COSE474-2024F: Final Project Proposal
Prompt Tuning for Rust Code Generation Using OpenPrompt

Korinek Lukas

1. Introduction

In recent years, the demand for automated code generation tools has grown, driven by fast advancements in natural language processing, which gained significant popularity following the introduction of ChatGPT. As a student learning Rust, I've found that Rust code generation tools not only help me learn the language but also improve my coding practices. The performance of these tools can vary depending on the effectiveness of the prompts used. This project explores how prompt tuning, a technique to optimize prompts for better performance, can enhance the accuracy and efficiency of Rust code generation using OpenPrompt, an open-source framework for prompt-based learning.

2. Problem definition & challenges

The objective of this project is to compare the performance of Rust code generation using standard prompts versus prompts enhanced by prompt tuning. I aim to work with Llama (Touvron et al., 2023) developed by Meta AI or the CodeGen (Nijkamp et al., 2023), an open-source model developed by Salesforce, which is available in four different sizes: 350M, 2B, 6B, and 16B parameters.

Prompt tuning is expected to improve the model's ability to generate accurate Rust code. OpenPrompt, a framework designed to assists in prompt tuning, will be used to fine-tune the prompts on a Rust specific dataset. The challenges include understanding and applying prompt tuning techniques, setting up the OpenPrompt framework, and effectively comparing results.

3. Datasets

For testing, I will use the HumanEvalPack (Muennighoff et al., 2024), an extension of the HumanEval (Chen et al., 2021) dataset developed by OpenAI for their Codex (Mark Chen, 2021) model. HumanEval contains 163 Python code samples, each paired with prompts, canonical solutions and test cases to evaluate the correctness of the generated code. The HumanEvalPack extends this dataset to include five additional languages, including Rust.

4. Related Works

A related study presents a Rust code-generating model known as RustGen (Wu et al., 2023). This model uses several techniques, including prompt-tuning, to improve the quality of the results. It presents the specific prompt template used which can serve as an inspiration for the design of prompt-tuning strategies for this project.

Furthermore, OctoCoder (Muennighoff et al., 2024), also mentioned in the same paper as the HumanEvalPack (dataset planned for testing), is another Rust code-generating model. When evaluated on the HumanEvalPack dataset, OctoCoder performed the worst (among the six languages) on Rust, achieving an accuracy of just 23.4%. This performance further highlights opportunities for improvement in this language.

5. State-of-the-art Models

Current state-of-the-art models for code generation include Codex (Mark Chen, 2021), developed by OpenAI, which powers GitHub Copilot. While Codex is highly effective for generating code, it is not open-source; thus, experiments can primarily be conducted through GitHub Copilot itself. Another well-known model is Llama (Touvron et al., 2023), developed by Meta AI, which is open-source and suitable for research.

References

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., and et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Mark Chen, Jerry Tworek, H. J. Q. Y. H. P. d. O. P. J. K. e. a. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Muennighoff, N., Liu, Q., Zebaze, A., Zheng, Q., Hui, B., Zhuo, T. Y., Singh, S., Tang, X., von Werra, L., and Longpre, S. Octopack: Instruction tuning code large language models, 2024. URL https://arxiv.org/abs/2308.07124.

Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., and Xiong, C. Codegen: An open large language model for code with multi-turn program synthesis. *ICLR*, 2023.

Touvron, H., Lavril, T., Izacard, G., and Rodriguez, A. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Wu, X., Cheriere, N., Zhang, C., and Narayanan, D. Rustgen: An augmentation approach for generating compilable rust code with large language models. In *ICML 2023 Workshop on Deployable Generative AI*, 2023. URL https://openreview.net/forum?id=y9A0vJ5vuM.