Ling83800 HW2 Write Up

Cameron Gibson

February 24, 2021

At present, *split.py* splits conll2000.tag data set into train, dev, and test sets and writes those sets to train.tag, dev.tag, and test.tag respectively. Conll2000.tag data is randomly shuffled before split, and the shuffle is seeded at 272. The data is split such that test.tag contains 80% of the data, dev.tag contains 10% of the data, and test.tag contains the final 10%. *Split.py* has been reformatted with *black* code reformatter; it passes *flake8* checks; and it passes mypy static type checking tool tests. It also prints logs of sentences and tokens for each file to command line when script is executed.

The main challenge I faced was how to split the data correctly to each individual list (train, dev, test) in the test_train_split function. We needed to split the data to the appropriate lists such that the same sentence does not end up in two different sets. We wouldn't want our model to be able to cheat after training, so we have to ensure that each sentence appears once in each set. We also need to make sure that our sentences are randomly selected from the data and added to each list.

My first solution to this problem was to calculate the size of the data set we each list would need using 'len(some_data_set) * 0.8' for training data sets and 'len(some_data_set) * 0.1' for dev and test data sets. I used a for i in range loop to iterate through the data *x* number of times and *.pop()* a selected sentences from the set, where the value to be popped was a randomly selected integer from 0 to the length of the data set. This implementation was meant to make sure that no sentences were appended to multiple lists. However, this resulted in repeated IndexErrors. I'm assuming this is because even though the length of data was shrinking correctly, the random number that was generated for popping sometimes resulted in the pop function being fed an index that had already been popped. I'm assuming this is because popping items from a list means that that index is vacant (eg. If I have items indexed at 3,4, and 5, and I pop the item at index 4,then my item at index 5 does not fill the vacancy at index 4). I tried to keep tracked of used index values, and having the the function pass if the value had been used, which I think could have worked, but I was getting tired, and I don't think I implemented it correctly, so I still returned IndexErrors on almost every run. Fortunately, by reading further into the random library documentation, I found out that random.shuffle() is a thing. So, I instead decided to just shuffle the data using random.shuffle. I then seeded that instance using random.Random(seed).shuffle(data) to make sure that the

shuffle function returned the same shuffled set each time. From there, I used list indexing to set the value of each list, which solved the problem. I'd also like to get better with using logging to track progress, errors, runtime, and all of those good things in future assignments.