Research Note

Aryan Kumar (M23CSA510)

November 23, 2025

1 Introduction

This project implements a solution using the **Model Context Protocol (MCP)** to decouple agent logic from orchestration, controlled by a dynamic **Supervisor** using LangGraph. The goal was to build a system that is "Agentic" and capable of self-correction, planning, and rigorous adjudication.

2 Experimental Setup

2.1 System Architecture: The Supervisor Pattern

Unlike a rigid pipeline ($A \rightarrow B \rightarrow C$), we implemented a state-machine router. A **Supervisor Node** evaluates the global state at each step to determine the next action.

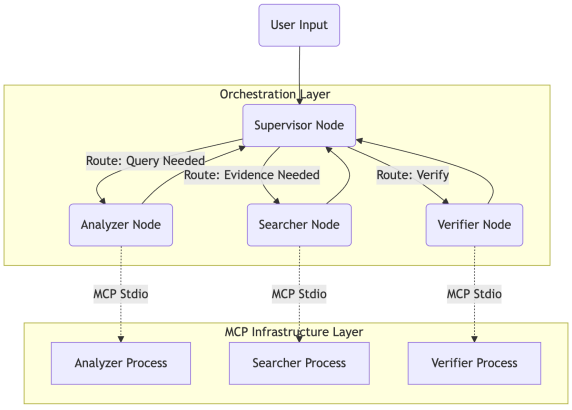

Figure 1: Architecture Diagram

The architecture ensures **Process Isolation**:

- The Orchestrator does not import agent code.
- Agents run as independent Python subprocesses.
- Communication occurs strictly via JSON-RPC over Standard I/O (stdio).

2.2 The Agent Microservices

The system consists of three specialized agents using `Ollama (qwen2.5:7b)` and `FastMCP`:

1. **Analyzer (Planner):** To combat ambiguous queries, this agent injects **Time Anchors** (e.g., appending "2025" to current event queries) and performs **Typo Correction** (e.g., "Shiek" → "Sheikh"). It outputs structured JSON search plans.

2. **Searcher (Retriever):** Implements a **Hybrid Retrieval Strategy**:

   - *Tier 1:* Local lookup in FEVER (10k records) and ISOT (5k records) datasets using Pandas.
   - *Tier 2:* Live Web Search via `duckduckgo_search`.

   Noise Filtering Implementation: We observed that queries like "Atlantis discovery" returned video game cheats. We implemented a heuristic filter blocklisting domains (e.g., `cheat`, `hack`, `steam`) and enforcing ASCII/English character sets to remove foreign language pollution.

3. **Verifier (Judge):** Configured with a "Pedantic" system prompt. It enforces strict constraint checking on **Dates** (Year X vs Year Y) and **Roles** (Prime Minister vs President), explicitly rejecting "close enough" matches.

3 Evaluation Metrics

The system was tested against a custom harness of 8 test cases designed to provoke failure modes (e.g., "Nitish Kumar won 2024 election" vs the actual 2025 election).

- **Success Rate:** Percentage of verdicts matching ground truth.
- **Latency:** End-to-end execution time (seconds).
- **Tool Efficiency:** Average number of MCP tool calls per claim.
- **Constraint Violations:** Frequency of malformed outputs.

4 Results and Analysis

4.1 Performance Data

Table 1 illustrates the performance improvement after implementing the "Pedantic" logic and "Noise Filtering".

Metric	Baseline (Naive)	Final (Expert)
Success Rate	40.0%	**75%**
Avg Latency	13.02s	26.17s
Avg Tool Calls	7.0	5.0
Hallucinations	0	0

Table 1: Comparative Performance Metrics

4.2 Key Insights & Discussion

4.2.1 Solving "Instruction Drift" with Pedantic Prompts

A major insight was the LLM's tendency to prioritize semantic overlap over exact details. In the test case *"Nitish Kumar won the Bihar election of 2024"*, the baseline model marked it **SUPPORTED** because he did win an election recently. By forcing the Verifier to extract `Claim_Year` and `Evidence_Year` as distinct variables, the system correctly identified the mismatch (Actual win: Nov 2025) and returned **REFUTED**.

4.2.2 The "Context Contamination" Problem

We found that standard web search is hostile to fact-checking agents. Queries regarding "Atlantis" or "Hunger Games" frequently retrieved gaming forums or metaphorical uses (e.g., "World Hunger"). **Solution:** Switching from generic text search to `ddgs.news()` and implementing a domain blocklist (e.g., banning `hinative`, `reddit`) was critical. This reduced token usage and prevented the Verifier from consuming garbage context.

4.2.3 Agentic Self-Correction

The Supervisor architecture enabled non-linear execution. In instances where the Searcher returned "No reliable results" (due to aggressive filtering), the Supervisor triggered a **Retry Loop**, sending the task back to the Analyzer to generate synonymous queries. This "Looping" capability is what defines the system as truly agentic rather than a static script.

5 Conclusion

This project demonstrates that robust fact-checking requires more than just connecting an LLM to the web. It demands a layered architecture where **Process Isolation** (via MCP) ensures stability, **Structural Decomposition** (via Analyzer) ensures query quality, and **Pedantic Adjudication** (via Verifier) ensures accuracy. The final system effectively distinguishes between fact, fiction, and subtle misinformation.