代码智能技术互补性研究

一、 背景介绍

代码智能技术 (Code Intelligence Techniques) 是指利用人工智能帮助软件开发人员提高开发过程生产力的技术。随着软件产品类别越来越丰富、软件产品规模越来越庞大，此类技术在软件开发过程中扮演的角色越来越重要。

然而，现实世界中开发人员的需求多种多样，致使单一的代码智能技术很难在所有场景下实现较高有效性，因此，本课题尝试探索将多种代码智能技术组合起来使用，以克服单一代码智能技术陷入的瓶颈。

具体而言，现有代码智能技术大体上可以分为三类，一是基于启发式的方法，如 Biker[1]，此类方法主要利用一定的启发式规则，从词法/语法层面做判断；二是基于有监督学习的方法，如 DeepAPI[2]，此类方法将代码进行嵌入表征(embedding)，试图将代码语义信息映射到高维向量空间，从而开展后续的任务；三是基于预训练的方法，如 CodeBERT[3]，此类方法也进行嵌入表征,但在训练过程中试图通过大规模无监督的预训练使模型掌握更丰富的程序语言特征与领域知识。本课题出发点在于，既然各代码智能技术之间的工作原理不同，可能存在如下情况：某一类技术不起作用时另一类技术的效果很好，也即，不同类型的技术之间存在**互补性**。因此，本课题希望探索各类技术之间的互补性，将各类技术进行融合，从而达到效果的提升。

二、探索的任务及技术

本课题将探索两个常见的智能化软件开发中的任务：

1. API 推荐，即给定一条描述程序员需要实现的功能的语句，返回程序员们实现该功能可能需要用到的 API
   - 启发式的方法：Biker
   - 有监督学习的方法：DeepAPI
   - 预训练的方法：CodeBERT
2. 代码补全，即根据当前程序中的上下文信息，预测开发人员接下来将要输入的代码
   - 启发式的方法：N-gram [4]
   - 有监督学习的方法：LSTM [5]
   - 预训练的方法：T5 [6]

三、融合方法

我们将用到的融合方法分为基础版与进阶版两种：

- 基础版：利用信息检索领域经典方法—数据融合[7]，对各类技术的结果进行融合；
- 进阶版：基础版的融合方法是已有的经典方法，不够有创意，不足以支撑将本课题成果发表于 CCF-A 类会议或期刊上，在进阶版中，我们试图针对某一任务提出特定的融合手段，其技术路线需要我们在完成基础版的过程中进行探索

四、时间安排

本课题大约持续两学期,第一学期 (即 2023 年春季学期) 完成基础版效果的评估,第二学期 (即 2023 年秋季学期) 完成进阶版方法设计、实现与评估

五、可学到的知识与合适人选

通过开展本课题的研究，同学们将会得到以下益处：

- 熟悉智能化软件开发领域中的热点话题及其相关研究；
- 熟练掌握深度学习框架 Pytorch 的使用；
- 熟悉并了解大规模代码预训练模型的前沿研究进展

本课题将为硕士/博士期间有兴趣开展智能化软件工程方面研究的同学们打下良好的专业背景，也因此适合这些同学们开展

六、联系方式

指导老师：刘烨庞 (liuyp1@sustech.edu.cn)
指导学长：王尚文 (wangshangwen13@nudt.edu.cn)

七、参考文献

[1] Qiao Huang, Xin Xia, Zhenchang Xing, David Lo, and Xinyu Wang. 2018. API method recommendation without worrying about the task-API knowledge gap. In 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 293–304.
[2] Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016. Deep API learning. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE). 631–642.
[3] Feng Z, Guo D, Tang D, et al. CodeBERT: A Pre-Trained Model for Programming and Natural Languages[C]//Findings of the Association for Computational Linguistics: EMNLP 2020. 2020: 1536-1547.
[4] Hellendoorn V J, Devanbu P. Are deep neural networks the best choice for modeling source code?[C]//Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (FSE). 2017: 763-773.
[5] Li J, Wang Y, Lyu M R, et al. Code completion with neural attention and pointer networks[C]//Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI). 2018: 4159-25.
[6] Ciniselli M, Cooper N, Pascarella L, et al. An Empirical Study on the Usage of Transformer Models for Code Completion[J]. IEEE Transactions on Software Engineering, 2021.
[7] Vogt C C, Cottrell G W. Fusion via a linear combination of scores[J]. Information retrieval, 1999, 1(3): 151-173.