HW2 write up

Reuben Raff

10/24/2021

I found this homework assignment extremely reminiscent of the work I did on the Google team I worked on as a TVC for 2.5 years before I came to The Graduate Center. I ran bash scripts with flags very similar to the fairseq suite of options with a training script, interactive mode script and scoring script to calculate precision and recall. I found the preprocessing was easy, although my initial code was sloppy. Nonetheless, I formatted the data relatively easy. I found training to be achievable, if not a bit tedious. The git repo flags, and the flags listed on the docs linked in the assignment README were not entirely aligned. This wasn't a huge deal. I just searched StackOverFlow for a few hours and after digging I found basically everything. It's cool that this workflow is open source. I feel like it's probably suboptimal in some ways I haven't realized yet. I noticed that the tasks are all pretty much translation related. I couldn't see a way to change the task flag's TASK value. Is it possible to do classification with this tool? In the flavor of sentiment analysis or assign labels to text? Empirical question probably. The generate script ran easily enough. Thanks to our meeting in office hours, I was able to check the strings of the H- and T- rows with the csv library. I obtained a WER score of 17. For fun, I built a biLSTM in Keras to watch the nitty gritty training in action. But I trained it on imdb because it's built in and easy.

My WER score was calculated by iterating over the phones in columns. I extracted the H- and T- rows from the original predictions file. Then I took each string and sorted() the string. I learned this in a LeetCode exercise I did once prepping for an interview. I noticed there was a left trailing tab for all the phones in the first column, so I removed it. Then I checked if each character in the phone1 list was equal to the phone2 list. I incremented a count variable by 1 if this condition was satisfied. I subtracted my count value from the length of the phone1 column. It has 100 values and 83 phones matched. This gave me the remaining error of 17.

The transformer was a bit more finicky. I got the model to run quickly and tuned the hyperparameters in accordance with the paper linked in the stretch goals.

All of the target 'phones' have at least one <UNK> in every target phoneme so the overall WER score is 100. I didn't have time, but I could do an error rate for each *phone* in the future. This is expected because we discussed that LSTM architectures perform better with local dependencies and small datasets such as the dataset we have in this assignment compared to transformers, which perform better with much larger datasets.

Moving forward, I would like to reattempt this assignment but build out the model using pytorch. The training routine is tricky as we discussed but I would like to get a better handle on the technical model building and model tuning aspects of LSTMs and neural nets in general. Would it be more fruitful to use a tool like fairseq's command line interface and apply it to a novel data set? In a discussion I had I got a recommendation to begin to use various techniques and architectures to address my research interests. So, maybe it'd be better to test fairseq transformers and LSTMs in a contrastive analysis on data I analyze with Rhetorical Structure Theory or Centering

theory. I appreciate your thoughts and feedback.