Read Me

This document consists of the brief explanation of parameters, train and then code generation(testing) phases of the code

Parameters

- model: T5 model instance for training.
- tokenizer: Tokenizer associated with the T5 model.
- train_dataset: Training dataset (instance of HtmlDataset).
- epochs=3: Number of training epochs (default is 3).
- learning_rate=5e-5: Learning rate for the optimizer (default is 5e-5).

Training

Training is done using custom train function that performs the training loop, including loading batches of data, tokenizing inputs and labels, computing gradients, and updating the model parameters. The function prints the average loss for each epoch and saves the fine-tuned model at the end of training

It is utilizing the following classes and functions

1. Custom Components:

- HTMLAwareTokenizer: Custom HTML-aware tokenizer.
- HtmlDataset: Custom dataset class for handling input and label data.
- preprocess_html: Custom function for preprocessing HTML text.
- tokenize_html_aware: Custom function for HTML-aware tokenization.

2. Predefined Components:

- T5ForConditionalGeneration and T5Tokenizer: Predefined classes from the Hugging Face transformers library for T5 model and tokenizer.
- torch.device: PyTorch class for specifying the device.
- torch.optim.Adam: PyTorch class for the Adam optimizer.
- torch.utils.data.DataLoader: PyTorch class for efficient data loading.

3. Training and Optimization:

- Utilizes PyTorch's training loop for gradient computation, backpropagation, and model parameter updates.

Testing

For testing a custom function generate_correction is used which generates a corrected version of the input text using the T5 model and tokenizer parameters.