LLM Code Generation Case Study

Willis H. Reid

UNC Charlotte

ITCS 5156 Applied Machine Learning

Minwoo Jake Lee

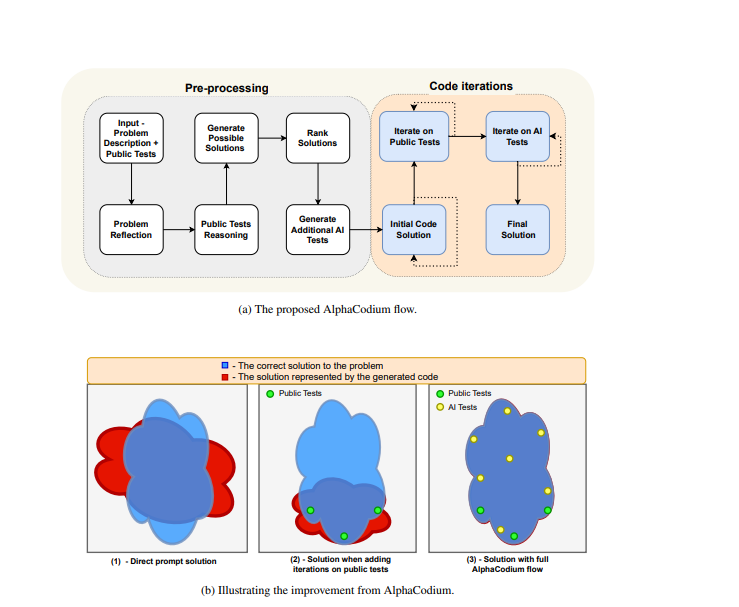
April 14, 2024

# Abstract

This research work aims to address the unique challenge posed by code generation tasks which differ significantly from typical nature, language, and problems. While code generation requires precise send text, matching consideration of various Edge cases, and adherence to specific code requirements. Many traditional optimization techniques used in natural language generation may not be efficient for coding tasks. To tackle this research, we propose a novel approach called Alpha Codium, leveraging large language model LLMs. Alpha Codium is a test base, multi-stage code code-oriented iterative flow designed to enhance machine learning performance on code problems.

# Introduction

The introduction outlines the challenges of code generation tasks, emphasizing the difficulty of assessing the usefulness of partial or incorrect solutions due to the complexity of programming problems. It introduces the Code Contests dataset and discusses previous approaches like Alpha Code and Code Chain. The paper presents Alpha Codium, a novel iterative approach to code generation, highlighting key design principles and practices such as generating additional tests and utilizing code-oriented techniques. Alpha Codium significantly improves language model performance on Code Contests problems compared to previous methods while requiring fewer computational resources, demonstrating its effectiveness and potential applicability to general code generation tasks.



# Motivation

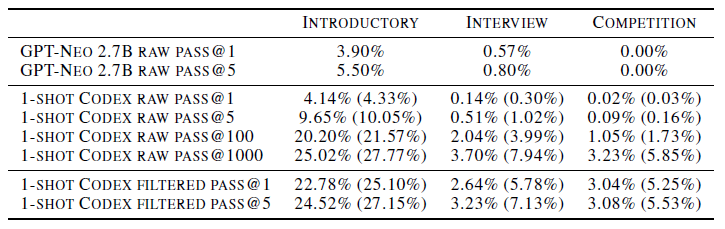
My motivation for this research assignment stems from a fascination with the future of open AI and machine learning. I pride myself on staying up to date with the latest technology and trends in our tech community. While during my search, I discovered a fascinating paper called Alpha Codium. This cutting-edge technology facilitates code-oriented iterative flow and aims to improve machine learning with code generation.

# Problems and Questions Fact-Checking

There are several challenges when it comes to natural language generation. These challenges include ensuring syntactic correctness and semantic consistency, leveraging domain-specific knowledge, resolving ambiguities, optimizing code for efficiency, handling complex logic, implementing anagrams, arrow handling, and debugging mechanisms. This promotes maintainability and evolvability and integrates seamlessly into existing development workflows. This allows code to be generated that accurately reflects high-level specifications, while balancing simplicity and performance optimization. It requires careful consideration and expertise. Alpha Codium must effectively address these challenges in order to realize its potential to enhance machine learning workflows and advance the field of artificial intelligence.

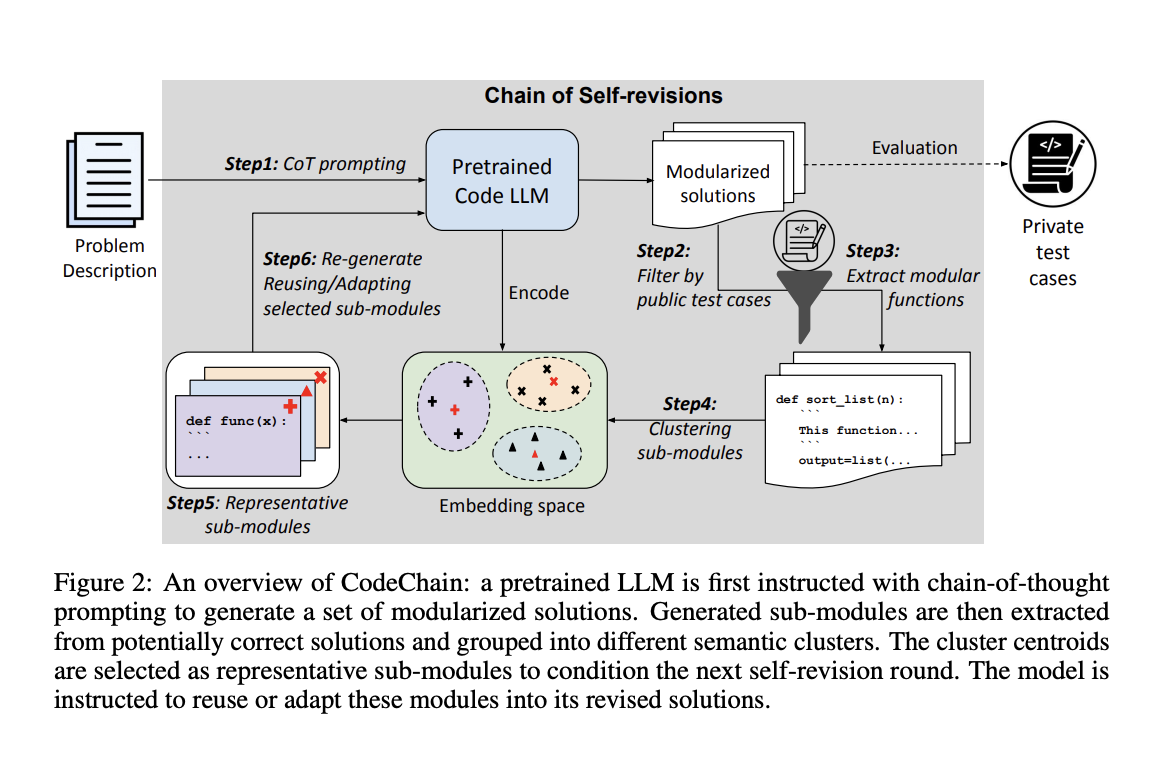
# Related Works One

Method number one involves evaluating large language models trained on Codex train, which was derived from over 54 million popular software repositories hosted on GitHub. The APPS dataset comprises 5000 training and 5000 test examples of coding problems. This method helps identify common model errors and aids in improving these errors, enhancing speed efficiency, and providing improved solutions.



# Related Works Two

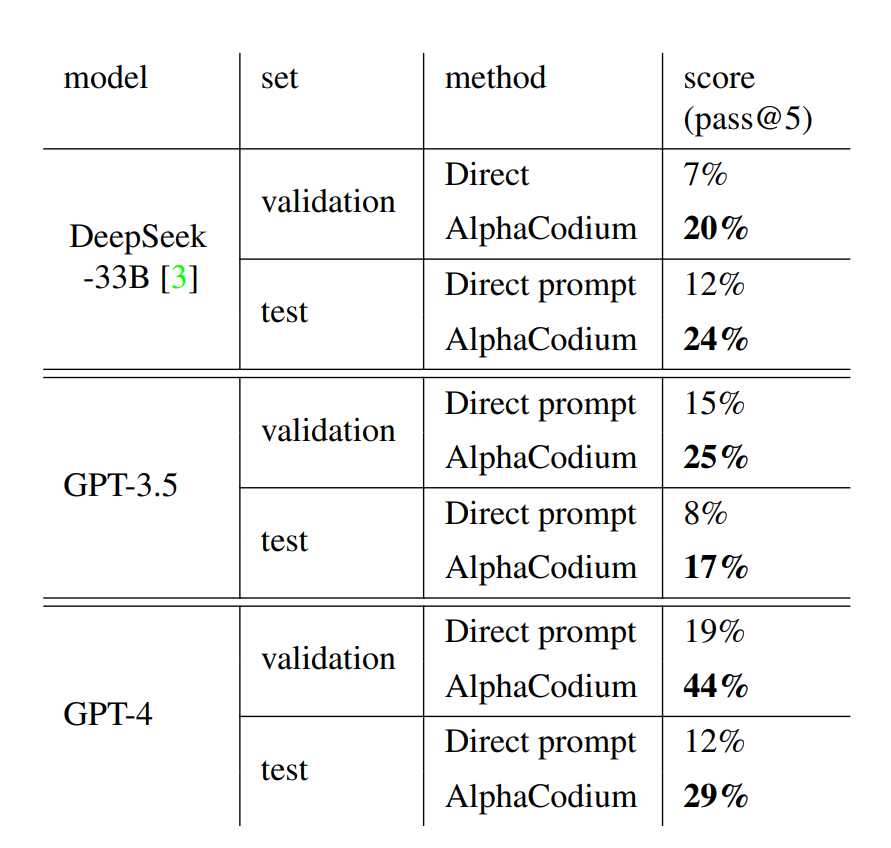
Method two, for related work, introduces Code Chain, a fascinating code generation approach that forms a chain of self-revisions iteratively to improve modularized code. The process involves identifying representatives with sub-modules within the codebase and using them as templates for generating new modules or revisions. Since the code is refined in this manner, Code Chain aims to promote modularity, scalability, and maintainability in software projects. It also adopts an approach to potentially streamline the development process, enhance code quality, and facilitate easier updates and maintenance in challenging software systems. Below, there will be an image illustrating the Cold Chain process.



# Method That Was Duplicate

The method that was duplicated is Alpha Codium, which is a flow engineer with an approach to generating code by Large Language Models (LLMs) that proves the performance of coding problems. It tests this based on a multi-stage code-oriented iteration flow that continues to refine and improve, generating code with different tests, achievements, and with better accuracy.

While Alpha Codium demonstrates a significant improvement over the traditional prompt engineering approach for both open-source and deep-seek and closed-source models of GPT, for example, the accuracy of GPT-4 increases from 19% with a single, well-designed direct prompt to 44% with the Alpha Codium flow. Just by looking at the chart below, it shows an interest in, and illustrates the efficiency of Alpha Codium’s iterative refinement approach in guiding Language Models to the most accurate solutions. As stated earlier in the paper, it is very interesting because it has the opportunity to generate more efficient code, increase accuracy, and provide better solutions overall.



# 

# 

# *Conclusion And Future Work*

Alpha Codium is very interesting, which relates to my future work of obtaining a dataset from GitHub that showcases user errors. The data I found is particularly intriguing for my future endeavors, as it highlights errors made by humans, ranging from common mistakes such as pull request issues to more regular ones. My goal is to continue working on this project and explore how Alpha Codium can help prevent some of these issues. Hopefully, we can keep moving forward with this project.

# References

Chen, Mark, et al. “Evaluating Large Language Models Trained on Code.” arXiv.Org, 14 July 2021, arxiv.org/abs/2107.03374. Accessed 23 Feb. 2024. (https://arxiv.org/abs/2107.03374)

Le, Hung, et al. “CodeChain: Towards Modular Code Generation through Chain of Self-Revisions with Representative Sub-Modules.” arXiv.Org, 28 Nov. 2023, arxiv.org/abs/2310.08992. Accessed 23 Feb. 2024. ( <https://arxiv.org/abs/2310.08992>)

Ridnik, Tal, et al. “Code Generation with Alphacodium: From Prompt Engineering to Flow Engineering.” arXiv.Org, 16 Jan. 2024, arxiv.org/abs/2401.08500. Accessed 23 Feb. 2024. ( https://arxiv.org/abs/2401.08500)