Continual Learning in LLMs

Presented by: Abhimanyu Anand
Toronto LLM Meetup Group

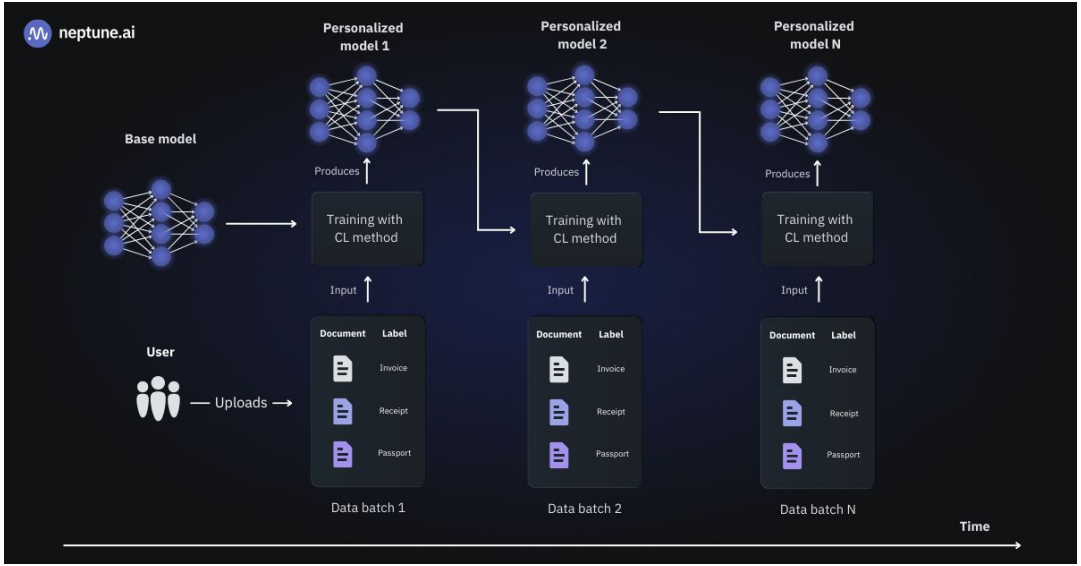
Overview

- **What is Continual Learning?**
- **Continual Learning for LLMs**
- **Break**
- **Applications & Future Directions**
- **Demo**

What is Continual Learning

What is Continual Learning?

- A paradigm for training models on a continuous stream of data, rather than a single, static dataset.
- Enables models to accumulate knowledge sequentially from new tasks, classes, or domains over time.
- The model might have limited access to past data



Stability vs. Plasticity

The Problem: Catastrophic Forgetting

- Learning new information causes the model to abruptly overwrite and forget previously learned knowledge.

The Goal: Balance Two Opposing Needs

- **Stability (Memory):** Retain and consolidate past knowledge.
- **Plasticity (Adaptation):** Quickly learn from new information.

The Ambition: To mimic how humans learn new skills without completely forgetting old ones.

Continual Learning Scenarios

Task-incremental learning: The aim is to incrementally learn a set of distinct tasks based on a given task-id.

$$h^* = \arg\min_h \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{T}_t} \left[\mathbb{1}_{h(\boldsymbol{x},t)\neq y}\right].$$

Domain-incremental learning: The context or input distribution varies over time, whilst the task remains constant (e.g. learning to drive in different weather conditions).

$$h^* = \arg\min_h \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_t} \left[\mathbb{1}_{h(\boldsymbol{x})\neq y}\right].$$

Class-incremental learning: The aim is to incrementally learn to discriminate between a growing number of objects or classes, where task identification is also required.

$$h^* = \arg\min_h \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{T}_t} \left[\mathbb{1}_{h(\boldsymbol{x})\neq(t,y)}\right].$$

Strategies for Continual Learning

Replay Based Methods: Utilise stored examples from past tasks to reinforce prior knowledge during training

$$\widehat{\mathcal{L}}_{\text{replay}}(h) \triangleq \underbrace{\sum_{i=1}^{t-1} \widehat{\mathcal{L}}_{M_i}(h)}_{\text{proxy for past domains}} + \underbrace{\widehat{\mathcal{L}}_{S_t}(h)}_{\text{current domain}} ,$$

Regularization-Based Methods: Mitigate catastrophic forgetting by adding constraints to the learning process, ensuring that important weights from previous tasks or domains are preserved.

$$\widehat{\mathcal{L}}_{\text{reg}}(h_{\boldsymbol{\theta}}) \triangleq \underbrace{\lambda \cdot \|\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1}\|_{\Sigma}}_{\text{proxy for past domains}} + \underbrace{\widehat{\mathcal{L}}_{S_t}(h_{\boldsymbol{\theta}})}_{\text{current domain}} ,$$

Architecture-Based Methods: Modify model's structure to accommodate new tasks while retaining previously learned knowledge.

Continual Learning Evaluation Metrics

- **Average Performance:** Assesses the ability of a model or algorithm to effectively learn from and adapt to a sequence of data streams or tasks over time

- **Forward Transfer Rate (FWT):** Assesses the impact of knowledge acquired from previous tasks on the initial ability to perform a new task, prior to any dedicated training for that new task

- **Backward Transfer Rate (BWT):** Measures catastrophic forgetting by comparing a model's performance on old tasks before and after learning new ones.

- **General Ability Delta (GAD):** Assesses the performance difference of an LLM on general tasks after training on sequential target tasks

Challenges in Continual Learning

- Catastrophic Forgetting
- Computation-efficient Continual Learning
- Continual Learning with Controllable Forgetting
- Training Stability
- Lack of research

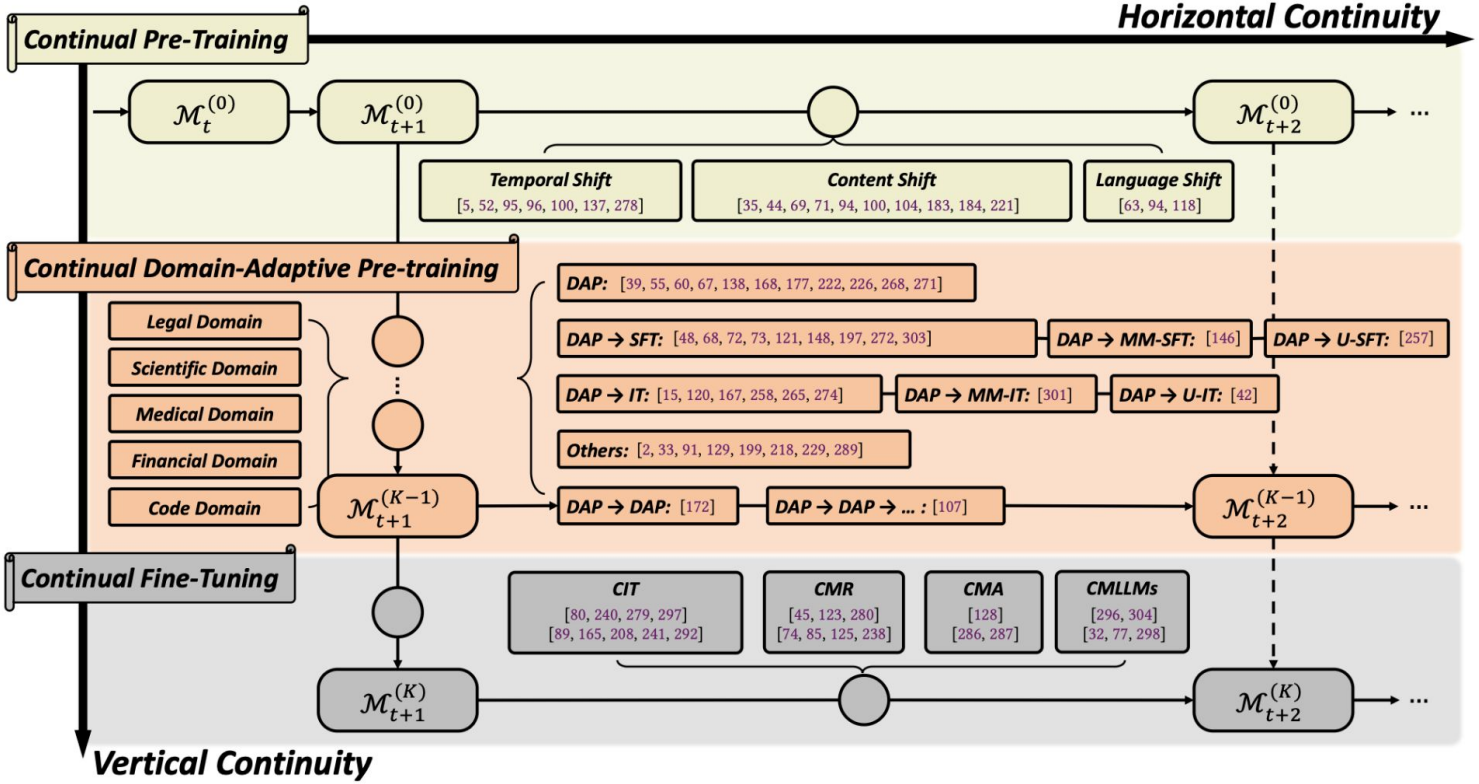
Continual Learning for LLMs

Is CL still necessary in the era of FMs?

- The objective of CL is to have a machine learning **model that can be adapted** quickly to shifts in data distribution or “tasks”, enabling it to **retain already acquired knowledge** and concepts and **reuse these representations** to facilitate **better learning across new tasks**.
- Challenges with FMs:
  - **Staleness**: FMs are snapshots, fixed at training.
  - **Homogenization & Bias**: General knowledge averages out domain-specific nuance.
  - **Distribution Shift & Task Heterogeneity**: Lack intrinsic adaptability to new environments.
  - **Long-Running Agent Tasks**: Inability to consolidate knowledge over time.
  - **High Computational Cost**

Information	RAG	Model Editing	Continual Learning
Fact	✓	✓	✓
Domain	✓	×	✓
Language	×	×	✓
Task	×	×	✓
Skills (Tool use)	×	×	✓
Values	×	×	✓
Preference	×	×	✓

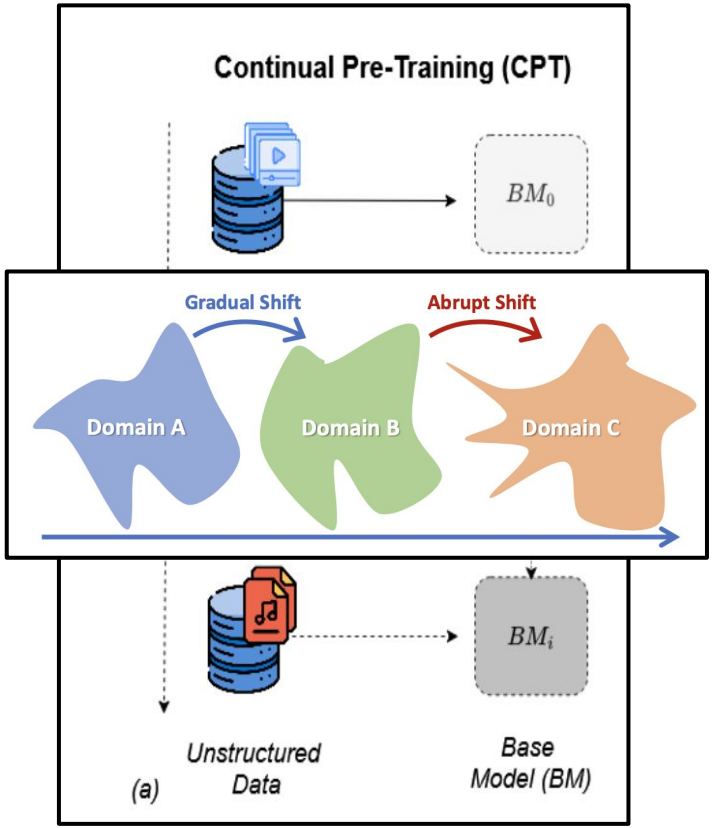
CL for LLMs



Ref -1

Introduction to Continual Pre-training (CPT)

- Process of incrementally updating the knowledge of FMs after their initial pre-training.
- **Goal:** Maintain foundational abilities while assimilating emerging information to extend operational lifespan.
- **Motivations**:
  - **Dynamic Knowledge Integration**
  - **Methodological Evolution**
  - **Mitigation & Management of Forgetting**
  - **Downstream Improvements**



Ref - 2

CPT Challenges

Core Challenges:

- **Balancing Efficiency vs. Model Drift:** Finding a computationally efficient update (subset of parameters/data) without degrading performance on original domains.
- **Catastrophic Forgetting:** The persistent problem of losing prior knowledge.
- **Reinforcement of Biases:** Continuously ingesting uncurated data can amplify existing biases or introduce new fairness issues.

CPT: Solutions

Promising Solutions:

- **Incremental Pre-training Strategies:** Reuse/initialize from previous weights (e.g., recyclable tuning).

- **Selective Memory & Rehearsal:** Use **latent replay** (compact feature vectors) or **selective sampling** (e.g., choosing the most novel or high-perplexity data for training).

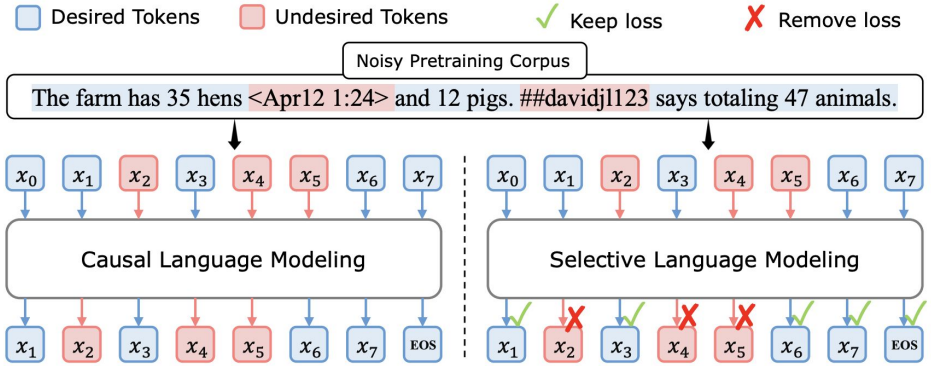


Figure 2: **Upper:** Even an extensively filtered pretraining corpus contains token-level noise. **Left:** Previous Causal Language Modeling (CLM) trains on all tokens. **Right:** Our proposed Selective Language Modeling (SLM) selectively applies loss on those useful and clean tokens.

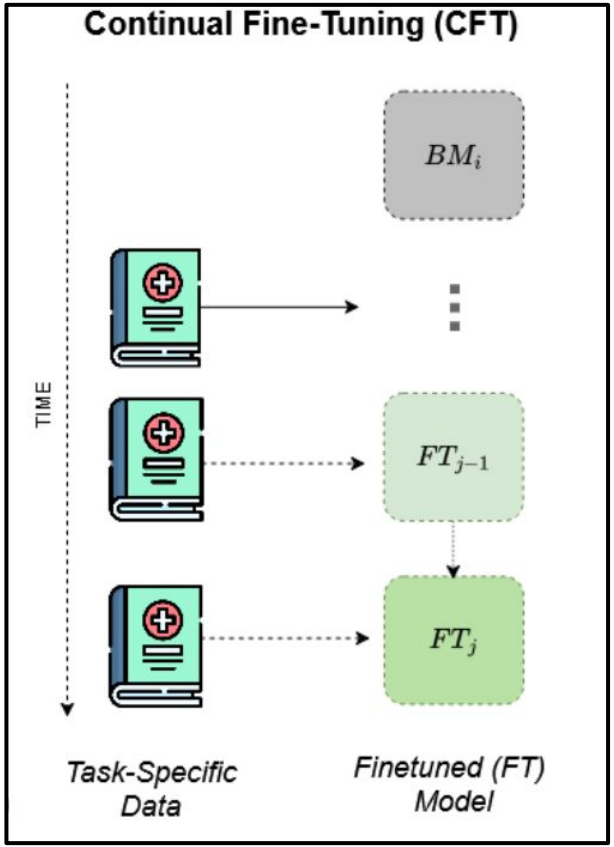
Ref - 3

Continual Fine-Tuning

- Applying a stream of **lightweight, task-specific updates** to a model *after* deployment.
- Allows the model to evolve alongside newly arriving, labeled data.

2. What CFT enables:

- **Instruction Tuning:** Improves the ability to follow textual instructions, bridging the gap between general and task-specific performance.
- **Model Refinement:** Rectifies model errors on specific inputs while preserving general performance (error correction).
- **Model Alignment:** Ensures outputs align with human values, ethics, and preferences (safety/bias correction).



Ref - 2

CFT: Motivation

- **Personalization:** Tailor responses for individual users or organizations.
- **Proprietary Data:** Adapt to private data while maintaining on-premise privacy compliance.
- **Quick Adaptation:** React quickly to domain drift without RAG latency or very long context windows.
- **Latency/Efficiency:** Fraction of the compute budget needed for full-scale updates

CFT: Challenges

- **Balancing Specificity vs. Generalization:** The stability-plasticity dilemma—optimize for a task without eroding general knowledge.
- **Catastrophic Forgetting:** The loss of prior task-specific knowledge.
- **Data Efficiency & Privacy:** Adapting to new tasks often means dealing with scarce, high-quality, or private domain-specific data.

CFT: Solutions

- **Parameter Efficient Fine-Tuning (PEFT):** Update only a small fraction of parameters (e.g., LoRA).
  - *PEFT works for TIL*
  - *Advanced:* LoRA-based CL (**DualLoRA**, or C-LoRA) uses orthogonal subspaces to mitigate forgetting.

- **Model Merging:** Combine multiple specialized models (learned over time) to preserve knowledge.
  - *Techniques:* **TIES-MERGING, DARE** (to handle parameter interference).

- **RL-based FT**

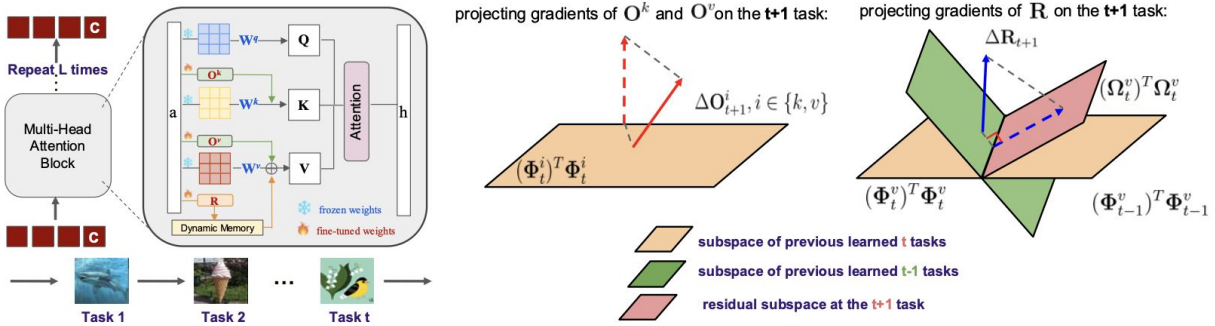


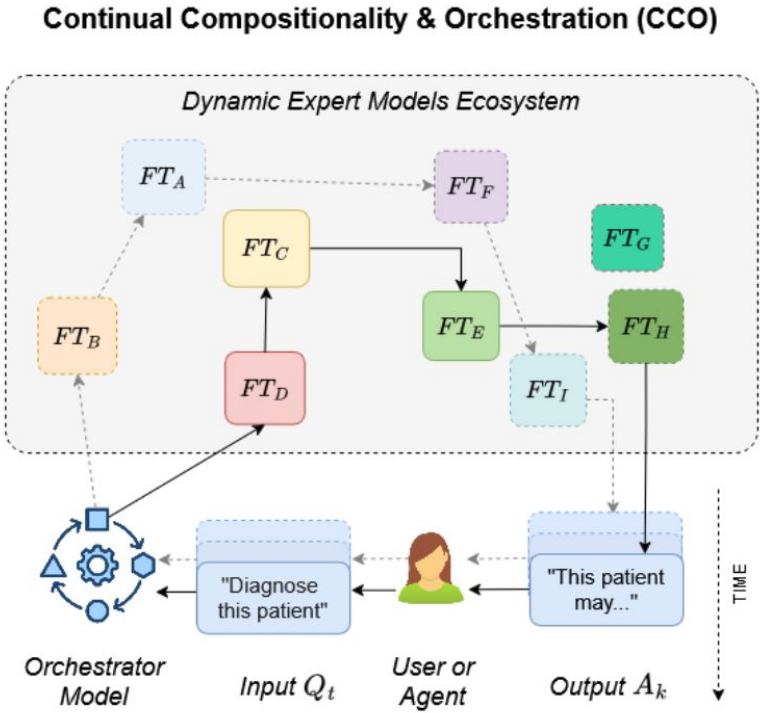
Figure 2. Illustration of our proposed DualLoRA paradigm (left) and design insights of orthogonal adapter and residual adapter (right), where the solid arrow denotes the original update and the dashed arrow denotes the projected update.

Ref - 4

Continual Compositionality & Orchestration

Dynamic integration of multiple AI agents over time to solve complex, **higher-level tasks**.

- **Motivation:**
  - **Shift from LLMs to Systems:** Monolithic models hit diminishing returns (GPT-4.5); complex tasks (e.g., ARC-AGI, BIG-bench) require abstraction and collaboration.
  - **High-Frequency Adaptation:** Enables sustainability and scalability by reusing specialized modules.
  - **Downstream Improvements:** Communication and modularity lead to superior performance and transparency.



Ref - 2

CCO: Challenges of Multi-Agent Systems and Solutions

Core Challenges:

- **Task Decomposition and Specialization:** Difficulty in breaking down complex tasks and ensuring each expert acquires unique skills (avoiding superficial specialization).
- **Role-based Collaboration and Interactions:** Optimizing role assignments and managing multi-round, dynamic coordination between agents (e.g., handling roles not seen in training).
- **Propagation of Errors:** Erroneous outputs (hallucinations, biases) from one agent can cascade, get amplified, and spread throughout the whole system.

Solutions:

- **Multi-Agent Fine-Tuning:** Training agents for cooperation and coordination (e.g., dynamic selection and composition).
- **Online Learning & Robust Protocols:** Developing communication protocols that accommodate heterogeneous components (vision, symbolic, LLMs) and decentralized knowledge sharing.

Break

Applications & Future Direction

Real World Applications: CPT

- **CPT:** Llama 3

3.4.1 Initial Pre-Training

We pre-train Llama 3 405B using AdamW with a peak learning rate of 8×10^{-5}, a linear warm up of 8,000 steps, and a cosine learning rate schedule decaying to 8×10^{-7} over 1,200,000 steps. We use a lower batch size early in training to improve training stability, and increase it subsequently to improve efficiency. Specifically, we use an initial batch size of 4M tokens and sequences of length 4,096, and double these values to a batch size of 8M sequences of 8,192 tokens after pre-training 252M tokens. We double the batch size again to 16M after pre-training on 2.87T tokens. We found this training recipe to be very stable: we observed few loss spikes and did not require interventions to correct for model training divergence.

Adjusting the data mix. We made a several adjustments to the pre-training data mix during training to improve model performance on particular downstream tasks. In particular, we increased the percentage of non-English data during pre-training to improve the multilingual performance of Llama 3. We also upsample mathematical data to improve the model's mathematical reasoning performance, we added more recent web data in the later stages of pre-training to advance the model's knowledge cut-off, and we downsampled subsets of the pre-training data that were later identified as being lower quality.

3.4.2 Long Context Pre-Training

In the final stages of pre-training, we train on long sequences to support context windows of up to 128K tokens. We do not train on long sequences earlier because the compute in self-attention layers grows quadratically in the sequence length. We increase the supported context length in increments, pre-training until the model has successfully adapted to the increased context length. We assess successful adaptation by measuring whether **(1)** model performance on short-context evaluations has recovered completely and **(2)** the model perfectly solves "needle in a haystack" tasks up to that length. In Llama 3 405B pre-training, we increased context length gradually in six stages, starting from the original 8K context window and ending in the final 128K context window. This long-context pre-training stage was performed using approximately 800B training tokens.

Ref - 5

Real World Applications: CPT

- **CPT/CFT:** LLM / MLLM based recommenders at companies like Netflix, LinkedIn, Spotify

1. **Incremental training :** Foundation models are trained on extensive datasets, including every member's history of plays and actions, making frequent retraining impractical. However, our catalog and member preferences continually evolve. Unlike large language models, which can be incrementally trained with stable token vocabularies, our recommendation models require new embeddings for new titles, necessitating expanded embedding layers and output components. To address this, we warm-start new models by reusing parameters from previous models and initializing new parameters for new titles. For example, new title embeddings can be initialized by adding slight random noise to existing average embeddings or by using a weighted combination of similar titles' embeddings based on metadata. This approach allows new titles to start with relevant embeddings, facilitating faster fine-tuning. In practice, the initialization method becomes less critical when more member interaction data is used for fine-tuning.

Ref - 6

Real World Applications: CPT

just member interaction data. Thus, our foundation model combines both learnable item id embeddings and learnable embeddings from metadata. The following diagram demonstrates this idea.

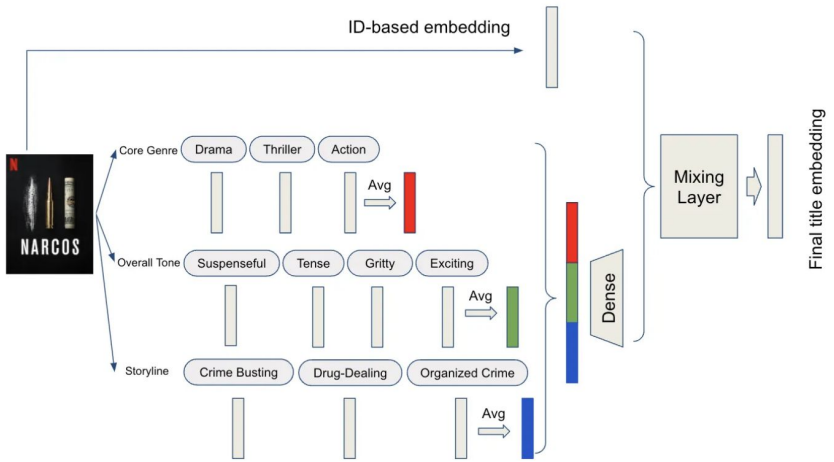


Figure 2. Titles are associated with various metadata, such as genres, storylines, and tones. Each type of metadata could be represented by averaging its respective embeddings, which are then concatenated to form the overall metadata-based embedding for the title.

To create the final title embedding, we combine this metadata-based embedding with a fully-learnable ID-based embedding using a mixing layer.

Ref - 6

Real World Applications: CFT

Limitations of traditional approaches

Traditional recommender systems, often trained on fixed historical datasets, struggle to adapt to evolving user preferences or a constantly changing catalog. Periodic retraining can help, but it is slow and coarse. Many follow a pipeline approach, with multiple sequential components, making it difficult to assign credit or blame when recommendations succeed or fail.

Reinforcement learning (RL) offers a more adaptive alternative by learning directly from interaction data. However, RL systems can regress when the underlying base model is updated, since learned preferences do not always transfer cleanly. Adding to this challenge, the catalog of items is itself constantly evolving, so the system must continually adapt to shifting preferences and choices.

Real World Applications: CFT

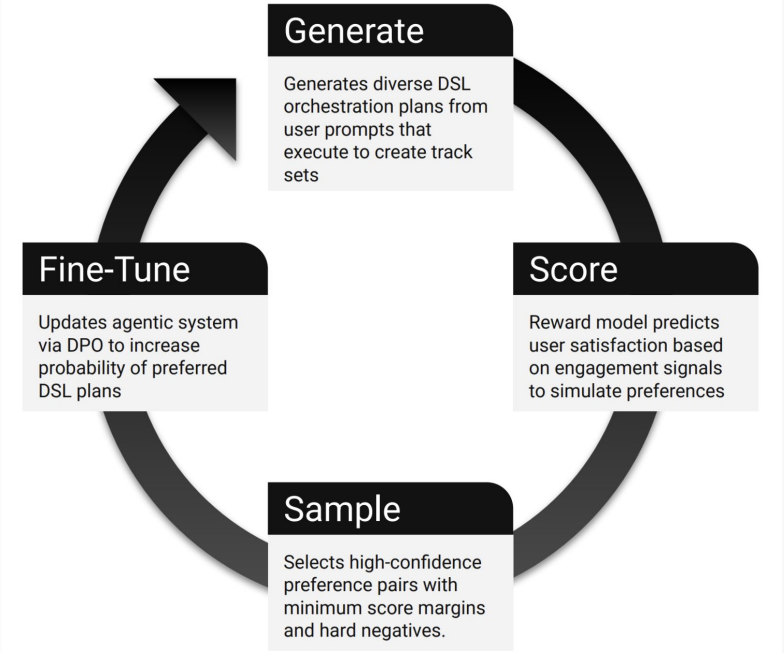
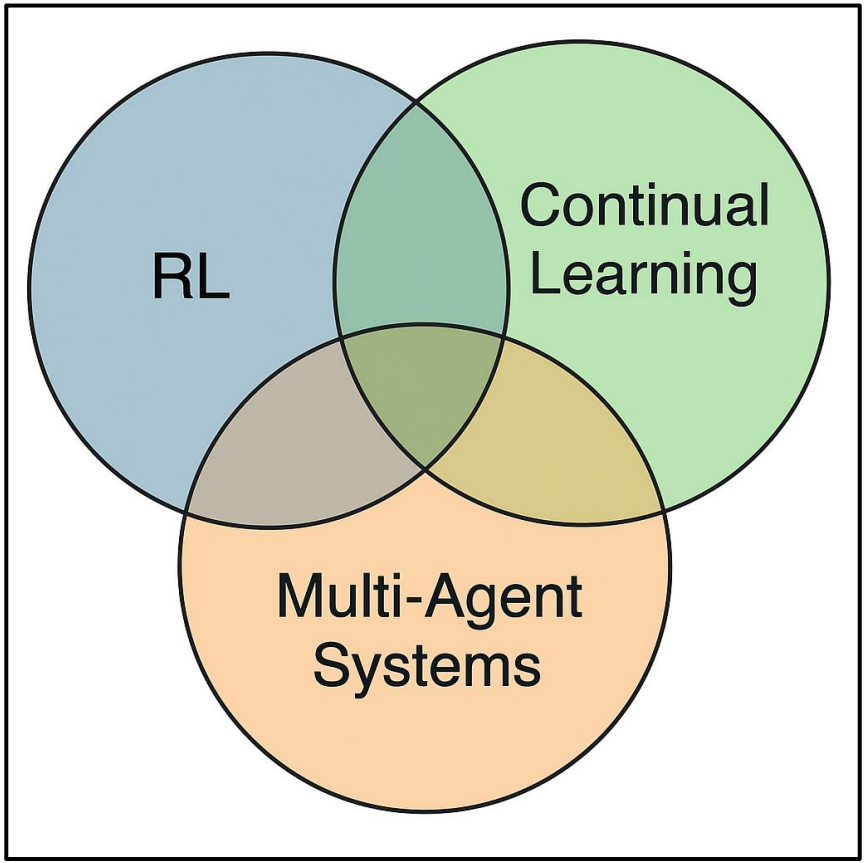


Figure 1: *Preference tuning flywheel showing the four-stage iterative process: Generate diverse DSL orchestration plans (the tool calls for playlist synthesis), Score candidates with reward model, Select high-confidence preference pairs, and Fine-Tune the agentic system using DPO. This cycle continuously improves preference alignment from user interaction data.*

Ref - 7

Future Directions: RL x L-MAS x CL



Future Directions: CL x Agents

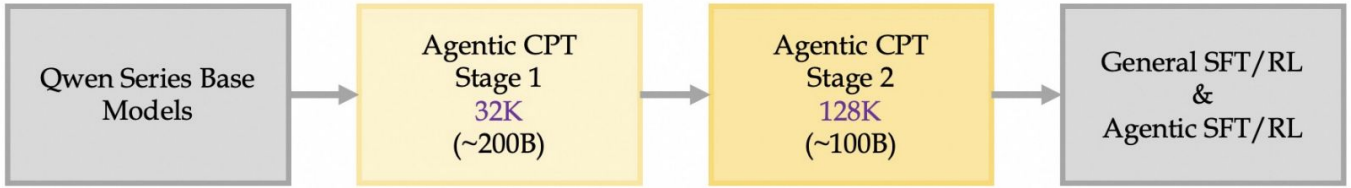


Figure 2: Agentic Training Pipeline.

- Agentic CPT Stage 1: We process approximately 200B tokens of agent data and knowledge reasoning corpora with 32K context length, following the same next-token prediction paradigm as Eq. 1. This stage enables the preliminary acquisition of agentic behaviors including tool invocation patterns and multi-step reasoning chains.
- Agentic CPT Stage 2: We further refine these capabilities using 100B tokens of carefully curated, high-quality agent data with extended 128K context windows, allowing the LLM to develop a sophisticated understanding of complex action spaces and long-horizon planning strategies.

Future Directions: CL x RL

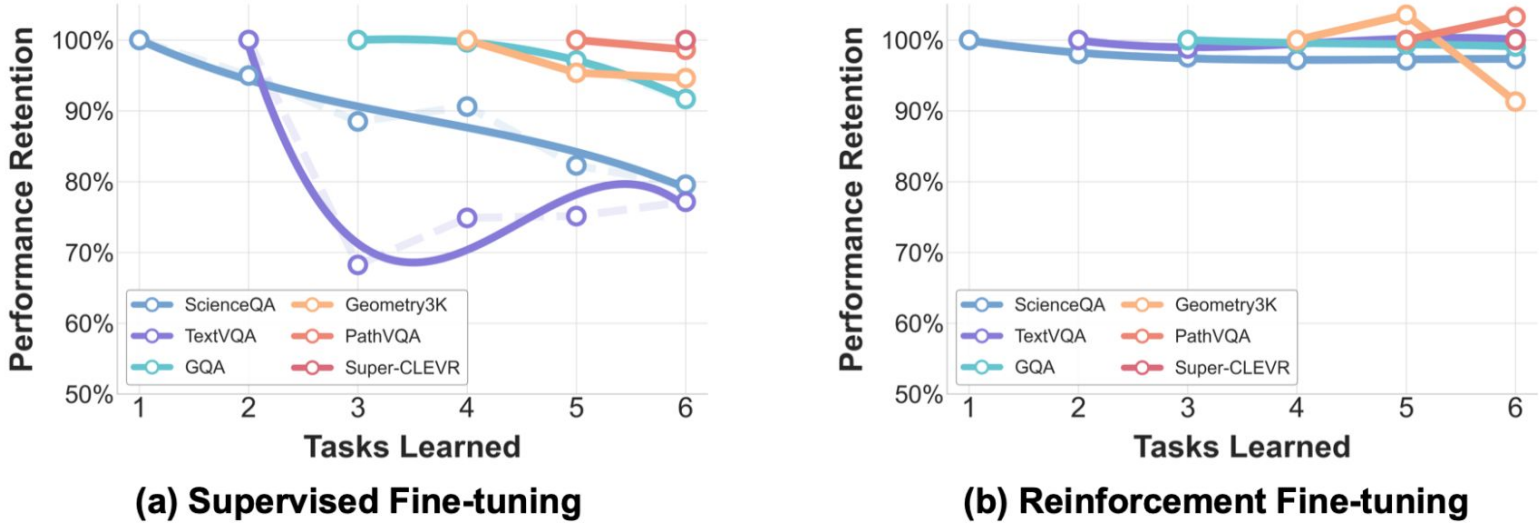


Figure 1: Comparison of performance retention between SFT and RFT in continual post-training. We plot the performance relative to the initial training performance on each task as learning progresses through a sequence of tasks. **(a)** SFT exhibits catastrophic forgetting; **(b)** RFT maintains performance on previously learned tasks.

Future Directions: RL as Enabler

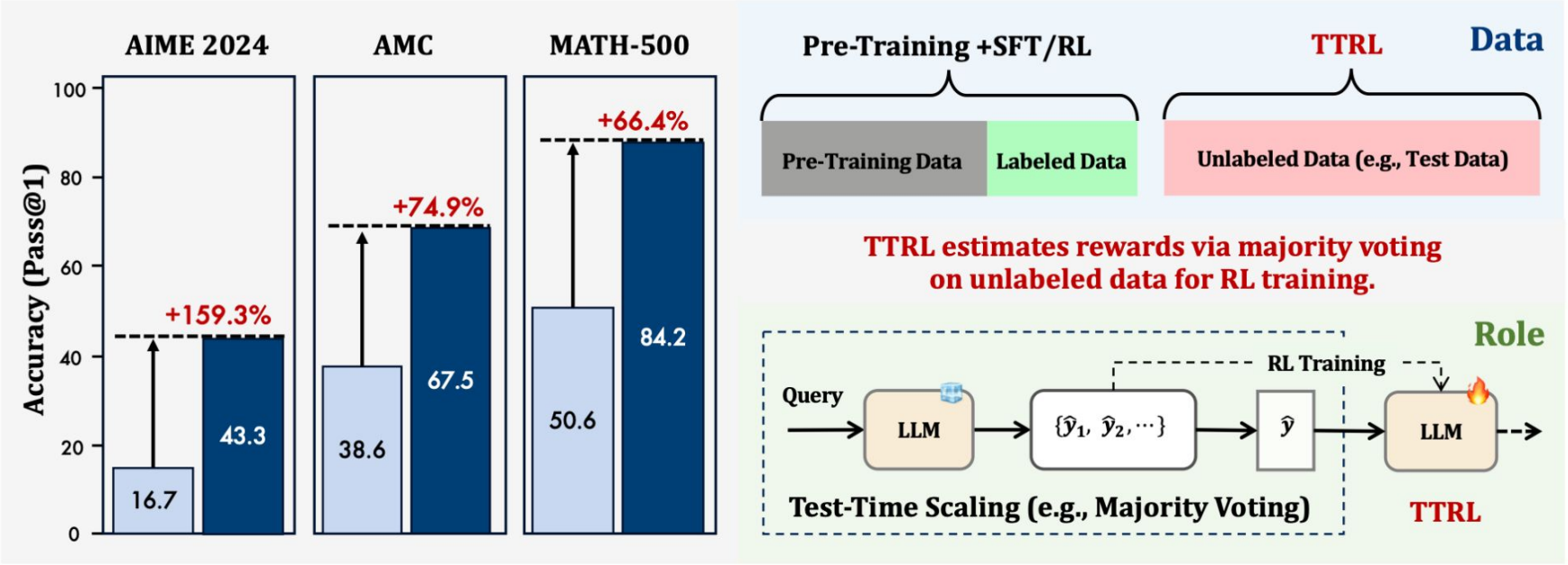


Figure 1: Performance and Position of **TTRL**.

Future Directions: RL as Enabler

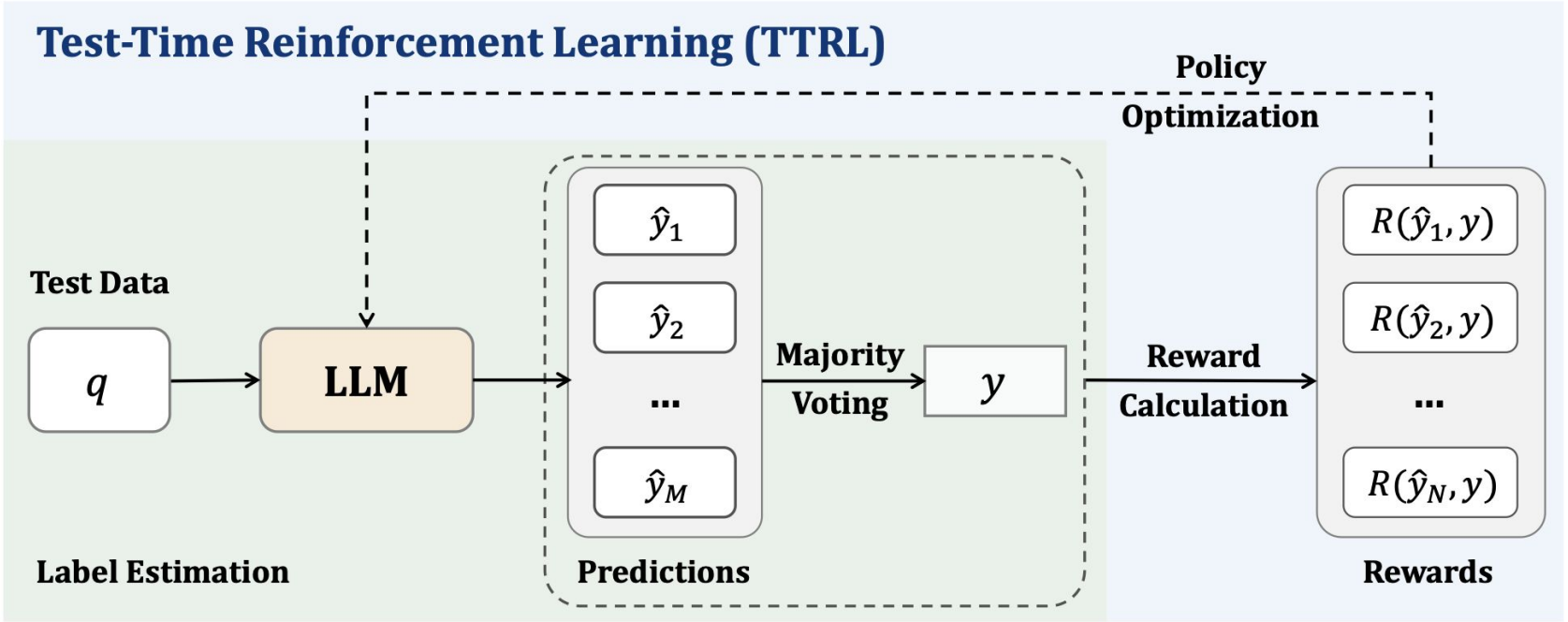


Figure 2: **TTRL** combines both Test-Time Scaling (TTS) and Test-Time Training (TTT).

Future Directions: RL x Agents

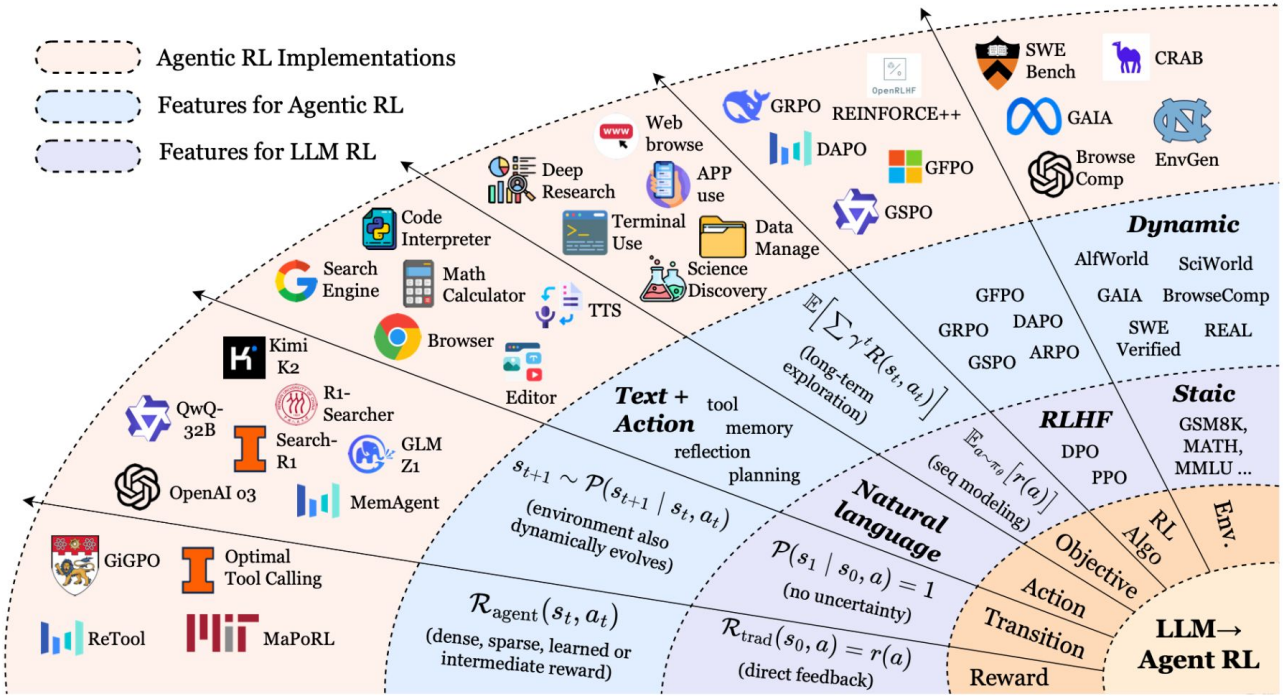


Figure 2: Paradigm shift from LLM-RL to agentic RL.

Future Directions: RL x Agents

Table 8: A summary of reinforcement learning and evolution paradigms in LLM-based Multi-Agent Systems. "Dynamic" denotes whether the multi-agent system is task-dynamic, *i.e.*, processes different task queries with different configurations (agent count, topologies, reasoning depth, prompts, *etc*). "Train" denotes whether the method involves training the LLM backbone of agents.

Method	Dynamic	Train	RL Algorithm	Resource Link
RL-Free Multi-Agent Systems (not exhaustive)				
CAMEL [421]	✗	✗	-	GitHub HuggingFace
MetaGPT [268]	✗	✗	-	GitHub
MAD [423]	✗	✗	-	GitHub
MoA [422]	✗	✗	-	GitHub
AFlow [429]	✗	✗	-	GitHub
RL-Based Multi-Agent Training				
GPTSwarm [425]	✗	✗	policy gradient	GitHub Website
MaAS [431]	✔	✗	policy gradient	GitHub
G-Designer [432]	✔	✗	policy gradient	GitHub
MALT [154]	✗	✔	DPO	-
MARFT [433]	✗	✔	MARFT	GitHub
ACC-Collab [146]	✗	✔	DPO	-
MAPoRL [434]	✔	✔	PPO	GitHub
MLPO [435]	✔	✔	MLPO	-
ReMA [436]	✔	✔	MAMRP	GitHub
FlowReasoner [437]	✔	✔	GRPO	GitHub
CURE [275]	✗	✔	rule-based RL	GitHub HuggingFace
MMedAgent-RL [438]	✗	✔	GRPO	-
Chain-of-Agents [439]	✔	✔	DAPO	GitHub HuggingFace
RLCCF [440]	✗	✔	GRPO	-
MAGRPO [441]	✗	✔	MAGRPO	-

Future Directions: L-MAS

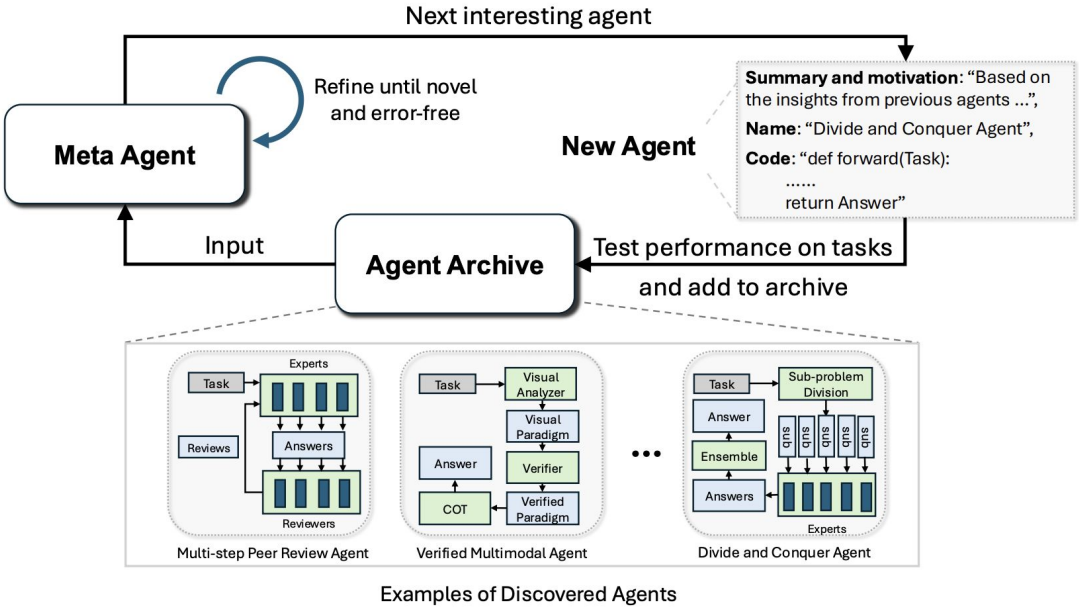


Figure 1: **Overview of the proposed algorithm Meta Agent Search and examples of discovered agents.** In our algorithm, we instruct the "meta" agent to iteratively program new agents, test their performance on tasks, add them to an archive of discovered agents, and use this archive to inform the meta agent in subsequent iterations. We show three example agents across our runs, with all names generated by the meta agent. The detailed code of example agents can be found in Appendix G.

Ref - 12

Future Directions: L-MAS

2 AUTOMATED DESIGN OF AGENTIC SYSTEMS (ADAS)

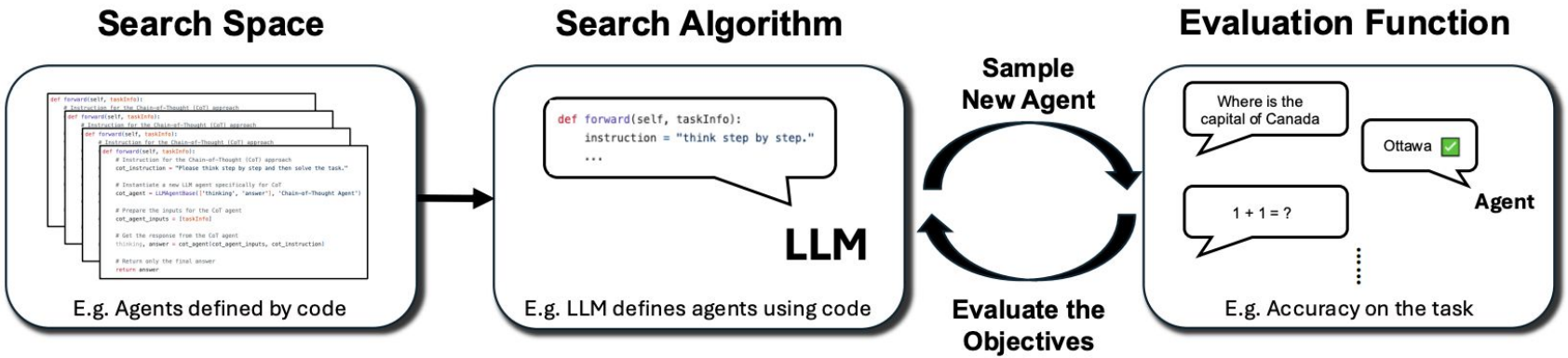


Figure 2: **The three key components of Automated Design of Agentic Systems (ADAS).** The search space determines which agentic systems can be represented in ADAS. The search algorithm specifies how the ADAS method explores the search space. The evaluation function defines how to evaluate a candidate agent on target objectives such as performance.

Future Directions: L-MAS

The Aime framework consists of four core components that work in concert to enable dynamic multi-agent collaboration.

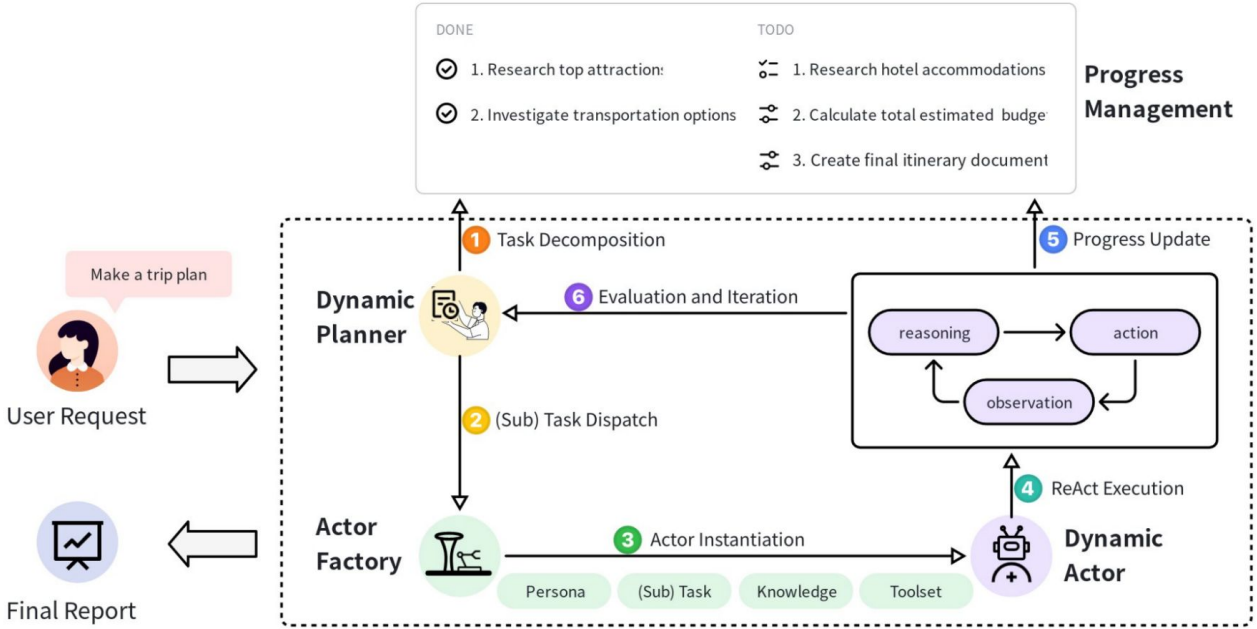


Figure 1: The workflow of Aime framework.

Ref - 13

Future Directions: RL x L-MAS x CCO

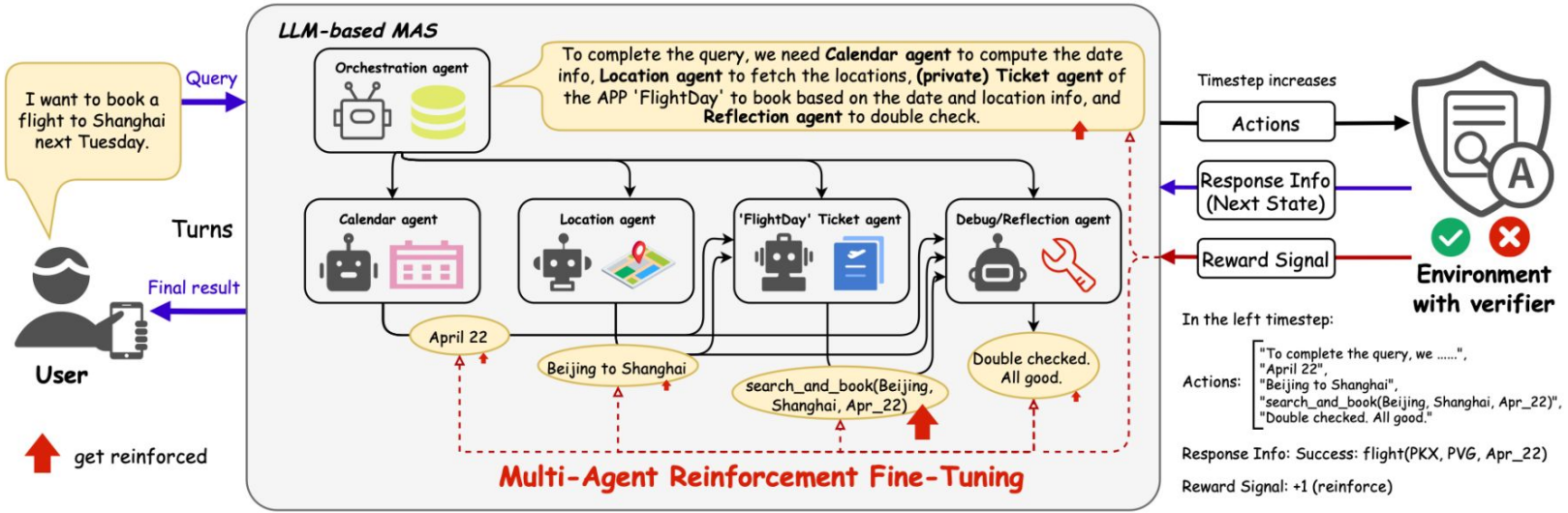


Figure 1: Illustration of MARFT in real-world agentic problem-solving scenarios.

CCO: Challenges of Multi-Agent Systems and Solutions

- **Continual Learning (CL)** is the shift from static models to **Adaptive, Resilient AI**.
- Goal: Learn new tasks/data streams without **Catastrophic Forgetting** (CF).

CL Techniques Span the LLM Lifecycle:

- **CPT (Continual Pre-Training):** Long-term knowledge update (e.g., Tongyi DeepResearch's Agentic CPT).
- **CFT (Continual Fine-Tuning):** High-frequency, task-specific adaptation (e.g., Spotify’s Preference Tuning Flywheel using DPO).

The Power of RL in CL:

- **RFT (Reinforcement Fine-Tuning):** Inherently **mitigates Catastrophic Forgetting** and **preserves general capabilities** better than SFT.

The Frontier: CL × RL × Agents (CCO)

- **TTRL (Test-Time Reinforcement Learning):** Enables **unsupervised adaptation** and **self-improvement** during inference, dramatically boosting performance without new labels.
- **MARFT (Multi-Agent Reinforcement Fine-Tuning):** The ultimate system for **Continual Compositionality & Orchestration (CCO)**, allowing complex agent teams to collaboratively learn and adapt to dynamic, multifaceted problems.

Demo

References

1. https://arxiv.org/pdf/2404.16789
2. https://arxiv.org/pdf/2506.03320 *
3. https://arxiv.org/pdf/2404.07965
4. https://arxiv.org/pdf/2411.00623
5. https://arxiv.org/pdf/2407.21783
6. https://netflixtechblog.com/foundation-model-for-personalized-recommendation-1a0bd8e02d39
7. https://research.atspotify.com/2025/9/personalizing-agentic-ai-to-users-musical-tastes-with-scalable-preference-optimization
8. https://arxiv.org/pdf/2509.13310, https://tongyi-agent.github.io/blog/introducing-tongyi-deep-research/ *
9. https://arxiv.org/pdf/2507.05386v2 *

References

10. https://arxiv.org/pdf/2504.16084 *

11. https://arxiv.org/pdf/2509.02547v1 *

12. https://arxiv.org/pdf/2408.08435

13. https://arxiv.org/pdf/2507.11988v2

14. https://arxiv.org/pdf/2504.16129 *

Thank you

Abhimanyu Anand