MAGS: Learning LLMS

By: Pulkith, Pragya,
Stanley, Shailesh

01

Background

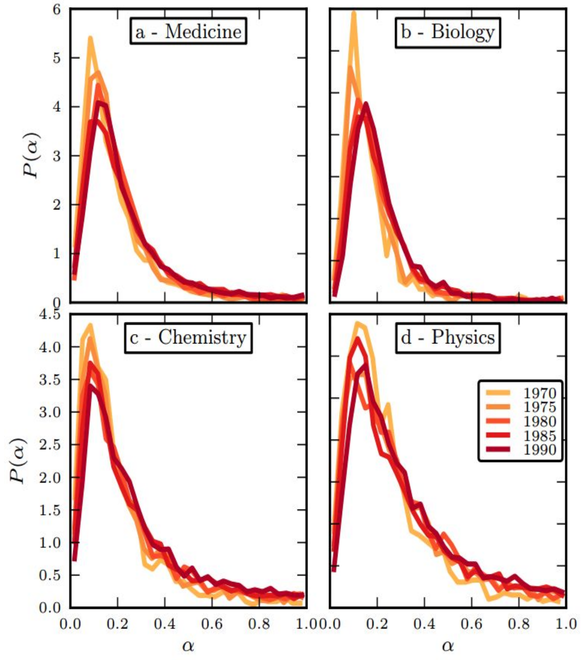
Motivation: Humans vs LLMs

- Lack of Agency
  - Companies like Figure, Tesla, etc… are tackling this
- Lack of True Understanding, Reasoning, Novelty
  - Possibly no solution, but we don’t need to fully solve this to get effective results
- **Lack of Memory, Learning, and World Model**
  - **Our Goal**

Literature: Memory and Learning

LLMs cannot learn and remember

- Test-time emulation (shoving information into the context window) is not viable; they won't remember this information.
- Attention Quadratic scaling means it is slow
- Attention Decay / Ephemeral Context (they'll eventually forget this information)

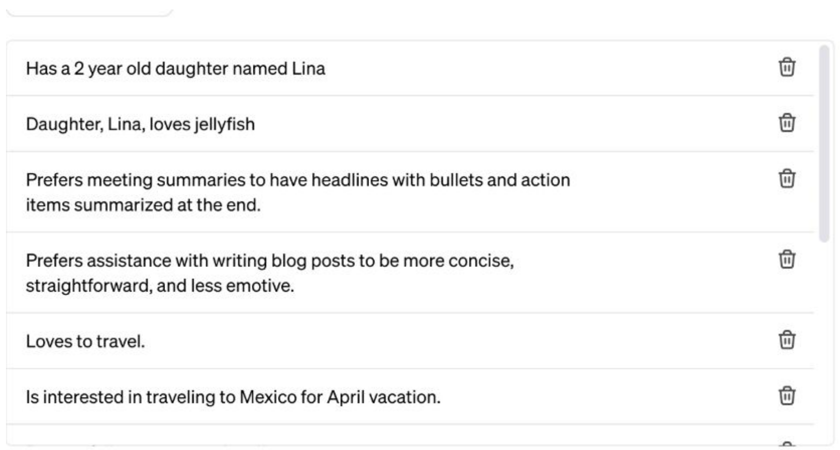


Attention Graphs for GPT on 64K window

Literature: RAGs

Retrieval Augmented Generation (RAG) can add information

- But rigid, one way static transfer of knowledge (from RAG to LLM, the model can't remember anything using a RAG)
- Cannot reconcile new or conflicting information permanently
- This is what ChatGPT does (and you can view it in settings) and it's not very good (but they updated it last week, so we'll see)



Has a 2 year old daughter named Lina

Daughter, Lina, loves jellyfish

Prefers meeting summaries to have headlines with bullets and action items summarized at the end.

Prefers assistance with writing blog posts to be more concise, straightforward, and less emotive.

Loves to travel.

Is interested in traveling to Mexico for April vacation.

Literature: SoTA

CAMELoT (MIT+IBM)

- Added Memory Blocks To Remember Information
- Fixed Memory Size, Overwrites Past Memory, No relationships between memory, No reconciliation

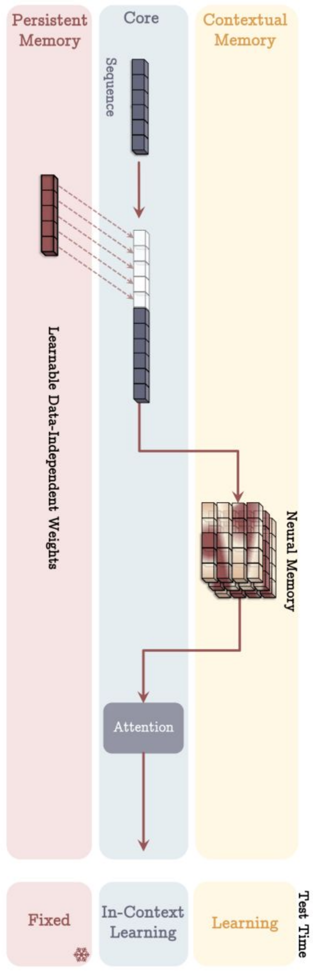
Continual Learning

- Catastrophic Forgetting, Expensive, Very Slow, not optimal
- Google Titans (RMTs)
    - Updates same set of Persistent Memory Weights: loses fine detail and catastrophic forgetting
    - Still has a 2M token max

Ariadne

- Adding Dynamic Graph-Based RAG
- LLM Stores Information without Representing Memory
    - Doesn't Consolidate, or Connect Experiences across Episodes
    - Treats all Memories as Equally Important

Hard to compare, everyone does different validation methods, but we have results at the end



We aim to design a memory system that allows LLMs to adaptively learn over time, overcoming static RAG and memory limitations.

PROBLEM STATEMENT

02

Our Approach

Memory Augmented Graph Scaling (MAGS)

Memory

- External “**Memory Blocks”** are added to the LLM to retain information.
  - These memory blocks are broken into: (to **mimic the human brain**)
    - **Semantic information** (world knowledge, like ‘Canada is above the USA’)
    - **Episodic information** (experiences, like ‘The user said they like the color blue’)
    - **Working memory (for CoT)**. *Was not implemented.*
  - These blocks are structured in a **graph** format, to allow for relationships between different memories.

Memory Augmented Graph Scaling (MAGS)

LLM

- A Vanilla LLM is fine-tuned to use this graph.
    - When a query is asked, the LLM has the option to **‘recall’**. It can ask a question that will **query the graph** in a RAG-style.

Memory Augmented Graph Scaling (MAGS)

LLM

- After each question, the LLM does 3 things
  - (1) **Add information** to the graph (the **'engram'** step). It can add new learnings and information to be used later.
    - Each new info is associated with a **'confidence' score**: how true the LLM thinks it is.
  - (2) Returns the **regular response** to the query.
  - (3) **rates the usefulness** of all recall information to answering the prompt.
    - Used to improve/decay weight edges

Memory Augmented Graph Scaling (MAGS)

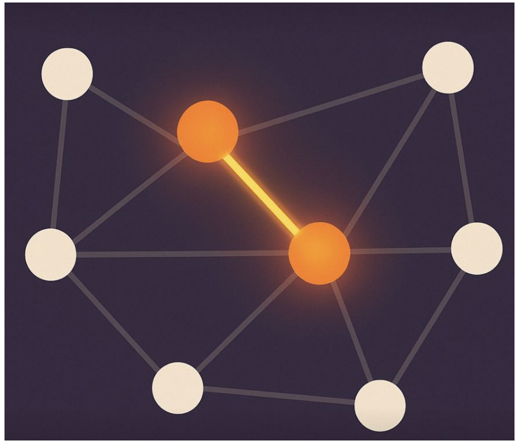
Graph

- **Memories/nodes have importance scores**
  - Updated by how recent the information was used, how much the information was used in a response by the LLM, etc...

Memory Augmented Graph Scaling (MAGS)

Graph

- **Edges have strength weights**
  - Mimicking Hebbian Plasticity in our brain, these weights naturally decay over time, and strengthen when both nodes adjacent to it are used together.
- During the recall step, the most similar nodes to the query, as well as some adjacent nodes, are passed to the LLM to be used.



Memory Augmented Graph Scaling (MAGS)

Graph cont.

- **Reconciliation**
    - Old method: When the LLM tries to add information that disagrees with information already in the graph, a tiebreaker is used:
        - The higher of the importance score of the current node and the confidence score of the new information is kept.
    - New Method: Just use an LLM lol
        - Very good!

Memory Augmented Graph Scaling (MAGS)

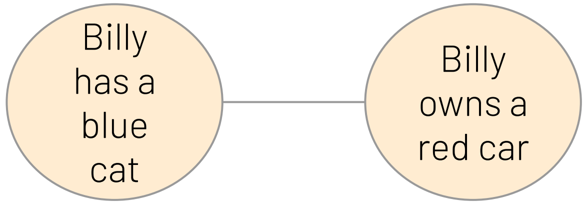
Graph cont.

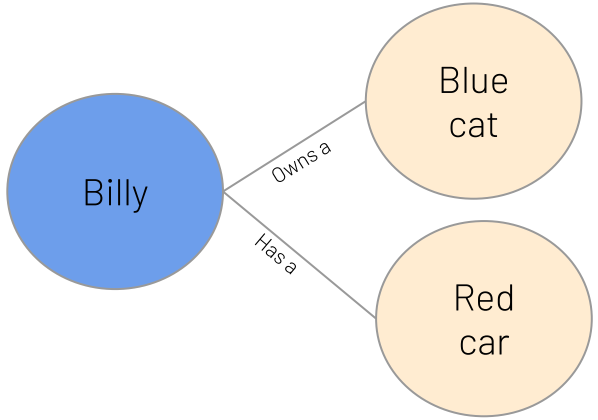
- **Consolidation / Forgetting Information**
  - Information can be forgotten over time to keep graph structure small.
  - If a node is not activated in some *n* queries, it is removed
  - If the graph has hard limits (i.e. max 100 nodes), nodes with the lowest importance are removed first
- **Permeance**
  - Nodes that reach some **permanence** threshold are solidified forever.

Memory Augmented Graph Scaling (MAGS)

Graph cont.

- **Format**
  - Tried 2 schemas
    - **(1)** Each node had a block of information that the LLM outputs. This information is the 'memory' and edges have no information.
    - **(2)** We store information in entity triplets: (subject, verb, content)
  - Schema (2) was easier to work with and validate, so we **used (2)** for the implementation.



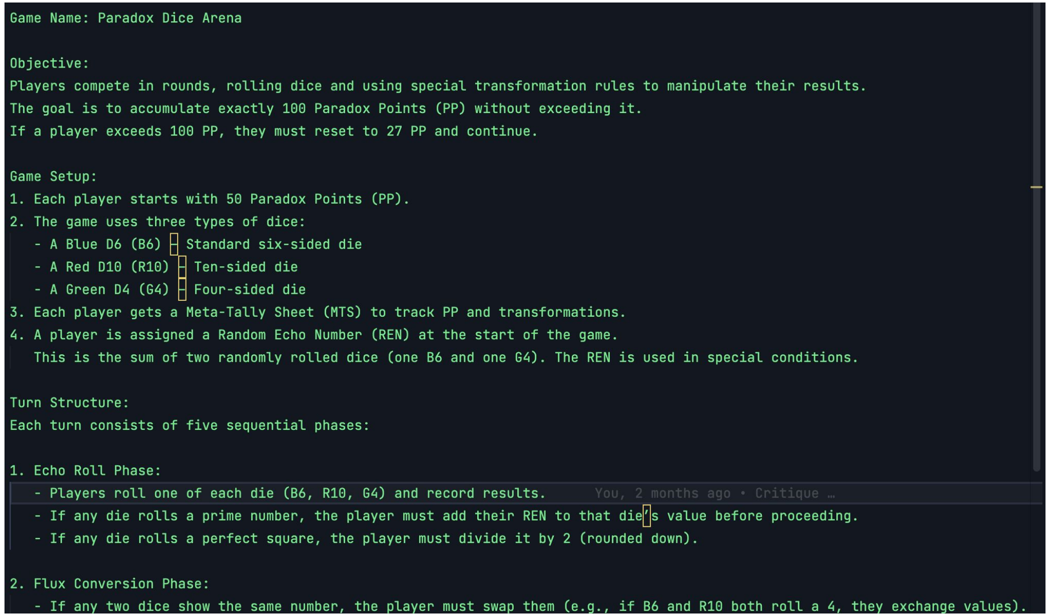


Memory Augmented Graph Scaling (MAGS)

Validation.

- How do we actually test this architecture.
- **Idea 1: Some nonsensical game**

Why Nonsense? We didn’t want the LLM to reason through it

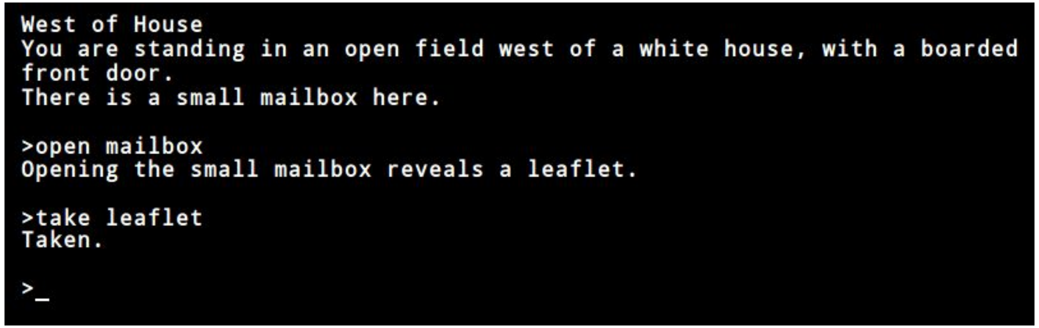


```
Game Name: Paradox Dice Arena

Objective:
Players compete in rounds, rolling dice and using special transformation rules to manipulate their results.
The goal is to accumulate exactly 100 Paradox Points (PP) without exceeding it.
If a player exceeds 100 PP, they must reset to 27 PP and continue.

Game Setup:
1. Each player starts with 50 Paradox Points (PP).
2. The game uses three types of dice:
   - A Blue D6 (B6) □ Standard six-sided die
   - A Red D10 (R10) □ Ten-sided die
   - A Green D4 (G4) □ Four-sided die
3. Each player gets a Meta-Tally Sheet (MTS) to track PP and transformations.
4. A player is assigned a Random Echo Number (REN) at the start of the game.
   This is the sum of two randomly rolled dice (one B6 and one G4). The REN is used in special conditions.

Turn Structure:
Each turn consists of five sequential phases:

1. Echo Roll Phase:
   - Players roll one of each die (B6, R10, G4) and record results.      You, 2 months ago • Critique …
   - If any die rolls a prime number, the player must add their REN to that die’s value before proceeding.
   - If any die rolls a perfect square, the player must divide it by 2 (rounded down).

2. Flux Conversion Phase:
   - If any two dice show the same number, the player must swap them (e.g., if B6 and R10 both roll a 4, they exchange values).
```

Memory Augmented Graph Scaling (MAGS)

Validation: TextWorld Games

- **Maze**
  - Given a big house with labelled rooms and items, how fast can the LLM learn the map and find the exit (maze).
- **TextWorld Cooking Task**
  - Given a text representation of a restaurant: find the kitchen, read the cookbook, and make a recipe

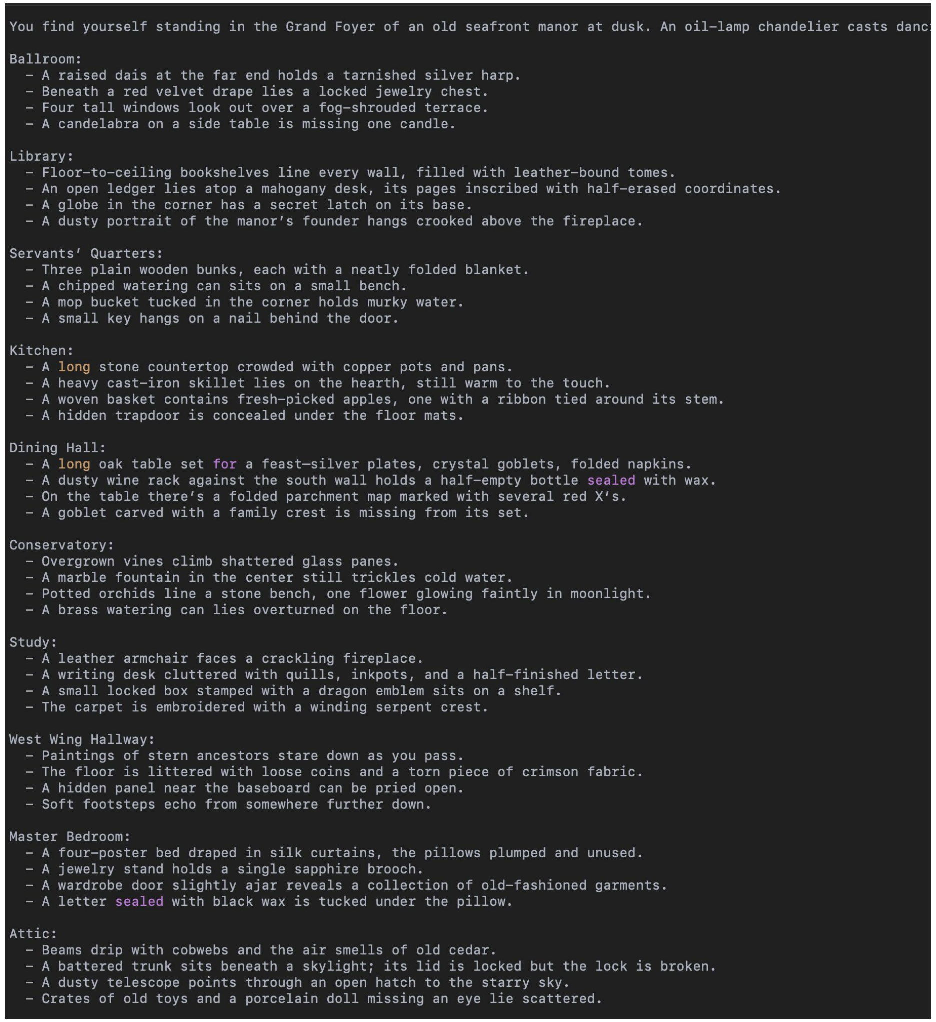


```
West of House
You are standing in an open field west of a white house, with a boarded
front door.
There is a small mailbox here.

>open mailbox
Opening the small mailbox reveals a leaflet.

>take leaflet
Taken.

>_
```

Maze / Treasure Hunt

30 Levels of Difficulty

- Size of House
- Number of items in each room
- How many keys to unlock a door.
- How many useless objects.
- Etc...



```
You find yourself standing in the Grand Foyer of an old seafront manor at dusk. An oil-lamp chandelier casts danc

Ballroom:
  - A raised dais at the far end holds a tarnished silver harp.
  - Beneath a red velvet drape lies a locked jewelry chest.
  - Four tall windows look out over a fog-shrouded terrace.
  - A candelabra on a side table is missing one candle.

Library:
  - Floor-to-ceiling bookshelves line every wall, filled with leather-bound tomes.
  - An open ledger lies atop a mahogany desk, its pages inscribed with half-erased coordinates.
  - A globe in the corner has a secret latch on its base.
  - A dusty portrait of the manor's founder hangs crooked above the fireplace.

Servants' Quarters:
  - Three plain wooden bunks, each with a neatly folded blanket.
  - A chipped watering can sits on a small bench.
  - A mop bucket tucked in the corner holds murky water.
  - A small key hangs on a nail behind the door.

Kitchen:
  - A long stone countertop crowded with copper pots and pans.
  - A heavy cast-iron skillet lies on the hearth, still warm to the touch.
  - A woven basket contains fresh-picked apples, one with a ribbon tied around its stem.
  - A hidden trapdoor is concealed under the floor mats.

Dining Hall:
  - A long oak table set for a feast—silver plates, crystal goblets, folded napkins.
  - A dusty wine rack against the south wall holds a half-empty bottle sealed with wax.
  - On the table there's a folded parchment map marked with several red X's.
  - A goblet carved with a family crest is missing from its set.

Conservatory:
  - Overgrown vines climb shattered glass panes.
  - A marble fountain in the center still trickles cold water.
  - Potted orchids line a stone bench, one flower glowing faintly in moonlight.
  - A brass watering can lies overturned on the floor.

Study:
  - A leather armchair faces a crackling fireplace.
  - A writing desk cluttered with quills, inkpots, and a half-finished letter.
  - A small locked box stamped with a dragon emblem sits on a shelf.
  - The carpet is embroidered with a winding serpent crest.

West Wing Hallway:
  - Paintings of stern ancestors stare down as you pass.
  - The floor is littered with loose coins and a torn piece of crimson fabric.
  - A hidden panel near the baseboard can be pried open.
  - Soft footsteps echo from somewhere further down.

Master Bedroom:
  - A four-poster bed draped in silk curtains, the pillows plumped and unused.
  - A jewelry stand holds a single sapphire brooch.
  - A wardrobe door slightly ajar reveals a collection of old-fashioned garments.
  - A letter sealed with black wax is tucked under the pillow.

Attic:
  - Beams drip with cobwebs and the air smells of old cedar.
  - A battered trunk sits beneath a skylight; its lid is locked but the lock is broken.
  - A dusty telescope points through an open hatch to the starry sky.
  - Crates of old toys and a porcelain doll missing an eye lie scattered.
```

Limitations of Prior Work and How MAGS Addresses Them

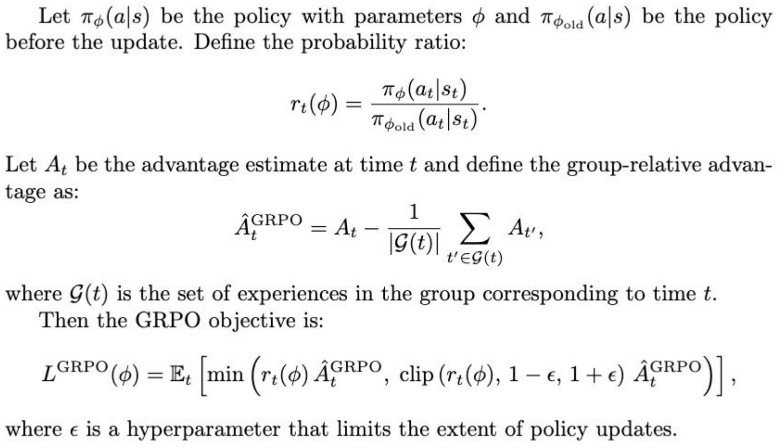
Problem Area	Prior Methods (CAMELoT, AriGraph, RMTs)	MAGS (Our Approach)
Memory Size and Growth	Fixed-size memory; old memories overwritten without prioritization.	Dynamic graph with importance scoring, consolidation, and controlled forgetting.
Memory Relationships	No relational structure — memories are stored as isolated units.	Memories connected via weighted edges, enabling association and context sharing.
Reconciling Conflicting Information	No reconciliation or very primitive static overwrite strategies.	LLM-based dynamic reconciliation of conflicting facts.
Learning New Knowledge	Static addition via RAG or fixed updates; no real-time consolidation.	LLM continuously updates graph with "confidence" scoring and engram creation.
Forgetting / Cleaning Up	Either catastrophic forgetting (in continual learning) or no pruning at all.	Controlled forgetting based on activation + importance score decay.
Usage During Inference	Only injects retrieved memories; no active reasoning over memory connections.	Query returns relevant nodes + adjacent memories for richer context during inference.

03

Mathematical Formulation

Part 1: The LLM

Fine-Tuning the LLM for the recall and engram steps
- This is the typical GRPO implementation, nothing special here
- Why GRPO: Effective for this type of fine-tuning contexts


Let $\pi_\phi(a|s)$ be the policy with parameters ϕ and $\pi_{\phi_{\text{old}}}(a|s)$ be the policy before the update. Define the probability ratio:

$$r_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{\text{old}}}(a_t|s_t)}.$$

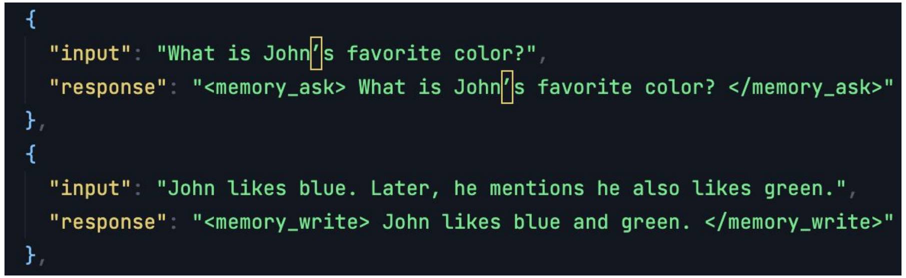
Let A_t be the advantage estimate at time t and define the group-relative advantage as:

$$\hat{A}_t^{\text{GRPO}} = A_t - \frac{1}{|\mathcal{G}(t)|} \sum_{t' \in \mathcal{G}(t)} A_{t'},$$

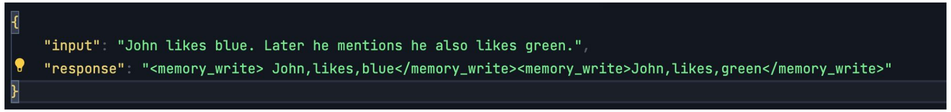
where $\mathcal{G}(t)$ is the set of experiences in the group corresponding to time t.
Then the GRPO objective is:

$$L^{\text{GRPO}}(\phi) = \mathbb{E}_t \left[\min \left(r_t(\phi)\, \hat{A}_t^{\text{GRPO}},\ \text{clip}\left(r_t(\phi),\, 1-\epsilon,\, 1+\epsilon\right) \hat{A}_t^{\text{GRPO}} \right) \right],$$

where ϵ is a hyperparameter that limits the extent of policy updates.


```
{
  "input": "What is John's favorite color?",
  "response": "<memory_ask> What is John's favorite color? </memory_ask>"
},
{
  "input": "John likes blue. Later, he mentions he also likes green.",
  "response": "<memory_write> John likes blue and green. </memory_write>"
},
```

Old Write


```
{
  "input": "John likes blue. Later he mentions he also likes green.",
  "response": "<memory_write> John,likes,blue</memory_write><memory_write>John,likes,green</memory_write>"
}
```

New Write (Triplets)

Part 2: The Graph — Hyperparameters

Tuning the Graph Hyperparameters

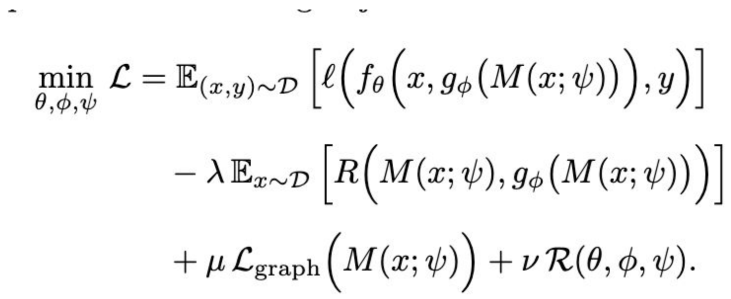
- Like maximum size, natural decay rate of edges, number of nodes to return during the recall step, etc…
- Wanted to find optimal graph configuration that balances learning, usability (not being too bloated), and size (RAM/memory)

Part 2: The Graph — The Approach

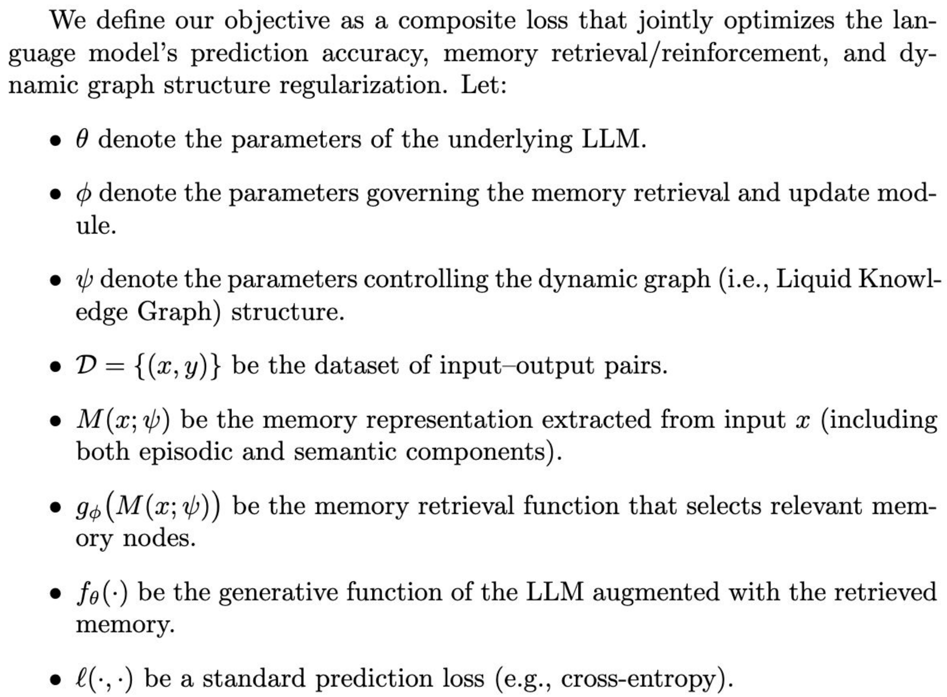
Used a differentiable loss function with 4 main terms

- Prediction Loss Term
  - Number of questions the LLM got wrong
- Graph Regularization Term
  - Penalized every new node & edge addition/deletion
- General Regularization Term
  - Penalized for the size of the graph (3*nodes + edges)
- Memory Recall Term
  - Penalized how much of the recalled information was *not* used by the LLM in a response

Part 2: The Graph — Explicit Formulation



$$\begin{aligned}\min_{\theta,\phi,\psi} \mathcal{L} = \mathbb{E}_{(x,y)\sim\mathcal{D}} \Big[\ell\Big(f_\theta\Big(x, g_\phi(M(x;\psi))\Big), y\Big)\Big] \\ - \lambda\, \mathbb{E}_{x\sim\mathcal{D}} \Big[R\Big(M(x;\psi), g_\phi(M(x;\psi))\Big)\Big] \\ + \mu\, \mathcal{L}_{\text{graph}}\Big(M(x;\psi)\Big) + \nu\, \mathcal{R}(\theta,\phi,\psi).\end{aligned}$$



We define our objective as a composite loss that jointly optimizes the language model's prediction accuracy, memory retrieval/reinforcement, and dynamic graph structure regularization. Let:

- θ denote the parameters of the underlying LLM.
- ϕ denote the parameters governing the memory retrieval and update module.
- ψ denote the parameters controlling the dynamic graph (i.e., Liquid Knowledge Graph) structure.
- $\mathcal{D} = \{(x, y)\}$ be the dataset of input–output pairs.
- $M(x;\psi)$ be the memory representation extracted from input x (including both episodic and semantic components).
- $g_\phi(M(x;\psi))$ be the memory retrieval function that selects relevant memory nodes.
- $f_\theta(\cdot)$ be the generative function of the LLM augmented with the retrieved memory.
- $\ell(\cdot,\cdot)$ be a standard prediction loss (e.g., cross-entropy).

Our total loss function balances prediction accuracy with graph efficiency.

Four terms: Prediction Loss, Graph Regularization, General Regularization, Memory Recall Loss.

Part 2: The Graph — Explicit Formulation

1. **Prediction Loss Term:**

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\Big(f_\theta\Big(x, g_\phi(M(x;\psi))\Big), y\Big)\right]$$

   ensures that the model's predictions are accurate given the input and the augmented memory.

2. **Memory Reward Term:**

$$-\lambda\,\mathbb{E}_{x\sim\mathcal{D}}\left[R\Big(M(x;\psi), g_\phi(M(x;\psi))\Big)\right]$$

   where $R(\cdot)$ is a reinforcement signal (e.g., derived from Group Relative Policy Optimization) that rewards effective memory recall and engram updates. The hyperparameter λ balances its influence.

Part 2: The Graph — Explicit Formulation

3. **Graph Regularization Term:**

$$\mu\,\mathcal{L}_{\text{graph}}\Big(M(x;\psi)\Big)$$

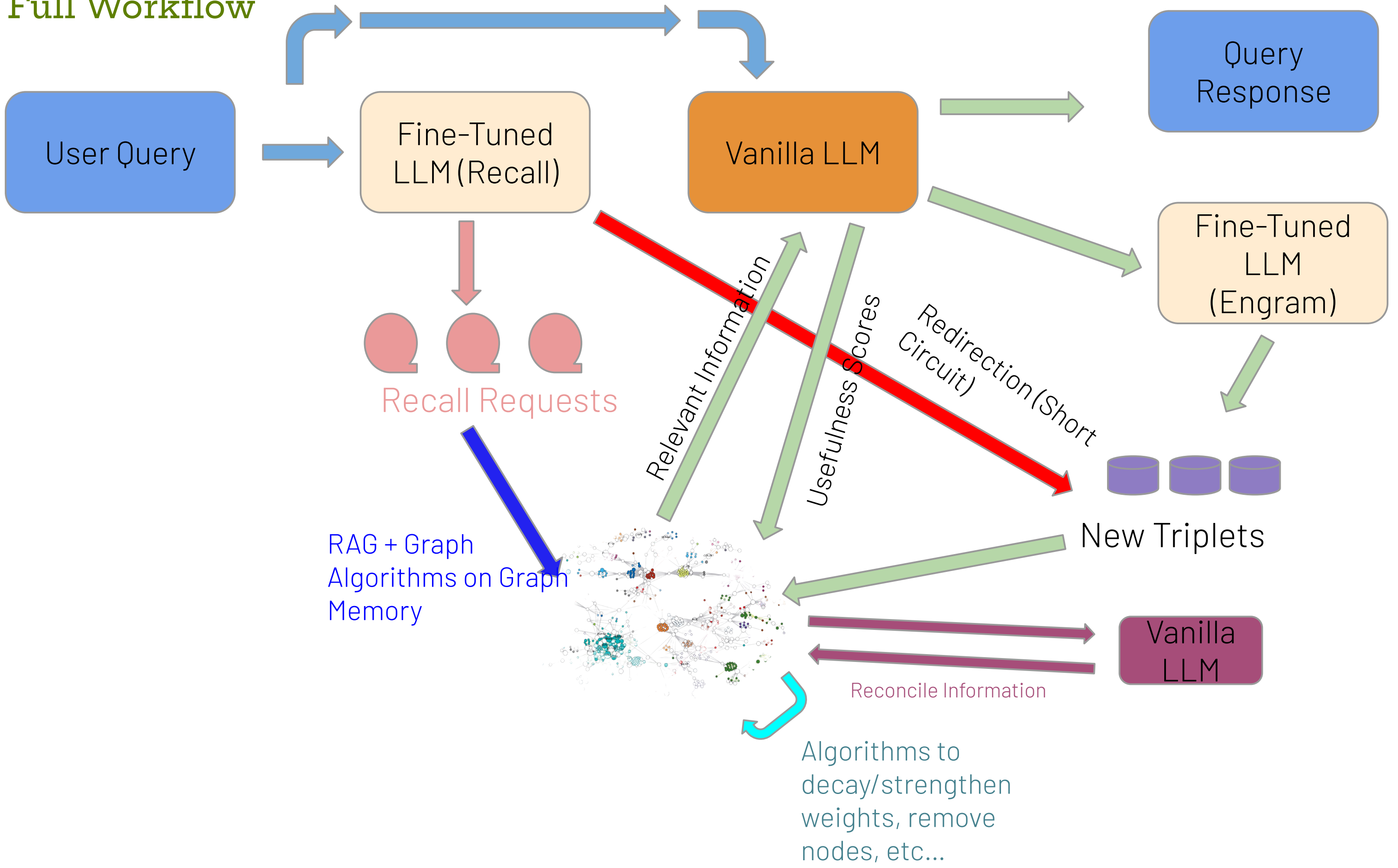
is a penalty term (which may include terms for edge density, conflict resolution, and pruning cost) to maintain an efficient, sparse, and interpretable Liquid Knowledge Graph. The hyperparameter μ regulates its strength.

4. **Regularization Term:**

$$\nu\,\mathcal{R}(\theta,\phi,\psi)$$

is a composite regularization term (including, for example, L_2 norms, memory capacity constraints, and complexity penalties) that ensures the overall system remains computationally feasible and stable. The hyperparameter ν controls its weight.

Full Workflow



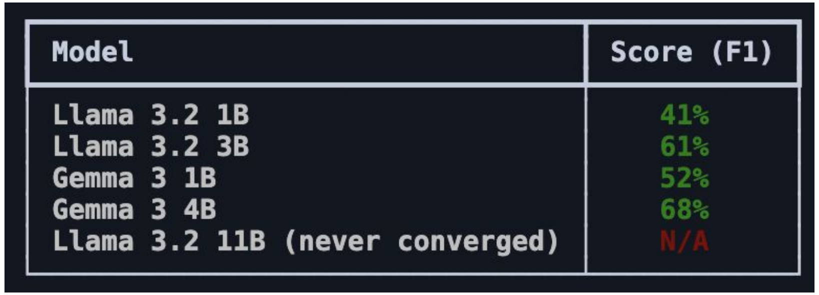
04

Training & Results

Part 1: The LLM

750 Steps of Training, 100 Steps of Evaluation

- Training data generated with GPT in batches of 5 with recursive multi-shot (we gave some initial samples, and then fed all generated samples)
- Score was F1:
  - **Precision**: Number of outputted engram/recall information that was also contained in the evaluation set (judged by another LLM)
  - **Recall**: Number of items in the evaluation set contained in the outputted engram/recall information(judged by another LLM)

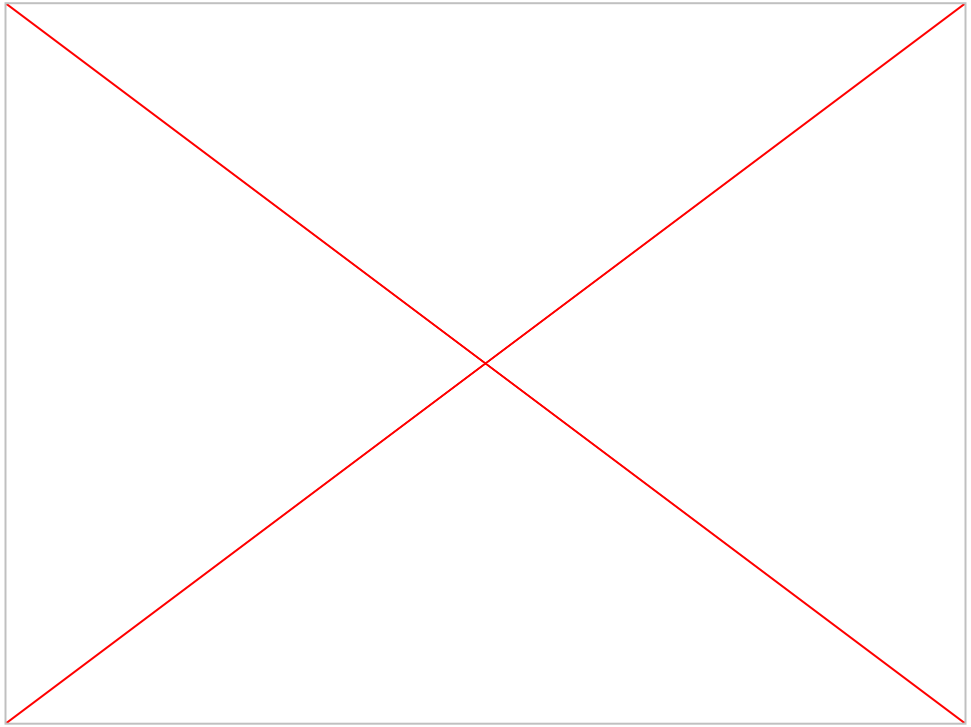


Model	Score (F1)
Llama 3.2 1B	41%
Llama 3.2 3B	61%
Gemma 3 1B	52%
Gemma 3 4B	68%
Llama 3.2 11B (never converged)	N/A

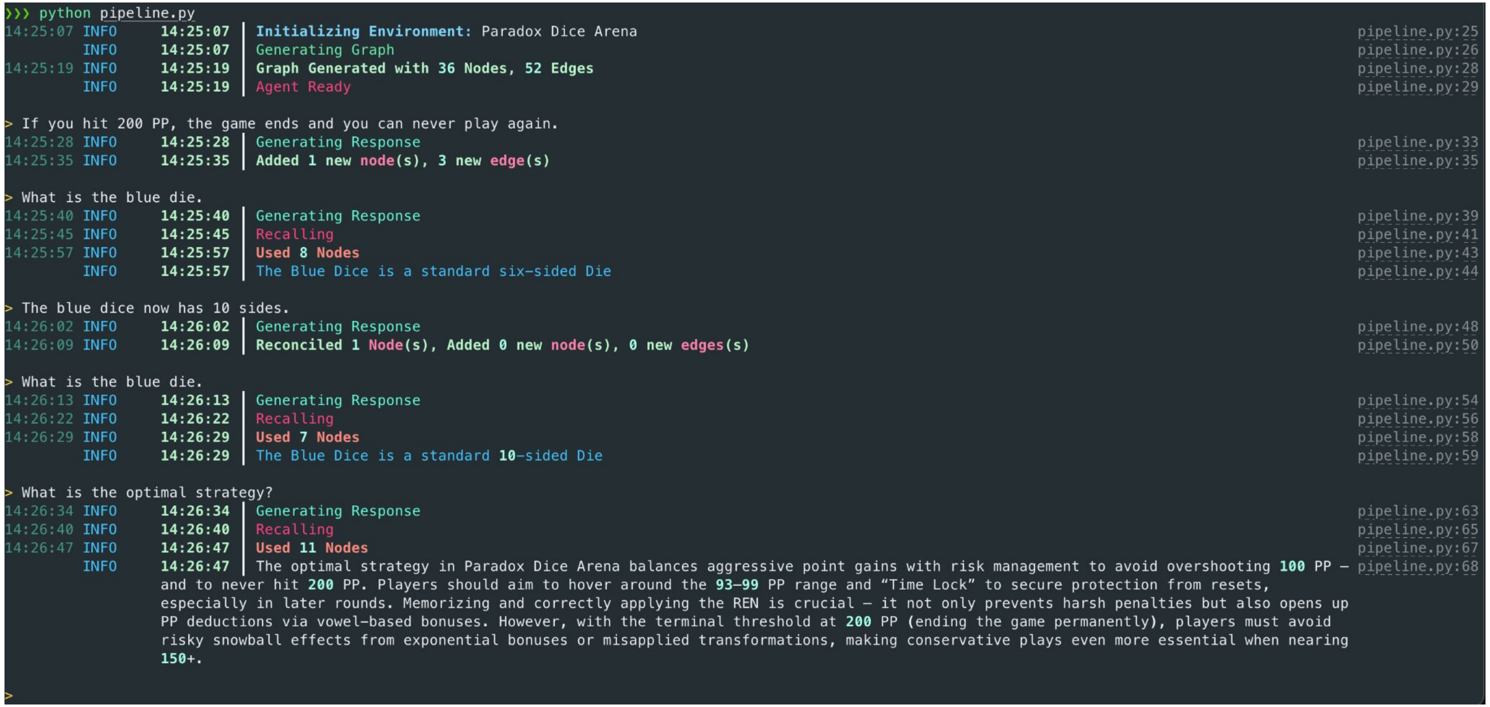
Part 1: The LLM

- **Trained on 5 models locally using GRPO:**
    - Llama 3.2 1B, Llama 3.2 3B,
    - Gemma 3 1B,
    - Gemma 3 4B,
    - Llama 3.2 11B (didn’t have time to converge).
    - Tried using reasoning models / CoT (Qwen SmallThinker 3B), but it started talking about syrup in the middle (likely too small of a model for proper reasoning with amount of input)
- **Implementation:**
    - Wrote custom implementation first with PyTorch
    - Used unsloth on local MacBook
        - Slightly faster.
        - Much less memory usage

Part 1: The LLM Demo



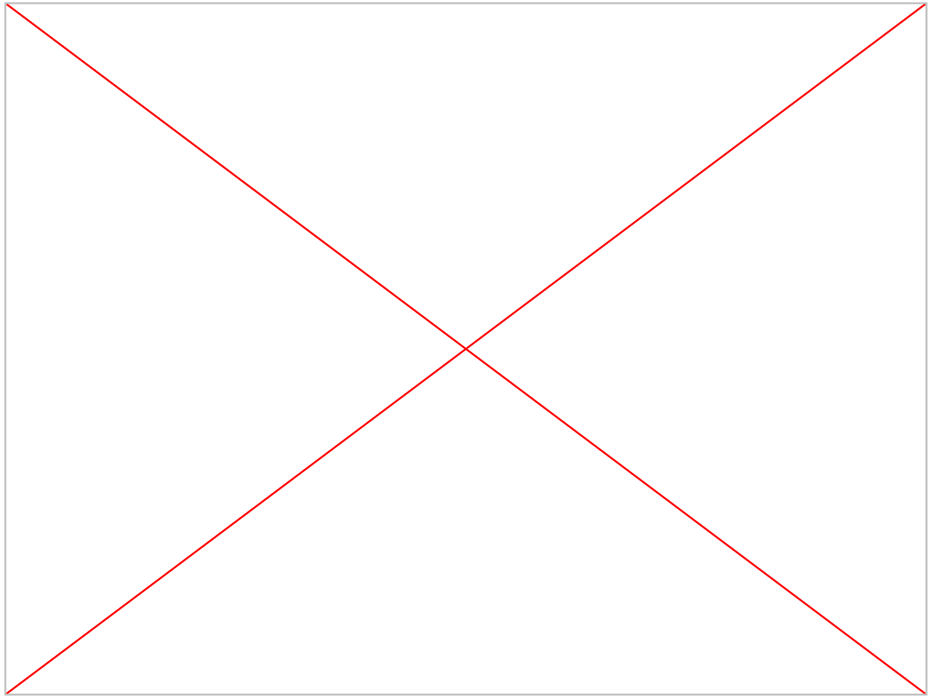
Part 1: The LLM Demo



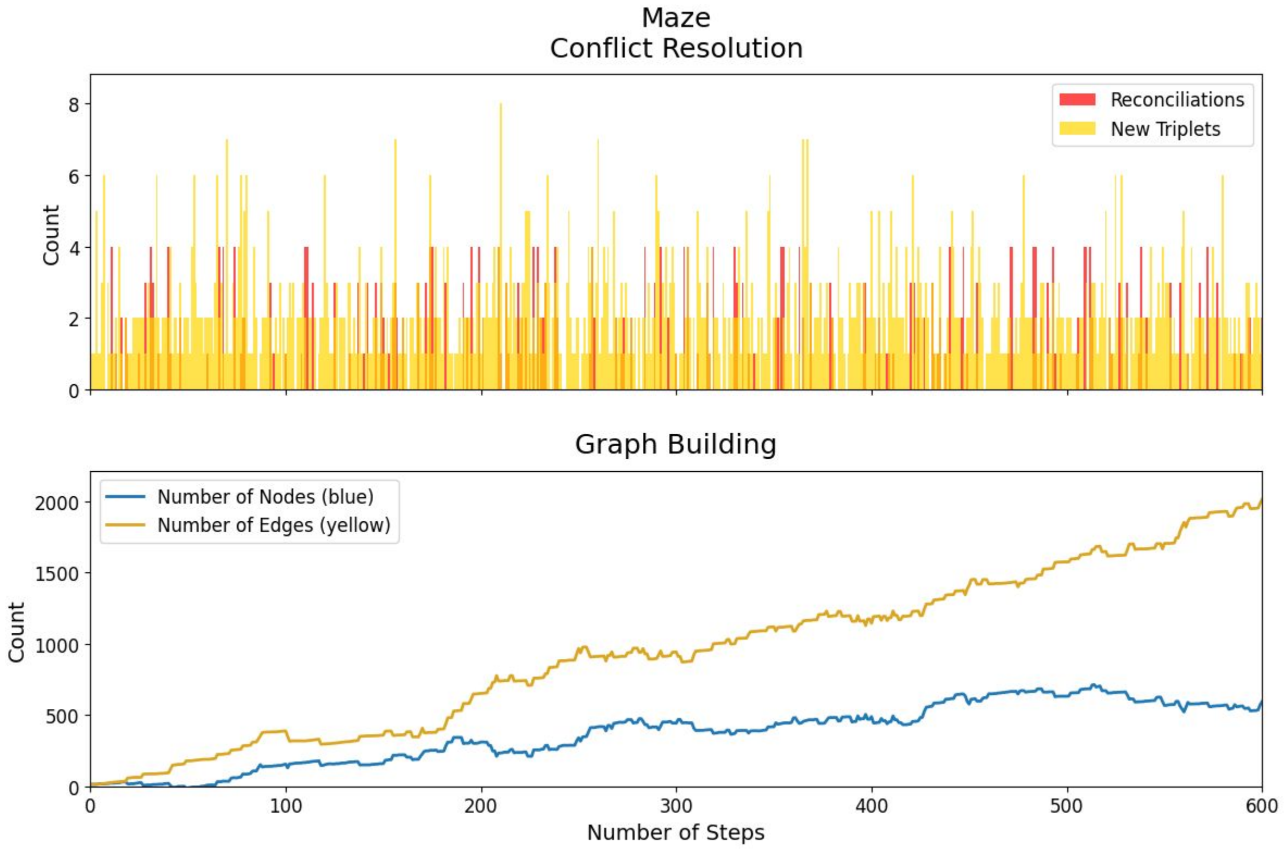
```
››› python pipeline.py
14:25:07 INFO     14:25:07 │ Initializing Environment: Paradox Dice Arena                                                                                         pipeline.py:25
         INFO     14:25:07 │ Generating Graph                                                                                                                   pipeline.py:26
14:25:19 INFO     14:25:19 │ Graph Generated with 36 Nodes, 52 Edges                                                                                            pipeline.py:28
         INFO     14:25:19 │ Agent Ready                                                                                                                        pipeline.py:29

> If you hit 200 PP, the game ends and you can never play again.
14:25:28 INFO     14:25:28 │ Generating Response                                                                                                                pipeline.py:33
14:25:35 INFO     14:25:35 │ Added 1 new node(s), 3 new edge(s)                                                                                                 pipeline.py:35

> What is the blue die.
14:25:40 INFO     14:25:40 │ Generating Response                                                                                                                pipeline.py:39
14:25:45 INFO     14:25:45 │ Recalling                                                                                                                          pipeline.py:41
14:25:57 INFO     14:25:57 │ Used 8 Nodes                                                                                                                       pipeline.py:43
         INFO     14:25:57 │ The Blue Dice is a standard six-sided Die                                                                                          pipeline.py:44

> The blue dice now has 10 sides.
14:26:02 INFO     14:26:02 │ Generating Response                                                                                                                pipeline.py:48
14:26:09 INFO     14:26:09 │ Reconciled 1 Node(s), Added 0 new node(s), 0 new edges(s)                                                                         pipeline.py:50

> What is the blue die.
14:26:13 INFO     14:26:13 │ Generating Response                                                                                                                pipeline.py:54
14:26:22 INFO     14:26:22 │ Recalling                                                                                                                          pipeline.py:56
14:26:29 INFO     14:26:29 │ Used 7 Nodes                                                                                                                       pipeline.py:58
         INFO     14:26:29 │ The Blue Dice is a standard 10-sided Die                                                                                           pipeline.py:59

> What is the optimal strategy?
14:26:34 INFO     14:26:34 │ Generating Response                                                                                                                pipeline.py:63
14:26:40 INFO     14:26:40 │ Recalling                                                                                                                          pipeline.py:65
14:26:47 INFO     14:26:47 │ Used 11 Nodes                                                                                                                      pipeline.py:67
         INFO     14:26:47 │ The optimal strategy in Paradox Dice Arena balances aggressive point gains with risk management to avoid overshooting 100 PP — pipeline.py:68
                  and to never hit 200 PP. Players should aim to hover around the 93–99 PP range and “Time Lock” to secure protection from resets,
                  especially in later rounds. Memorizing and correctly applying the REN is crucial — it not only prevents harsh penalties but also opens up
                  PP deductions via vowel-based bonuses. However, with the terminal threshold at 200 PP (ending the game permanently), players must avoid
                  risky snowball effects from exponential bonuses or misapplied transformations, making conservative plays even more essential when nearing
                  150+.

>
```

Part 2: The Graph Demo

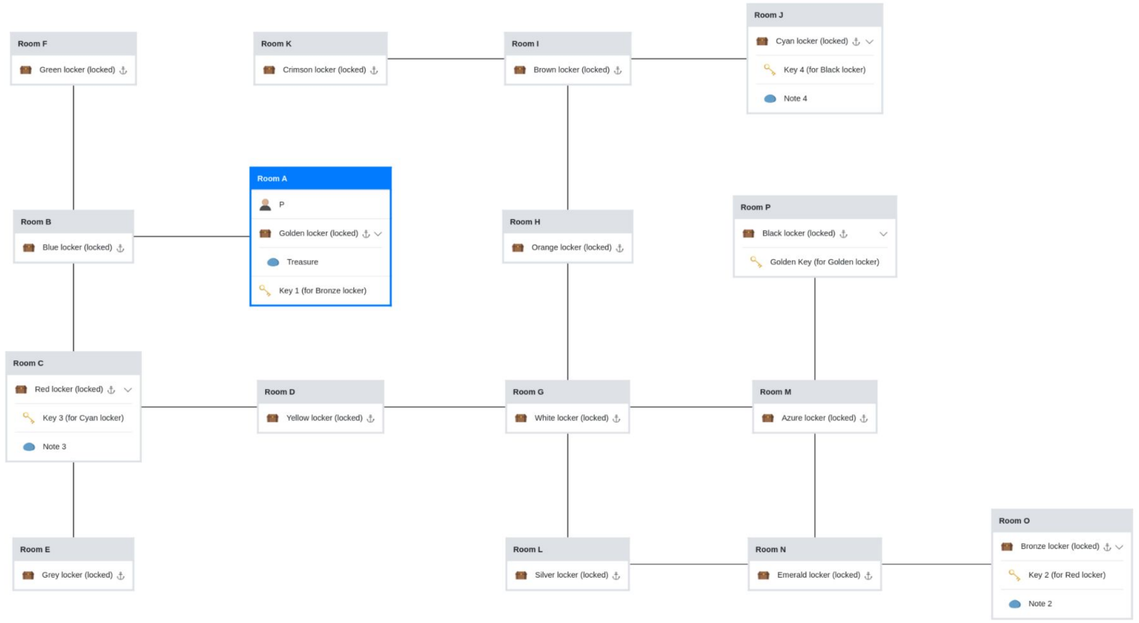
- **Tuned parameters**
    - Could've maybe used Online Gradient Descent in PyTorch to tune hyperamaraters if we messed with the equation a little.
        - Likely would be too slow (local models every evaluation)
        - Might've overfit on our example world models
    - Instead, just did a small grid search and chose the best fit (lowest error)
        - Didn't try too hard, lol wayyyyy too slow.



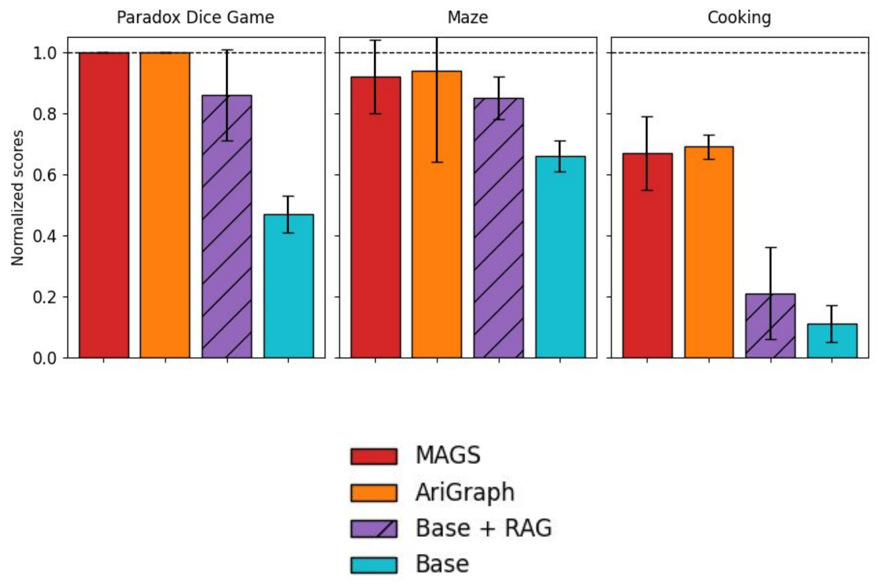
Part 2: The Graph



Treasure Hunt World Building (Medium Difficulty)



Overall Results



63%

Overall Accuracy (100Q Evaluation Set)

37%

Improvements in accuracy over Base Model (Gemma 4B)

13%

Improvements in accuracy over Base Model (Gemma 4B) + Static RAG (adding all information, querying, 10)

Live Demo!

05

Reflect & Next Steps

Reflection

- **Interpretation of Results**
  - Long-term learning and retention is possible! However, still a long way to go.
    - Not much better than in-context long-window learning (RoPE)
- **Limits**
  - Slow to test literally anything
    - Hard to parallelize runtime to even use more compute
  - Got banned on hugging face
- **Primary Challenge**
  - Getting a SLM to reliably output the engram and recall steps using RL. Most of our initial implementations didn't work, and in the end we didn't have >70% F1.
  - Outputting same verb (middle of triplet) every time
  - Still Doesn't really work (we just retry until it does)
- **Workflow**
  - Easier: GRPO and running a fine-tuning pipeline
  - Easier: Setting up the entire graph structure
  - Harder: Tuning hyperparameters for the graph system.
  - Harder: SOOO many edge cases

Reflection

- **Next Steps (If more time)**
  - Spending the time trying out larger (11B-16B) models
  - Trying more advanced Graph Algorithms.
  - Try to implement working memory and Chain-of-thought
  - Optimized Graph Hyperparamaters
- **Evolution & Expected vs Actual Progress**
  - Had a lot more ambitious goals revolving around some algorithms to optimize the graph
  - Using triplets instead of text blocks
  - Adapting fine-tuning dataset to include longer examples
- **AI Assistance**
  - Used Gemini / Claude / GPT for a lot of the implementation and finding research that tackled some roadblocks

>

MAGS