Peyton Golden

CSCI 420

11 April 2025

The objective of this project was to fine-tune the Salesforce/codet5-small transformer model with the Hugging Face's Transformers library to predict if condition statements in Python methods. The objective was to mask the if statements in Python methods and train the model to reconstruct masked elements.

Dataset preparation was a crucial starting point. A cleaned Python dataset was used, where every record contained a cleaned\_method column consisting of the entire method without comments or boilerplate methods and a target\_block column consisting of the precise line(if condition) that would be masked when training. The target\_block was found for every method and replaced with a <mask> token to create the masked\_method. This masked strategy was employed as input to the model, and the target\_block originally was employed as target output. The resultant dataset was split into training, validation, and test sets and saved in CSV files.

Fine-tuning was accomplished with Hugging Face's Trainer API. The model, Salesforce/codet5-small, was trained up to a maximum of seven epochs with early stopping, learning rate of 5e-5, weight decay of 0.01, and training batch size of 16 and evaluation batch size of 2. Model saving and evaluation were performed after each epoch, and the best model was selected on the basis of minimum validation loss. In addition, the tokenizer was also extended to support a custom <mask> token, and the model's embedding layer was resized accordingly to accommodate this new token.

In the evaluation, several metrics were used to measure the quality of the generated if conditions. These included BLEU-4, CodeBLEU, exact match accuracy, and comparisons between predicted and expected conditions. While BLEU-4 provides an n-gram similarity score, CodeBLEU can understand code syntax and data flow. CodeBLEU scores were consistently higher than BLEU-4 scores, showing that the model did a good job understanding how python code is written; however, it was not as accurate in predicting the actual code. Exact match rates were very low, and many predictions were missing or partially completed, resulting in a large part of the dataset being skipped during evaluation. This was likely due to errors in the code that I wrote.

In Conclusion, there were plenty of errors in my usage and fine-tuning of the codet5 model. However, with the data I have collected, it appears that the model does gain a good understanding of the Python code and how it is employed; however, there is an error preventing most of the predictions from occurring.