Project Progress Report: Method Generation Group

Changshu Liu (cl4062), Zhangyi Pan (zp2223), Huiyan Zhang (hz2757)

1. Overview

Code completion has become one of the most important functions in Integrated Development Environments (IDEs), which can speed up the developing process by providing auto-completion popups, querying parameters of functions, and providing relative hints. Lots of studies have been conducted on it in order to provide programmers with better tools to code conveniently. Recently, researchers have proposed a new direction to help programmers or even non-technical people write code quickly. The new direction is called method generation. More specifically, given a docstring and signature (method name), researchers believe that there are models that can predict the method body successfully. One of the most popular related studies is CodeXGLUE, which inspires our experiment. In CodeXGLUE, it simply followed the pipeline of code generation and concatenated signature and docstring together as a new input. And with the help of CodeGPT, it was able to predict the method body. We argue that a concatenation may not fully exploit the connection between signature and docstring. In this study, we are going to set up a new multi-task learning framework, containing a common encoder and two different decoders. The first decoder will generate the function name and the second decoder will generate the method body. We believe this innovation will improve the baseline model.

2. Value to User Community

The development of method generation is able to provide lots of benefits. For software engineers, more accurate method generation tools will release them from heavy coding implementation tasks. Instead of spending time on writing easy or repetitive functions, they can focus more on high-level design and tricky function implementations. When it comes to people who don't know much about coding, method generation becomes even more useful. After coming up with a design of a method, they can get the code by simply typing in their description and method name. This will be extremely helpful for product managers, UI designers, and other non-technical people who work closely with codes. However, even though method generation can bring tons of advantages, the current version of it is not accurate enough for a wide range of use in the real world. We wish what we are doing can propose a possible solution to improve the current model. And hopefully, the model can become accurate enough to be used in the industry one day.

3. Research Questions

Research Question 1: How does our model perform in comparison with other baselines? To answer this question, we need to compare our model with the CodeGPT and CodeBert model.

Research Question 2: How much does the auxiliary task contribute to the overall performance of our proposed model? To answer this question, we need to conduct ablation studies by building variants of the proposed models.

Research Question 3: Will pretrained weights affect the performance of our model? To answer this question, we need to try both cases: training from scratch and training with pretrained weights.

4. Project Progress

Zhangyi Pan:
I am in charge of implementing the function to retrieve information from the signature and conduct experiments with CodeXGLUEon the small datasets we created.

For the information retrieving, we want to extract the function name from the signature and convert the name into an array of individual words. The function name in our dataset follows the format of "word1_word2_word3…", thus we extract each word by splitting with "_". For example, a function name "do_transform" will be translated into ["do", "transform"]. This could be helpful for our separate decoder for function names.

Another job I did is to conduct experiments on our small dataset with CodeXGLUE. Since the original dataset was too large and will take extremely long to train, I generated the baseline results with our mini-dataset. With a smaller dataset, we achieve a result of 27.98 edit similarity and 1.97 BLEU. We will use the statistics as the baseline and see if our method makes improvement.

Huiyan Zhang:
In this phase, I have been working on the datasets. The original dataset used in the study is the CodeSearchNet Python dataset. They have cleaned the dataset by removing the methods that don't have docstrings and whose name contains 'test'. I pulled down the dataset and realized we had around 900,000 instances of training data which was extremely big. Given the time limitation, I narrowed down the input size to its 1/20 and 1/40. Therefore, we got a mini dataset with around 22,500 training instances, and a small dataset with around 45,000 training instances. All instances were selected randomly. And the test dataset and the dev dataset were also narrowed down accordingly. I have also

been spending some time working with the original code to test and understand our baseline model.

Chansghu Liu

I have built a transformer encoder-decoder model which takes the concatenation of the signature name and NL guidance as the input. Like the CodeGPT model, it is also a part of our baselines.

I have spent some time on our multi-task learning model, but I haven't finished yet.

5. Code delivery

We plan to deliver our code in the following public github repository: https://github.com/Huiyan0409/6156FinalProject.

The published methodology we are using is in the following github repository: https://github.com/microsoft/CodeXGLUE/tree/main/Code-Code/Method-Generation.

6. Demo

We plan to use a slide to show our model design and experiment results (including ablation studies). Apart from these we are considering how to show our model online. .For example, we can load our model using a Colab notebook so that we can feed it with some inputs and see the model's prediction. We will try in this way.