Homework 2 Writeup

Reuben Raff

3/8/2021

Introduction

In this lab I explored how to write a short program splitting a corpus into training data, development data and test data. I leveraged the *argparse* python module in order to create command line flags for the file paths for the training, development and test data. I read in a corpus of data from the conference on computational natural language learning from the 2000 workshop. The data is parsed in three columns and read in using a helper function. I used the random library to shuffle the data before splitting it. I additionally added a seed argument to stabilize the pseudo random number generator which underlies the mechanics of the random.seed argument. I additionally utilized the logging library to log output.

Methodology

I began my program solving the primary issue of getting a solid fundamental understanding of the argument parser tool. I've worked with command line arguments and bash scripts in industry settings before, but I had never written them. This was a fun and novel challenge. I relied heavily on the examples we reviewed in practicum and with moderate testing I was able to get all the flags to behave as intended. Splitting the data appropriately was another challenge that took some time to think out accurately. I found a search on stack overflow particularly useful. I was able to cast an 80% partition of the list corpus that the *read_tags()* routine returned. I cast the product of the length of the list and the 0.8 partition for training and used this as an index on which to split the training set from the entire data set. I then created a second partition with 90% of the data. This was stored in a variable called *split_2*. Training data was stored in the partition up to the 80% index, development data was stored in the 10% between the 80% index and the 90% partition. Test data was the remaining 10% partition.

I split the data using an 80% partition of the length of the list returned by the read_tags() method and then I cast that as an int. I then indexed the corpus at this value which I called split_1. I created a variable, *split_2* which took a 90% partition of the data cast as an int as well. I split the list stored in the corpus variable to store the training, validation and test data. I used a write_tags() argument to generate files, loop through each list for the training, dev and test datasets and wrote each partition to its own *.tag* file. I added a seed argument after initializing the random seed. I shuffled the data before splitting using the *random.shuffle* method as well. "Seed" is also a required argument in my set of required argument parser arguments. The train.tag file has a word count of 207069, about 80% of the data, and the dev and test data each have approximately 10% of the data.

I was also able to achieve the stretch goals. I struggled the most with logging. After some consultation online with some particularly helpful YouTube videos and Lucas' gist I found a good tutorial on logging messages to a log file. I was also able to run my code through black and reformat according to the -l79 character limit. I additionally tried to add type signatures to my functions. I got an error in my type signature for the *write_tags()* method but it didn't prevent the code from executing. Additionally, I worked

through bugs for my write_tags() function to
create stable file output with help from
Cameron Gibson.