研究計画書
Research plan

氏名/Name YUE, Ziran　日付 /Date 2024 年 09 月 11 日

研究課題 Theme of your research	抽象構文木を用いたコード補完のための Graph-Based 方法 Graph-Based Approach for Code Completion Using Abstract Syntax Tree

IPS で行う研究について、(1)背景、(2)目的と目標とその理由、(3)様式 4 記載内容との関連、(4)方法、(5)期待される効果(新規性あるいは有効性など)を論理的、具体的に記述してください。なお(3)では、研究計画が様式 4 記載内容と関連がある場合はその違いを、ない場合は経験不足を補う方法を記述してください。また、資料を参照する場合は、その出典を明記してください。剽窃は認められません。(本用紙枠内 1000 字以内厳守。但し、出典の記述は文字数としてカウントしない。別紙記入不可。印字可)

Describe your research plan in IPS logically and concretely (1) background, (2) purpose, goals, and their justification, (3) relationship with the description in Form 4, (4) methods, and (5) expected results (e.g. novelty or importance). In (3), if your research plan is related to the information in Form 4, explain the difference, if not, explain how to catch up the lack of experience. If you are referring to journal papers, state the source. In any case, plagiarism is not permitted. (Please limit your essay to 500 words on this form. References are not included in the count. Do not attach any separate sheet. Printing is acceptable.)

1. Background

Code completion is an essential feature of Integrated Development Environments (IDEs), helping developers by suggesting how to finish code. Traditional methods rely on static analysis using Abstract Syntax Trees (ASTs, as shown in Figure 1) to predict code that follows syntax rules. In contrast, modern deep learning techniques feed sequential input into Large Language Models (LLMs) [1] to produce semantically meaningful lines of code, which may not always strictly match syntax rules [2]. To address this issue, inspired by their complementary relationship, we explore combining these methods by feeding ASTs as non-linear structured data into generative Graph Neural Networks (GNNs) for code completion.

2. Goals and Their Justification

This research aims to develop a graph representation of program code using AST and propose a graph-based neural network model for predicting subsequent code as new nodes in the graph for code completion. This study aims to bridge the gap between generative GNNs and code completion through AST representation.

3. Relationship with Bachelor's Thesis

Both research efforts focus on Python code completion as a subtask of Natural Language Processing (NLP). While my bachelor's thesis used a sequential character-level LSTM to enhance the static completion engine in IDEs, this research will explore a non-sequential input format (AST) and feed the input into a generative GNN, which is more suitable for extracting structural features from data structures like AST. Moreover, the evaluation methods proposed in my bachelor's thesis can still be applied.

4. Methods

Dataset: Gather target code fragments from the internet. Remove redundant parts to ensure a clean input.

Tokenization: Separate parts of the target code fragments to form legal ASTs. This can be challenging, as there is currently no algorithm for truncating a grammatically correct subtree from an AST.

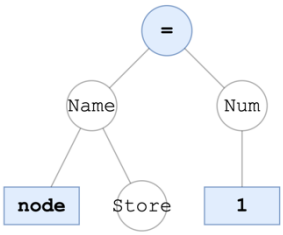

Figure 1. Abstract syntax tree of "node=1"

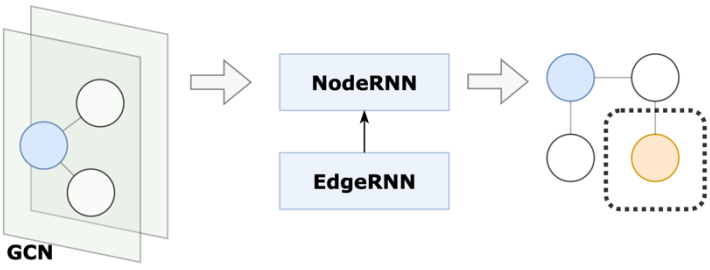

Figure 2: Graph-based model for code completion

Model: The network uses multiple Graph Convolutional Network (GCN) layers that take partial AST as input. The GCN output is passed to a generative module composed of two Graph Recurrent Neural Networks (GraphRNNs) [3]: one decides whether to add a new node, while the other predicts its connection to existing nodes. The final output is generated by identifying the graph's head (the AST root), and re-parsing it into human-readable code. Code generated is shown as the node rounded by dashed line in Figure 2.

5. Expected Results

We expect this research to produce a novel graph-based neural network model that enhances code completion by effectively utilizing ASTs to build the connection between syntactic structure and semantic understanding. These findings could advance methodologies in programming tools and inspire further research in graph-based representations for coding and other NLP tasks.

6. References

[1] S. Kim, J. Zhao, Y. Tian, and S. Chandra, 'Code Prediction by Feeding Trees to Transformers', in 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2021, pp. 150-162.

[2] D. Shen, X. Chen, C. Wang, K. Sen, and D. Song, 'Benchmarking Language Models for Code Syntax Understanding', in Findings of the Association for Computational Linguistics: EMNLP 2022, 2022, pp. 3071-3093.

[3] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, 'GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models', in ICML, 2018, vol. 80, pp. 5694-5703.