MAGS: Learning LLMS

By: Pulkith, Pragya, Stanley, Shailesh

Humans vs LLMs

- Lack of Agency
  - Figure, Tesla, etc…
- Lack of True Understanding, Reasoning, Novelty
  - ???
- **Lack of Memory, Learning, and World Model**
  - **Our Goal**

Motivation & Problem

Memory and Learning

- LLMs cannot **learn and remember**
  - Test-time emulation with context windows is not viable
    - Quadratic scaling
    - Attention decay and ephemeral context
    - Stateless session
  - Retrieval Augmented Generation (RAG) can add information
    - But rigid, one way static transfer of knowledge
    - Cannot reconcile new or conflicting information permanently

Motivation & Problem

SoTA / Literature

- 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 Attention loses fine details, and slow + 2M token max
- AriGraph
  - Adding Dynamic Graph-Based RAG
  - LLM has no representation of memory, only adds information (no updating+consolidating), no importance of different episodes, memory not connection across episodes

Motivation & Problem

Technical Approach

M	A	G	S
Memory	**Augmented**	**Graph**	**Scaling**
Episodic and Semantic Memory Blocks (RAG) along with Long-term vs Working Memory Supports separation of general knowledge, experience, and importance.	Compatible with any vanilla frozen LLM. Fine-tuned for engram+recall steps, so LLM can interact with memory. LLM can query knowledge before answering and store novel experiences and learnings.	Liquid Knowledge Graphs, that can add+remove nodes, change edge weights, consolidate and forget knowledge. Replicates human Hebbian Plasticity, Neurogenesis, and Synaptic Pruning.	Allows for context scaling limited by memory constraints rather than attention decay. Similar time inference. Chat History is part of Working memory, so agents are more modular.

Technical Approach

Structure

Knowledge broken down into Long-Term (important info), and Working Memory (chat history). Memory is moved from Working to Long-Term based on usage, ‘surprise’, or importance.

Recall

Anchor Nodes selected from query, BFS over decaying edge weights. SCCs consolidated. Can also query for connection between nodes.

Engram

Updates nodes and edge weights based on query importance. Can add or update information.

Validation

Custom game with nonsensical rules (so mode cannot reason). LLM can store newfound experiences, rule changes, surprising things, general game rules, etc...

Mathematical Formulation - GRPO

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.

Mathematical Formulation – MAGS

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

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).
- $g_\phi\big(M(x;\psi)\big)$ 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).

1. **Prediction Loss Term:**

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

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

2. **Memory Reward Term:**

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

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.

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.

Results

30%

Improvements in accuracy over Base Model (Gemma 4B)

Slow testing and iteration

PPO training code implementation

Prioritizing model separation and building out a router

Current State

- A basic Graph RAG prototype is implemented, allowing users to query, add, and remove nodes from the knowledge graph.
- The system lacks dynamic memory features; key mechanisms like edge decay/strengthening, anchored traversals, contradiction handling, and full RL-guided graph updates are not yet built.
- Initial testing shows slow response times (~1 minute locally) and unreliable autonomous memory updates (engram), with the LLM sometimes adding irrelevant or incorrect information.
- The underlying graph library is functional, but performance metrics for the integrated, dynamic system are not yet established or measured.

Next Steps

- Prioritize implementing the core dynamic graph algorithms, including decay, strengthening, and anchor-based memory traversals, to move beyond basic RAG.
- Secure essential GPU compute resources (Colab, Metal GPU) to enable the necessary LLM fine-tuning using techniques like LoRA/PPO/GRPO.
- Develop the specific synthetic datasets required for training the model to perform the recall and engram steps effectively.