CSE 493G1 Project Proposal - Code Classification Architectures

Rich Chen
University of Washington
rc2002@cs.washington.edu

Andy Stanciu
University of Washington
andys22@cs.washington.edu

1. Introduction

Our deep learning project aims to classify code solutions to programming problems from LeetCode using various neural network architectures.

1.1. Background and Contextual Work

The bulk of the work in interactions between deep learning models and computer programming code has been in the area of code generation, for the most part. There are three main paradigms: description-to-code, code-to-description, and code-to-code [3]. Popular applications within these include code generation from natural language, documentation generation, and automated program repair. Common ML models used here include recurrent neural networks, transformers, and convolutional neural networks, among others. [3]

Other than this, there has been work done in the classification of code snippets, primarily to identify the programming language used [2]. Work in this area has primarily relied on natural language processing techniques.

In both cases, datasets are extracted primarily by sources such as programming reference websites like StackOverflow, or publicly available repositories like GitHub.

Finally, there has also been research on using graph neural networks to process code, especially for software analysis tools to predict naming of variables as well as misuse of variable names as a programming aid [1].

1.2. Problem Statement

From this, we present our novel problem, which is to classify code snippets by the problem they intend to solve, rather than the programming language used. In particular, we would like to train a model to, given a snippet of code (which may be correct or incorrect), successfully assign it to a problem that the code is intending to solve, in some sense outputting the purpose of the code.

1.3. Unique Insight

Our unique approach is to use graph neural networks as a classifier on the code. In particular, we believe that, because code can be represented as an abstract syntax tree, and thus a graph, we want to explore if a graph-based representation in a neural network can potentially be effective.

1.4. Technical Challenges

We anticipate implementing various neural network architectures will be the bulk of the challenge. Furthermore, we believing finding means to scrape the data and then clean the data could be time consuming (pre-processing could be a long process of filtering and then cleaning up the data). This process will likely include redacting any overly-informative identifiers (e.g. the name of a method could be the problem it is trying to solve) to ensure that our model is not excessively reliant on variable names.

1.5. Experimental Process

We intend on aggregating a large quantity of examples from various LeetCode problems, including both correct and incorrect solutions written in some compiled language like Java.

Then, we will use three approaches to build classification models for the data:

First, a recurrent neural network to process the code as if it were sequence data.

Second, a convolutional neural network, processing the code as an image (screenshot of the code).

Finally, our graph neural network approach, to process the code as its underlying abstract syntax tree.

1.6. Expected Outcome

We expect our graph approach to perform best, because it more accurately captures the underlying structure of the code, which is better represented as its syntactic structure preserving relationships between parts of the code, rather than as a sequence of tokens or as an image, especially as different solutions may be wildly different in sequential nature or look wildly different as an image.

References

[1] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs, 2018. 1

[2] Kamel Alreshedy, Dhanush Dharmaretnam, Daniel M. German, Venkatesh Srinivasan, and T. Aaron Gulliver. Scc: Automatic classification of code snippets, 2018. 1

[3] Enrique Dehaerne, Bappaditya Dey, Sandip Halder, Stefan De Gendt, and Wannes Meert. Code generation using machine learning: A systematic review. *IEEE Access*, 10:82434–82455, 2022. 1