# MALT: REVOLUTIONIZING MULTI-AGENT LLM TRAINING

https://www.linkedin.com/in/nimit-kothari/

# 📊 Introduction to MALT

01

🔍 **Problem Statement:**

- 🤖 Current LLMs operate in isolation
- 🔗 Missing joint training methods
- 🏗️ No established multi-agent framework

💡 **Why It Matters:**

- 🔄 Growing need for autonomous systems
- 🎯 Limited by individual model training
- 👥 Gap in collaborative AI development

✨ **Key Innovation:**

- 🌟 First Multi-agent LLM framework
- ⚡ Sequential specialized agent setup
- 🎓 Joint training with credit assignment

# 🔧 Architecture-Core Components and System Design

02

**🤖 THREE SPECIALIZED AGENTS:**

**GENERATOR (G) 📝**

- INPUT → QUESTION (Q)
- OUTPUT → INITIAL SOLUTION
- ROLE → SOLUTION GENERATION

**VERIFIER (V) 🔍**

- INPUT → GENERATOR OUTPUT + QUESTION
- OUTPUT → QUALITY FEEDBACK
- ROLE → CRITICAL EVALUATION

**REFINEMENT MODEL (R) ✨**

- INPUT → ALL PREVIOUS OUTPUTS
- OUTPUT → FINAL REFINED ANSWER
- ROLE → SOLUTION IMPROVEMENT

**➡ FLOW:**
**QUESTION → G → SOLUTION → V → FEEDBACK → R → FINAL ANSWER**

**https://www.linkedin.com/in/nimit-kothari/**

# 🔄 Training Process

## MALT Training Methodology

### 📊 Data Generation:

- Sampling Strategy:
  - 🌳 Tree-based sampling
  - 📈 $n^3$ trajectory generation
  - 🔀 Exponential solution space
- Value Attribution:
  - ✅/❌ Binary rewards
  - ⬅ Backward value propagation
  - 📏 θ = 0.5 threshold

03

### 🔄 Training Pipeline

1. 📥 **Initial Dataset Collection:** • Raw data preprocessing • Question-answer pairs setup • Quality filtering
2. 🌳 **Trajectory Expansion:** • Branching factor n application • Multiple solution paths generation • Search space exploration
3. ⚖️ **Credit Assignment:** • Value propagation through tree • Performance attribution • Role-specific feedback
4. 🎯 **Model-Specific Training:** • Individual agent optimization • Role specialization • Capability enhancement

# 🔄 Technical Implementation

## Implementation Details & Algorithms

**📊 Credit Assignment Strategy:**

1. **Value Functions:**

- 📐 V(ri,j,k,l) = R(ri,j,k,l) ∈ {0, 1}
- 📏 V(vi,j,k) = El[V(ri,j,k,l)]
- 📈 V(gi,j) = Ek[V(vi,j,k)]

2. 🎯 **Binarization Process:**

- θ = 0.5 threshold
- ✅ Values > 0.5 → correct
- ❌ Values ≤ 0.5 → incorrect

04

**🛠️ Training Methods:**

- 📚 SFT (Supervised Fine-Tuning)
- 🎯 DPO (Direct Preference Optimization)
- ⚡ LoRA adaptation

# 📊 Slide 5: Results & Performance

05

Experimental Results & Benchmarks

📈 **PERFORMANCE IMPROVEMENTS:**

1. 🧮 **MATH Dataset:**
   - 📉 Baseline: 49.50%
   - 📈 MALT: 56.50%
   - ⬆ Gain: +14.14%
2. 📊 **GSM8k Dataset:**
   - 📉 Baseline: 84.25%
   - 📈 MALT: 90.25%
   - ⬆ Gain: +7.12%
3. 🤔 **CSQA Dataset:**
   - 📉 Baseline: 74.50%
   - 📈 MALT: 81.50%
   - ⬆ Gain: +9.40%

🎯 **KEY FINDINGS:**

- 📈 Consistent improvements
- 🎯 Effective specialization
- 🤝 Successful collaboration

# 🚀 Future & Applications

## Practical Applications & Future Work

06

💡 **Current Applications:**

1. 🧮 **Complex Problem Solving:**
   - Mathematical reasoning
   - Research support
   - Code development
   - Creative tasks
2. 🛡️ **Safety Applications:**
   - Verification systems
   - Oversight mechanisms
   - Trusted AI systems

🔮 **Future Research:**

1. 🛠️ **Technical Improvements:**
   - PPO implementation
   - Dynamic thresholding
   - Search optimization
2. 📈 **Scaling Directions:**
   - Multi-agent expansion
   - Role diversification
   - Architecture scaling

📌 **Key Implementation Notes:**

- Temperature: $\tau = 0.3$
- Base model: Llama 3.1 8B
- LoRA parameters optimized
- Branching factor $n = 3$

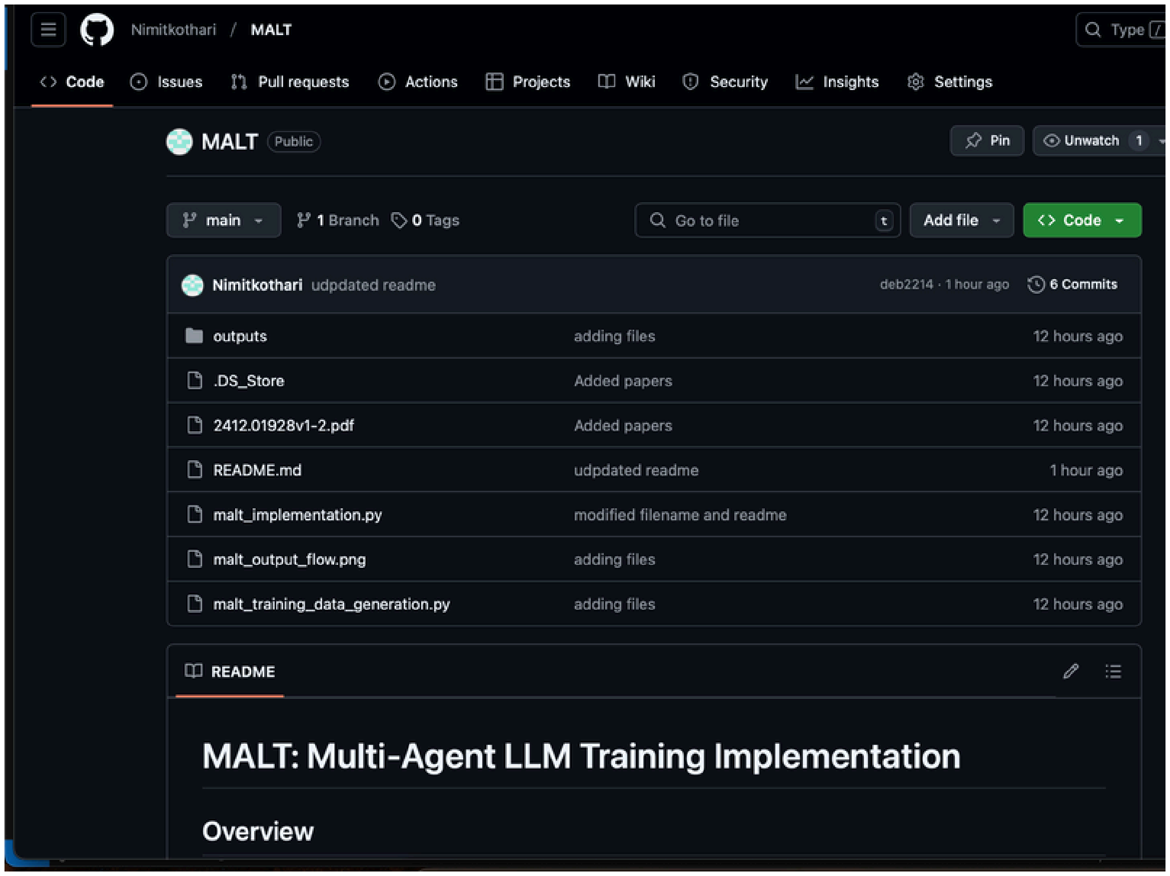
❓ **Research Questions:**

- 🤔 Optimal agent count
- 🎯 Role optimization
- 📊 Scaling efficiency

# Implementating the concept

## Code Base

06




https://github.com/Nimitkothari/MALT

https://www.linkedin.com/in/nimit-kothari/

NIMIT KOTHARI

# SAVE THIS POST IF YOU FIND IT USEFUL

https://www.linkedin.com/in/nimit-kothari/