ROCm’s Razor

Team: We Will ROCm You

Problem Solving Approach - Mathematical Classification of Problems

Fundamentally, each problem is can be classified into a specific form of mathematical process.

Once we can classify each problem into a mathematical one, the final (theoretical) step remaining is to translate it into a structured natural language problem that preserves the underlying logic.

This ensures that the LLM operates over a clean representation of the original mathematical reasoning, rather than surface heuristics.

Logical Reasoning Problem (1/2) - Propositional Logic

In the truth-teller and liar problem, each entity makes a statement.

We are then supposed to label each entity as truth-telling or lying such that all statements remain logically consistent.

Hence, we need to convey this logic of proof by contradiction to the model for correct answer generation.

For answer generation, the opposite is followed.

Create a free, arbitrary system and then derive a certain amount of consistent statements.

Logical Reasoning Problem (2/2) - Proof by contradiction

So the approach for this problem shall be:

1. Assume an entity is truthful or lying,
2. Propagate the consequences,
3. Reject inconsistent assignments.

Puzzles (1/2)- Graph Problem with Spatial Constraints

Seating arrangement problems can be modeled as graph problems with spatial or positional constraints.

Each entity (person) is a node, and each directional clue (e.g., "A sits to the left of B", "C sits opposite D") is a labeled edge encoding spatial relationships.

Puzzles (2/2)- Graph Problem with Spatial Constraints

To solve such problems, the LLM must:

1. Construct a spatial graph from the set of constraints.
2. Classify the final question into one of:
   - Location identification (e.g., "Where is A sitting?")
   - Entity tracing (e.g., "Who sits to the right of C?")
   - Positional relationships (e.g., "Who is between X and Y?")

The answer is then derived by traversing the graph in accordance with these constraints and the classified objective.

For question generation, we generate an arbitrary graph and simply define the constraints in natural language.

Blood Relations and Family Tree - Graph Traversal

The problems involving generations and family tree logic are simply Graph Traversal problems with labelled edges (for relationships).

Accordingly, the LLM must generate a graph based on the description in the question and parse it.

Once it is able to find the path between the nodes in question, its natural language capabilities must be used to label the relationships for the final answer.

For question generation, the opposite approach is taken. We define an arbitrary graph and then

Mapping Problem Structures to LLM Tuning Techniques

Problem Type	Mathematical Form	LLM Techniques Needed
Truth-Teller / Liar	Propositional Logic	Chain-of-Thought Prompt Engineering
Seating Arrangement	Graph + Spatial Constraints	Finetuning to construct graph CoT traversal
Family Trees	Graph + Transitive Logic	Finetuning to construct graph CoT traversal

Architecture

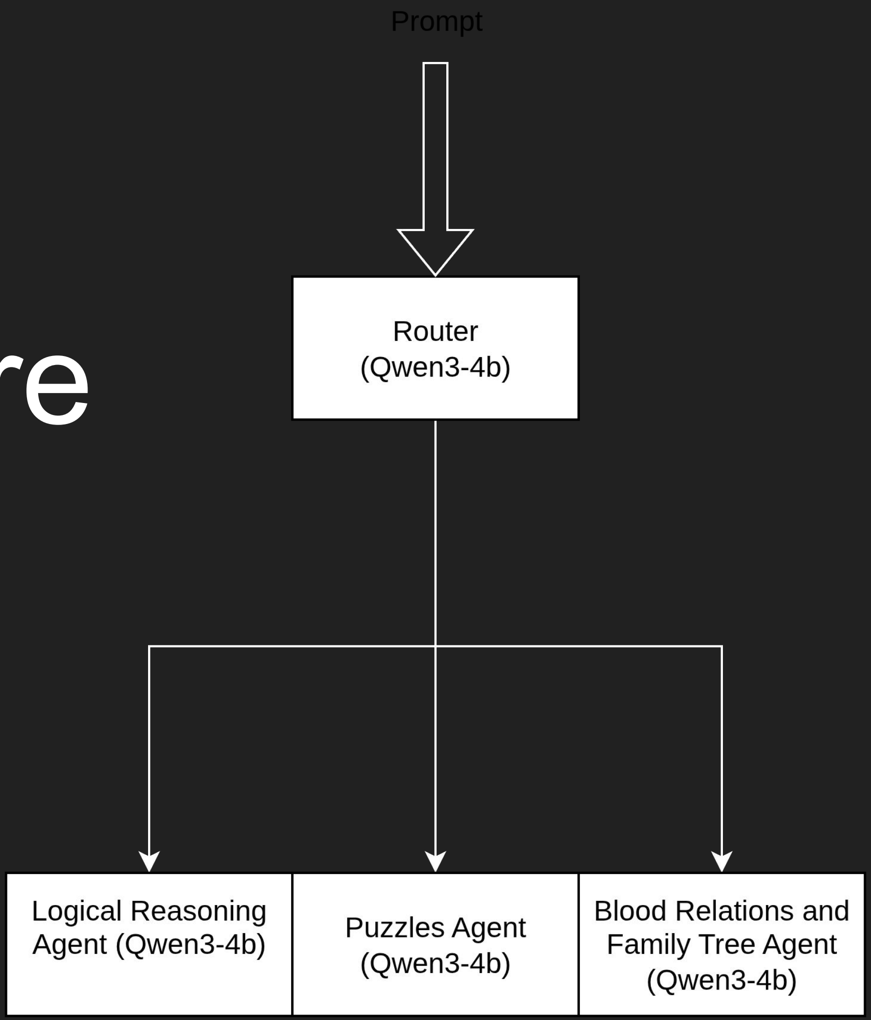
For the answer agent, due to the diverse logical structures of each problem type, a single model struggles to generalize across all of them with high accuracy.

While Mixture of Experts (MoE) is ideal for such modular specialization, our constraint was to use only Qwen3-4B, without modifying its internal architecture.

For the question agent, we used Chain of Thought methods for achieving the desired outcome.

Base Architecture (answer agent)

Agent Delegation


Prompt
Router (Qwen3-4b)
Logical Reasoning Agent (Qwen3-4b)
Puzzles Agent (Qwen3-4b)
Blood Relations and Family Tree Agent (Qwen3-4b)

Base Architecture - Agent Delegation

Design Overview:

1. Three Qwen3-4B models, each (eventually) finetuned for a specific problem class:
   - Truth-teller/Liar
   - Seating Arrangement
   - Family Trees
2. One Qwen3-4B "Router Agent":
   - Classifies the incoming question into one of the three categories
   - Delegates the task to the corresponding specialized model
3. The specialized model then handles decoding and answer generation

Implementation (so far) (1/2)

Step 1: Building the Agent Delegation Pipeline

- Developed a custom pipeline using the existing repo code to enable Agent Delegation
- Each incoming prompt is routed to the appropriate Qwen3-4B instance

Step 2: Handling GPU Memory Constraints

- Verified whether all four Qwen3-4B models could fit in GPU memory
- Found that VRAM usage remained stable even during simultaneous decoding
- We found no memory thrashing or unexpected slowdowns

Implementation (so far) - (2/2)

Step 3: Ensuring Real-Time Decoding Performance

- Experimentally ensured that decoding speed passed under the time constraints
- Result: The total inference time (including routing) remained practical, despite using four full-sized models

Step 4: Model Finetuning

- The final step was to finetune models for each problem
- However, due to a lack of enough data, we had to resort to prompt engineering

Final Outcome

The model achieves the following:

1. Correctly formatted questions and answers
2. Output in the constrained timeframe
3. Functioning Agent Delegation pipeline
4. (mostly) logically sane outcome

Thank You

Vedic Chawla
Spaarsh Thakkar
(IIIT Surat)