GenAI for Software Development: **Assignment 2**

AJ Arnold Max White

ajarnold@wm.edu mjwhite05@wm.edu

1 Introduction

In our implementation of this assignment we use a fine-tuned CodeT5 Transformer model (from Hugging Face) in order to automatically recommend potential if statements contained within Python functions. Specifically, we used a small version of CodeT5, codet5-small, and trained it on a set of data (ft_train.csv) provided by our professor.

To accomplish this, we clean and mask the data within our training, validation, and test data sets. This requires flattening the Python functions and exchanging the targeted if-statements within the function with a mask. Once this is complete, we load the pre-trained model and tokenizer, prepare a dataset that will be used in the training, define the training arguments and trainer, and train the model. Finally, we create a new results file that will house the model's predictions and our evaluation scores.

To evaluate the efficiency of the predictions, we use exact match, a CodeBLEU score, and a BLEU-4 score.

The repository can be found here: ajarnold920/Fine-Tuning_CodeT5

2 Implementation

2.1 Readying Data

Gathering Data: Our training, validation, and test sets were all provided by Professor Mastropaolo. They are found as ft_train.csv, ft_valid.csv, and ft_test.csv respectively.

Cleaning: In order to tokenize the data, it must be cleaned. To do this, we flatten the Python functions by removing all whitespace characters (excluding the regular space).

Code Masking: A mask must be given to the model so that it understands where predictions are supposed to be made. To do this, we iteratively replaced each targeted if statement within our datasets. The mask we decided to use is "<IF-STMT>".

Code Tokenization: After the datasets are formatted correctly, they are placed into a dataset dictionary and tokenized. This dataset dictionary is what is later used as input for the Python CodeT5 model object. We utilize the RobertaTokenizer Python package to tokenize the Python functions.

2.2 Model Training

Training Arguments: Importantly, our model saves results and evaluates performance after every epoch through the training set. The number of training epochs we use is 7, meaning the

training dataset is completely passed through 7 times, in batches of 16 samples to improve speed. However, due to the early stopping after two epochs without improvement, our model stopped training after 6 epochs.

2.3 Model Evaluation

Formatting Our Results: Once the model has been trained and made its predictions (based on the masked if-statements), we create a new dataset that will hold our desired output. The column contains six columns: Masked Method, Exact Match, Expected if Condition, Predicted if Condition, CodeBLEU Score, and Bleu4 Score. A breakdown of each column is as follows:

- Masked Method: the cleaned Python function that contains the mask over a targeted if-statement.
- Exact Match: a boolean that is true if the model's prediction exactly matches the Python function pre-masking.
- Expected if Condition: the targeted if-statement.
- Predicted if Condition: the if-statement predicted by the model.
- CodeBLEU Score: the CodeBLEU score that the model's prediction earns based on its quality (as compared to the desired prediction).
- Bleu4 Score: the Bleu4 score that the model's prediction earns based on its quality(as compared to the desired prediction).

This dataset can be found in the GitHub repository as testset-results.csv.

Model Evaluation:

We evaluate our model based on an exact match (previously defined), a CodeBLEU score, and a Bleu4 score. They are defined as follows:

- CodeBLEU Score: an evaluation metric that considers both grammatical and logical correctness. Uses a weighted n-gram match and syntactic AST match to measure grammatical correctness, and a semantic data-flow match to calculate logical correctness.
- Bleu4 Score: an evaluation metric that considers **only** grammatical correctness. Since the score only focuses on string similarity, we pair it with the CodeBLEU score to also cover logical correctness.

Testing Results: After we calculated these values on the entire test set we could find the average values for the various metrics. Of the 5000 test set if statements, 1526 were exact matches, leading to an exact match rate of 30.52%. The average CodeBLEU score was 0.895352 and the average BLEU-4 score was 0.883574, indicating a high level of similarity in the code fragments. It is important to note that those BLEU scores were calculated on the entire python method with the associated if statement, since BLEU metrics struggle on short code fragments. These results show the capabilities of fine-tuning to put the general code knowledge of a pre-trained model into use for real tasks.