Weekly Report – Agentic Training

Name – Anbhi Thakur

Date – 10 August , 2025

---

1. Overview

Agentic training refers to the process of creating autonomous AI agents that can plan, reason, and act toward achieving specific goals with minimal human supervision. For coding-related tasks, this involves combining LLM-based reasoning with tool usage, environment interaction, and iterative self-improvement.

---

2. Objectives for the Week

- Understand the core concept of agentic AI in the context of programming.
- Identify training methodologies for coding agents.
- Outline required components (data, tools, frameworks).
- Explore evaluation benchmarks for performance tracking.

---

3. Key Learnings

3.1 Definition & Characteristics

- Agentic AI = LLM + autonomy + multi-step reasoning + ability to interact with external tools.

- For coding, the agent should:
  - Interpret problem statements.
  - Plan a solution approach.
  - Write, test, and debug code.
  - Refactor or optimize code based on feedback.

3.2 Training Requirements

- Data Sources:
  - Public coding datasets (CodeSearchNet, The Stack, GitHub repos with permissive licenses).
  - Problem-solving datasets (LeetCode-like tasks, competitive programming archives).
  - Real-world codebases for multi-file reasoning.
- Skills to Train:
  - Syntax understanding (language-specific knowledge).
  - Algorithmic thinking.
  - Debugging & error handling.
  - API & library usage.
  - Code refactoring for efficiency.
- Architectural Needs:
  - Base LLM (e.g., Code LLaMA, GPT-style model).
  - Agent loop with:
    - Planning module (decides next action).
    - Tooling interface (access to compilers, linters, Git).
    - Memory (short-term for active task, long-term for reusable knowledge).

3.3 Training Methods

- Supervised Fine-Tuning (SFT): On high-quality problem-solution pairs.

- Reinforcement Learning (RLHF/RLAIF): Reward correct solutions, efficiency, minimal bugs.
- Curriculum Learning: Start from simple problems → scale to complex multi-module projects.
- Simulated Task Environments: Interactive coding sandboxes for trial-and-error learning.

3.4 Tools & Frameworks

- LangChain / LlamaIndex for orchestration.
- OpenAI Function Calling or Toolformer-style training for tool use.
- Eval Harness for automated testing.
- Docker-based sandboxes for safe execution of untrusted code.
- CI/CD pipeline for continuous evaluation & retraining.

3.5 Benchmarks for Coding Agents

- HumanEval / MBPP for single-function correctness.
- CodeContests for competitive programming.
- Multi-File Reasoning Benchmarks (e.g., SWE-bench).
- Execution Accuracy (%) and Test Coverage.

---

4. Challenges Identified

- Ensuring safety (avoid harmful or insecure code generation).
- Generalization to new coding styles and domains.
- Minimizing hallucinations (incorrect but confident answers).
- Efficient tool integration without excessive API calls.

---

5. Next Steps

- Build a prototype agent loop for solving coding tasks.
- Start collecting & preprocessing training data.
- Set up evaluation pipeline with benchmark problems.
- Experiment with tool-assisted reasoning (e.g., debugging via automated stack trace analysis).

---

6. Key Takeaway:

Agentic training for coding agents is not just about fine-tuning an LLM; it requires equipping it with a decision-making loop, reliable memory, tool access, and reinforcement-based learning in a controlled execution environment.