AlphaStack: AI-powered Project Generator via Multi-Agent Systems

Abstract

This paper presents AlphaStack, a novel approach to autonomous code generation utilizing multi-agent systems with iterative self-healing and comprehensive validation. AlphaStack transforms natural language descriptions into complete, production-ready codebases with Docker configurations and automated testing. Through a combination of a Planning Agent and a Correction Agent, the system autonomously resolves software errors without human intervention, ensuring high success rates across diverse programming paradigms. Our comprehensive evaluation demonstrates the efficacy of this approach on 40 programming challenges across four modern languages.

Introduction

The automation of software development has been a long-standing goal in computer science. With recent advancements in large language models (LLMs), there has been significant progress in code generation. However, generating complete, production-ready codebases from high-level natural language descriptions remains a challenging task. It requires not only generating syntactically correct code but also ensuring proper architectural design, resolving dependency conflicts, and verifying correctness through testing. AlphaStack addresses these challenges by employing a multi-agent architecture that separates planning and correction concerns, enabling iterative self-healing and robust validation in isolated environments.

Methodology

The AlphaStack methodology revolves around an intelligent multi-agent architecture consisting of a Planning Agent and a Correction Agent. The generation pipeline starts with analyzing the natural language input to create a software blueprint. This blueprint dictates the folder structure and file contents, encompassing source code, configurations, tests, and documentation. Crucially, AlphaStack integrates Docker-based validation to ensure the generated code is functional. It automatically creates Dockerfiles to sandbox build and test environments. If build or test failures occur, the Planning Agent analyzes the errors and formulates comprehensive fix strategies. The Correction Agent then executes these fixes, iteratively refining the codebase until successful validation or a maximum number of iterations is reached.

Architecture Diagram

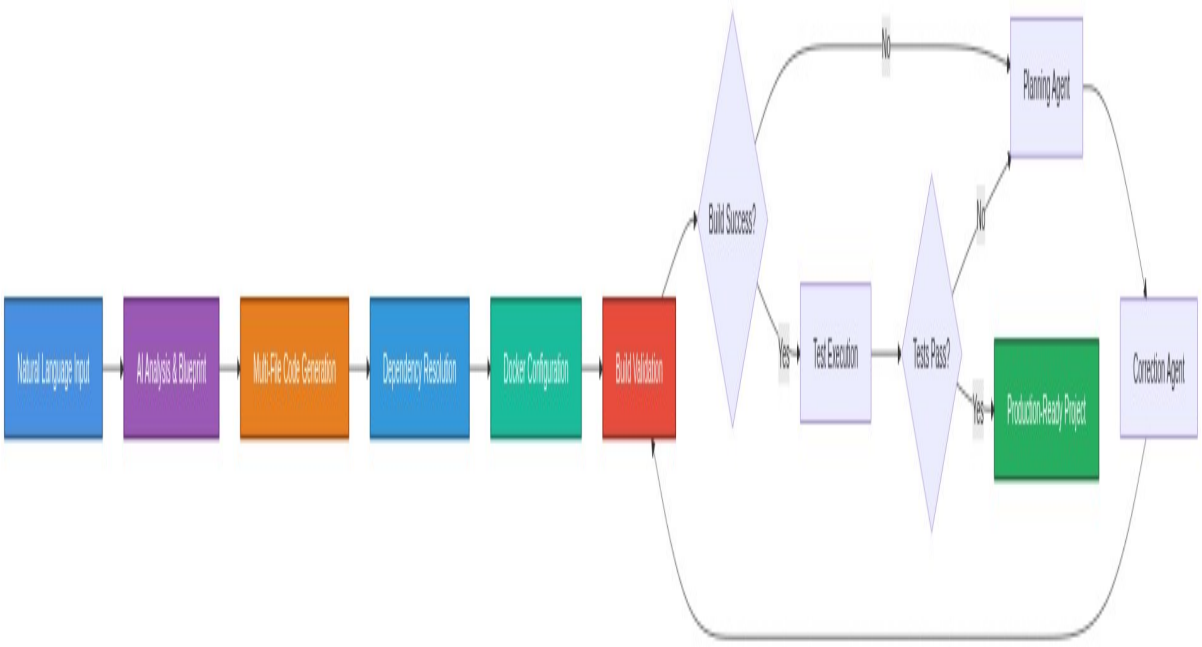

Figure 1: AlphaStack Multi-Agent Generation and Validation Pipeline

Results

We evaluated AlphaStack's capabilities using state-of-the-art language models on established benchmarks: HumanEval and the Multi-Domain Development Paradigm (MDDP). The models tested include GPT-5.2, GLM-5, MiniMax-M2.5, and Claude Sonnet 4.6. The results, as depicted in the graph below, highlight the strong performance of these models when integrated into the AlphaStack framework, demonstrating high success rates in generating functionally correct code.

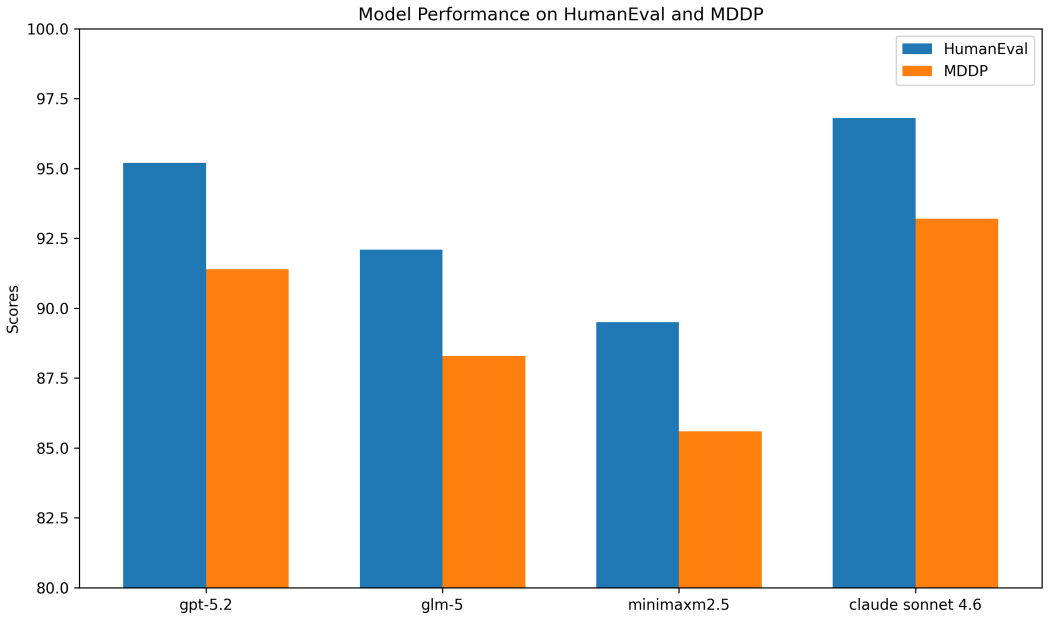

Figure 2: Model Performance on HumanEval and MDDP

Conclusion

AlphaStack introduces a highly effective methodology for autonomous code generation. By leveraging a multi-agent architecture with integrated iterative self-healing and Docker-based validation, it successfully bridges the gap between natural language intent and production-ready code. The robust evaluation demonstrates its versatility across various languages and complexities. Future work will focus on expanding language support, optimizing iteration efficiency, and integrating more advanced static analysis tools to further enhance the reliability of generated projects.

Supplementary Material

The source code, evaluation suite, and detailed benchmark logs for AlphaStack are available in the project repository. The evaluation suite includes 40 challenges across CUDA, Go, Rust, and TypeScript, categorized into four difficulty tiers ranging from fundamentals to production systems.