `compare.tag`, and used `diff -y conll2000.tag compare.tag` to compare. This showed 99 instances of whitespace had been deleted from the original input file. I looked through a few of the places where the whitespace was lost and eventually realized the issue was with the `sentence.index(word)` portion of the `print` statement—in cases where the last `word` list appeared elsewhere in the `sentence` list, the above function wouldn't know it was the end of the sentence. Added a counter and it was fixed:

```
def write_tags(path: str, corpus: list):
    with open(path, "w") as sink:
        for sentence in corpus:
            counter = 0
            for word in sentence:
                print(" ".join(word) if counter != len(sentence)-1 else " ".join(word)+"\n", file=sink)
                counter += 1
```

2. Challenge 2: Implementing the `-v` optional.

This went pretty smoothly in that I got it working without any issues, but I'm not sure that I implemented it properly using the built-in logger--I wasn't sure if I should have created a custom logger instead of using the root logger. I'm also particularly uncertain about my choice to use a counter inside each `if` statement--it works, but is there a better way to count the number of times a function is called? I would have thought a logger could be used for tat (and then have the `write_tags` function only do one level of splitting, i.e. corpus into sentences or sentences into tokens, and run it twice to get the two different counts), but I didn't see anything like that in the documentation. Additionally, I couldn't get my implementation working without having `--verbose` in `parser.add_argument("-v", "--verbose", action="store_true", required=False, help="enables verbose mode",)`.