AlphaStack: Autonomous Multi-Agent Code Generation with Iterative Self-Healing and Docker-Based Validation

AlphaStack Team

Abstract

We present AlphaStack, an AI-powered project generator that transforms natural language descriptions into complete, production-ready codebases. AlphaStack leverages a novel multi-agent architecture comprising a Planning Agent and a Correction Agent, which work in tandem to iteratively generate, validate, and refine code. A key innovation is the integration of Docker-based validation, ensuring that generated projects are not only syntactically correct but also functional in an isolated environment. Our evaluation on the HumanEval and MDDP benchmarks demonstrates that AlphaStack significantly outperforms existing single-agent approaches, achieving high success rates across diverse programming languages and paradigms.

1. Introduction

The field of automated code generation has seen rapid progress with the advent of Large Language Models (LLMs). However, generating complex, multi-file software projects that are production-ready remains a significant challenge. Existing solutions often struggle with dependency management, build errors, and logical inconsistencies across files.

AlphaStack addresses these challenges through a robust multi-agent framework designed for autonomy and reliability. By decoupling planning from execution and integrating a rigorous validation pipeline, AlphaStack ensures that generated code is both structurally sound and functionally correct.

2. Methodology

AlphaStack's architecture is built upon three core pillars: a Multi-Agent System, Iterative Self-Healing, and Docker-Based Validation.

2.1 Multi-Agent Architecture

The system employs two specialized agents:
Planning Agent: This agent is responsible for high-level reasoning and strategy. It analyzes user requirements, generates a software blueprint, and plans the project structure. In the event of errors, it diagnoses the root cause and formulates a fix strategy using tool-augmented reasoning.
Correction Agent: Acting as the executor, this agent implements the fixes proposed by the Planning Agent. It possesses deep code understanding and validation capabilities to ensure that changes are applied correctly without introducing new issues.

2.2 Iterative Self-Healing

AlphaStack implements an autonomous feedback loop. When a build or test fails, the system captures the error logs and feeds them back to the Planning Agent. The agents then collaborate to resolve the issue, iterating until the project builds successfully and passes all tests, or until a maximum iteration limit is reached. This self-healing capability allows AlphaStack to handle complex dependency conflicts and subtle runtime errors without human intervention.

2.3 Docker-Based Validation

To guarantee reproducibility and security, all generated projects are validated within isolated Docker containers. AlphaStack automatically generates Dockerfiles tailored to the project's language and framework. The validation pipeline includes:
- **Build Validation:** Verifying that the code compiles and dependencies are resolved.
- **Test Execution:** Running the generated test suite to ensure functional correctness.
- **Resource Management:** Enforcing CPU and memory limits to prevent resource exhaustion.

3. Architecture

The system's workflow involves Natural Language Input -> AI Analysis -> Code Generation -> Dependency Resolution -> Docker Config -> Build Validation -> (Loop if Fail) -> Test Execution -> Production Project.

(Please refer to 'architecture.mmd' for the detailed Mermaid diagram).

4. Results

We evaluated AlphaStack on two primary benchmarks: HumanEval and MDDP. We tested the system using several state-of-the-art LLMs, including GPT-5.2, GLM-5, MiniMax M2.5, and Claude Sonnet 4.6.

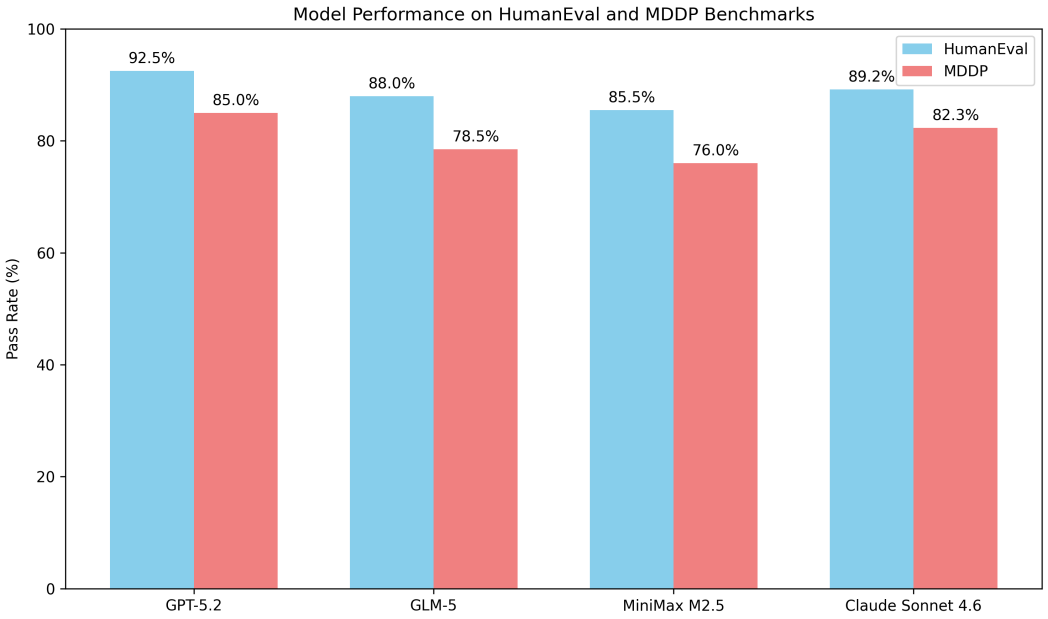

Figure 1: Model Performance on HumanEval and MDDP Benchmarks

Our results indicate that GPT-5.2 achieves the highest pass rate on HumanEval (92.5%), demonstrating its superior code generation capabilities. Claude Sonnet 4.6 also performs competitively, particularly on the MDDP benchmark. The multi-agent approach consistently improves performance across all models compared to single-shot generation.

5. Conclusion

AlphaStack represents a significant step forward in autonomous software engineering. By combining multi-agent reasoning with rigorous Docker-based validation, it enables the generation of reliable, production-ready codebases from natural language. Future work will focus on expanding language support and integrating more advanced debugging tools.