Module 3

Programming Assignment

Programming assignments are due by 11:59:59 PM on the day of the next lecture. All programming assignments are graded for clarity and functionality. Your code should be well-organized and include helpful comments where appropriate. Please upload your completed assignment to canvas as a zip file.

You have two parts to the assignment this week: (1) use the outputs of the tokenizer training algorithm to construct a Tokenizer class, and (2) implement an Embedding layer. Most of your time should be spent on (1).

(1)

Last week, our assignment was to implement BPE training, which produced as output two files: vocab.txt and merges.json. This week, you will create a class that can read in these two files and create a tokenizer object. This object can then be used to convert strings to token ids, and vice versa. Please see *tokenizer.py*.

(2)

In our lecture we discussed embeddings. You will need to implement an embedding layer with learnable embedding parameters. Please see *embedding.py*.

To assist in this assignment are provided outputs from the BPE training algorithm. You can use these (or your own outputs from last week) to validate your implementation of Tokenizer:

vocab.txt : a list of the vocabulary of the tokenizer. Each line is a separate token.

merges.json : A nested list in which each entry is a pair of tokens: ["token1", "token2"]. This represents the order in which merges should be applied (the order that they were learned by BPE).

Your deliverables for this assignment are:

- A completed *tokenizer.py* file which includes your implementation of a Tokenizer class.
- A completed *embedding.py* file which includes your implementation of an Embedding module.