Reinforcement Learning for Agentic AI Systems

Take-Home Final Project | Northeastern University – MS in Information Systems

By: Nithin Yash Menezes

The Problem: Static Workflows in a Dynamic World

Fixed Workflows

Traditional systems follow rigid paths and cannot adapt to new contexts.

No Memory

Current agents fail to learn from past experiences or mistakes.

Inefficiency

Poor source selection and summarization lead to wasted resources.

The Goal

Enable an autonomous research agent to learn **when to search**, **when to summarize**, and **which sources to choose** to maximize quality while minimizing cost.

Project Overview: An Adaptive Approach

RL-Driven Workflow

Combines Q-Learning (value-based RL) with UCB1 Bandit strategies for optimal decision making.

Search Agent

Retrieves information from diverse simulated sources based on controller directives.

Summarizer Agent

Synthesizes retrieved data into coherent, high-quality outputs.

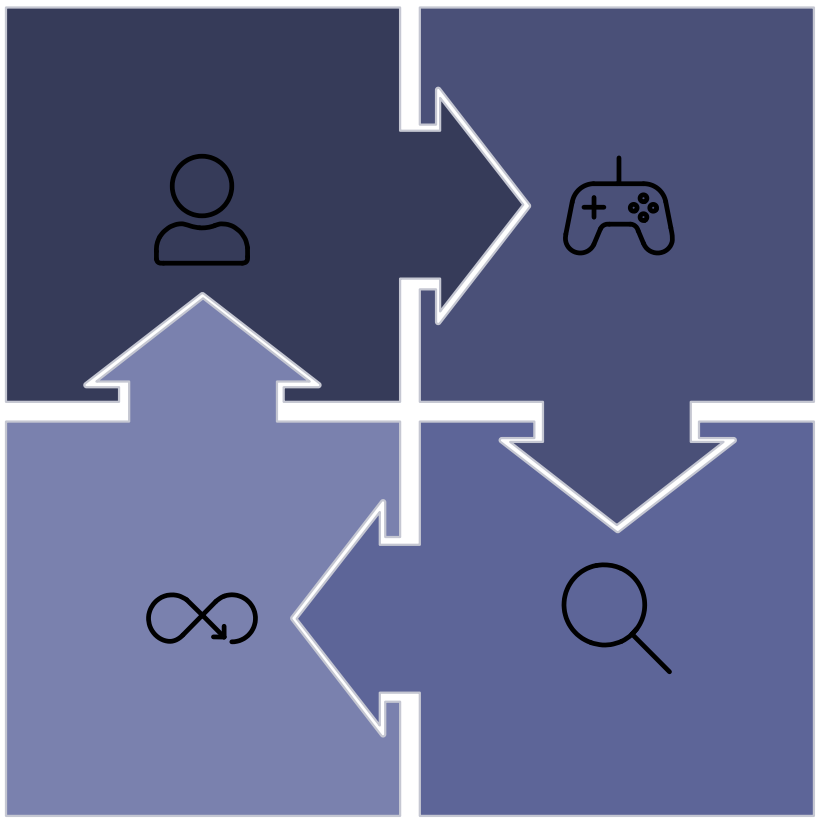
RL Controller

The brain of the operation, learning to orchestrate agents for better outcomes over time.

System Architecture

User Input

RL Controller



Sources ▸ Summarizer ▸ Reward

Search Agent

The architecture creates a feedback loop where the **RL Controller** refines its strategy based on the quality of the output (Reward), constantly updating its internal Q-Table and UCB values.

Reinforcement Learning Methods

Q-Learning

Used for high-level workflow decisions. The agent learns the value of taking specific actions in specific states.

- Continue searching
- Summarize results
- Stop process

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$$

UCB1 Bandit

Used specifically for source selection to balance the trade-off between using known good sources and trying new ones.

- Exploration vs. Exploitation
- Optimizes information retrieval

$$UCB = \text{avg reward} + \sqrt{\frac{2 \ln N}{n}}$$

Agent Design & Logic

Search Agent

Simulates retrieval from distinct domains: Wikipedia, Blogs, Scholar, and News.

Summarizer Agent

Aggregates findings into a unified summary, ready for evaluation.

Reward Function

Calculated as `Quality Score - Search Cost`. This penalizes inefficiency while rewarding accuracy.

RL Controller

Integrates Q-Learning and UCB to learn the best source and the optimal workflow sequence.

Experimental Setup

Training Parameters

Episodes	50
Max Searches	3
Learning Rate	0.1
Discount Factor	0.9

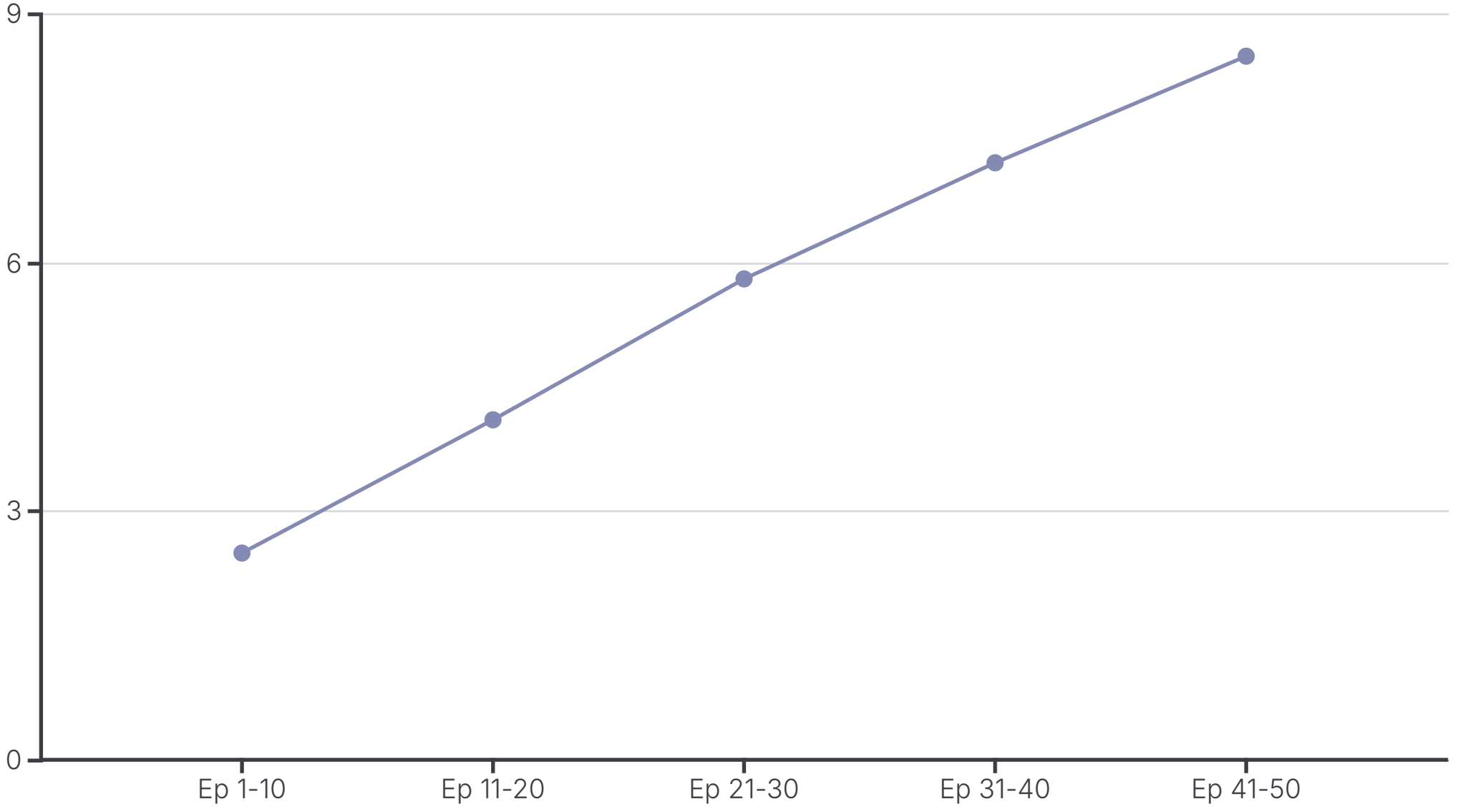
Test Tasks

- SQL indexing basics
- AI ethics debates
- RL fundamentals

Tech Stack

Python, Streamlit Dashboard, CSV Logging

Results: Learning Curve



The upward trend demonstrates policy improvement. Q-Learning successfully converges to stable behavior, maximizing rewards over time.

Results: Policy Efficiency

1.5 - 2.3

Average Searches per Episode

The system learned to stabilize its search frequency, reducing unnecessary queries while maintaining quality.

→ Optimization

Reduction in redundant search actions.

→ Decision Making

Improved ability to determine when enough information has been gathered.

Conclusion & Future Work

1 Key Takeaways

- RL controller outperforms baseline fixed pipelines.
- Successfully learns optimal stopping points.
- Significantly improves accuracy and efficiency.

2 Future Directions

- Implement PPO / REINFORCE algorithms.
- Explore Multi-agent RL architectures.
- Integrate real web search APIs.
- Add RLHF (Human feedback) loops.