FEDERATION OVER TEXT

Dixi Yao, Tahseen Rabbani, Tian Li
University of Chicago {dixi,trabbani,litian}@uchicago.edu

ABSTRACT

LLM-powered agents often reason from scratch given a new problem instance and lack effective mechanisms to transfer learned skills to other agents. We propose the first federated-like framework, *federation over text* (FoT), where multiple agents solving different tasks collectively learn a library of metacognitive insights by federating local reasoning processes iteratively. Instead of federation over gradients (i.e., distributed training), FoT operates at the **semantic level** without any gradient optimization. Each agent does local thinking and reflection on their specific tasks and shares reasoning traces with a server, which aggregates them into a cross-task (and cross-domain) insight library that agents can leverage to improve performance on new reasoning tasks. Experiments show that FoT improves reasoning ability and efficiency on a wide range of applications including mathematical problem-solving, scientific question-answering, coding, and research insight discovery. For math problems, we achieve up to **36%** improvement in accuracy and **28%** reduction in generated tokens across benchmarks.

1 INTRODUCTION

LLM-powered agents are increasingly deployed for complex, multi-step tasks (Guo et al., 2025) . However, they can suffer from two major limitations. First, reasoning can be inefficient: agents often reason from scratch, repeatedly solving similar problems, which can be computationally expensive and slow. Second, reasoning typically occurs in an isolated manner, where agents simply transmit user prompts or final outputs (e.g., Ke et al., 2025). One agent's hard-won skills or reasoning procedures for solving a problem can be lost. It remains unclear how to easily transfer learned skills (thinking procedures) to other agents or domains.

How agents can interact at a higher metacognitive level to create reusable insights (Didolkar et al., 2025; Tran et al., 2025) that generalize to new agents or domains remains underexplored. At the same time, traditional federated or distributed learning schemes (McMahan et al., 2017) may not be suitable for learning generalizable insights from agents' reasoning traces. Averaging the gradients or weights of entire LLMs is computationally prohibitive and, more importantly, is not designed to explicitly share *abstract, high-level reasoning strategies*. We propose *federation over text* (FoT), the first federated-like framework for multi-agent collaboration that shares learned skills (i.e., reasoning processes) rather than model weights or prompts. FoT enables a set of agents to collaboratively build a shared, evolving library of reusable *insights* from metacognitive reasoning traces without sending local problem instances. The insight library is a high-level aggregation of the common principles and ideas of the agent tasks. For example, agents working on text modeling and agents exploring admixture inference in population genetics can share the same underlying principle and techniques.

Unlike traditional federated learning which averages gradients or model updates in a high-dimensional, uninterpretable space, FoT operates at the **semantic** level. Agents transmit concise metacognitive summaries of their problem-solving processes to a central (physical or virtual) server. The server, acting as an aggregator, identifies and distills recurring reasoning fragments from these summaries into new, explicit insights. This allows agents to create a library of generalizable insights that improves the reasoning capability and efficiency of all agents in the network. We note that FoT is a *general* framework that allows using any local agentic models coupled with any local thinking and self-reflection approaches, as long as the reasoning traces are shared at each iteration.

Interestingly, we highlight that our FoT approach is analogous to the classic federated/distributed learning setting. In particular, in Table 1, we show that these two regimes can be interpreted within

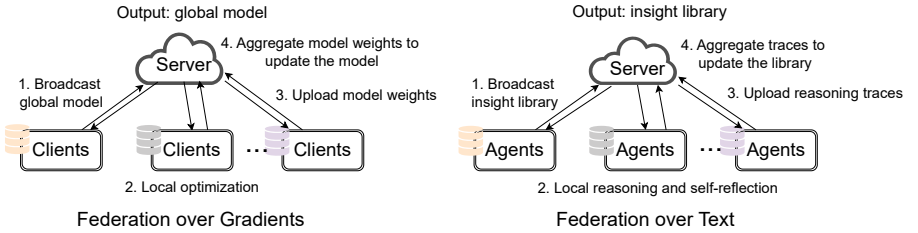

Figure 1: The workflow of federation over (accumulated) gradients and federation over text.

the same conceptual framework, where the core algorithmic components can be mapped to each other. By making these connections and drawing inspiration from federated optimization, this enables a range of potential design choices within FoT, regarding how to evaluate such a multi-agent iterative reasoning framework, how to personalize the services, and how to perform better joint reasoning. We discuss these implications and some future directions in more detail in Section 2.1.

In this work, we present preliminary results applying FoT to challenging mathematical problem solving and machine learning insight discovery (Section 3). For instance, we show that FoT can improve accuracy by **36%** on the AIME24 benchmark compared with directly querying the LLM model while decreasing the number of generated tokens by **28%** on average. In the second application, we demonstrate that our insight library generated by learning from old machine learning papers can cover the core technical contribution of over **70%** of subsequent machine learning papers.

2 Federation over Text

In Figure 1, we present the overall workflow of FoT, comparing it with federation over gradients (e.g., FedAvg (McMahan et al., 2017)). Federated learning aims to train a global model while keeping all data on local devices, whereas FoT aims to build an *insight library* that helps both existing and incoming agents better resolve various tasks, where the insight library is a high-level aggregation of common principles and ideas across heterogeneous agent tasks. As shown in Figure 1, federation over gradients first broadcasts an initial global model to each client, and each client trains a local model. Clients then upload model weights to the server, which aggregates them to update the global model. This process repeats iteratively. Similarly, FoT starts with an empty library, and each agent conducts local reasoning via any model. Reasoning traces are then uploaded to the server, which aggregates them to generate and merge new insights into the library. The library is next distributed in the following round for further improvement. The pseudo algorithm is in Appendix C.

In the workflow, several (or potentially many) decentralized agents each solve a specific problem or task. These tasks can span different disciplines such as mathematical reasoning, coding, and scientific question-answering. Given a problem, an agent first generates a solution, then reflects on whether the reasoning process contains reusable techniques and what key skills can be extracted. Finally, the agent produces a *reasoning trace* for each problem based on the solution and reflection, specifying when and how to apply the extracted skills through step-by-step instructions. Only traces are shared with the server; all answers, contents of reasoning process generated in the solution phase, and reflections remain local. If an insight library is available, agents use it as reference.

After collecting traces, the server first clusters reasoning traces sharing similar techniques, then uncovers connections among traces that solve the same problem using different techniques or can be combined for joint use. The server then updates the current insight library by distilling high-level insights rather than simply concatenating reasoning traces. Finally, the server broadcasts the updated library to all agents for the next iteration. The aggregated library can be re-used as an external tool to improve reasoning of either existing agents, or future agents that are not in the current network. We provide all the designed prompts that we use in Appendix D.

2.1 Implications of Connecting FL with FoT

Considering that both Federated Learning (FL) and Federation over Text (FoT) operate in a federated manner, we can leverage insights on how to optimize FL to similarly optimize FoT, realizing

Table 1: Comparison of key aspects between federated learning over gradients (e.g., FedAvg) and our proposed federation over text (FoT).

Components	**Federation over Gradients**	**Federation over Text**
Learning Objective	Learn a global model that generalizes to test data or new clients	Learn an insight library that scales up and improves the reasoning of existing and new agents
Local Work	Local training via any optimization method	Local problem solving via any thinking and self-reflection approach
Local $\rightarrow$ Server	Clients upload locally trained model weights or gradients	Agents upload reasoning traces from their problem-solving processes
Server Aggregation	Averaging over uploaded model weights or gradients	Text generation by an agent using curated prompts
Server $\rightarrow$ Local	Current learnt global model	Current learnt insight library
Privacy	Clients share model updates or parameters rather than raw data	Agents share abstracted insights rather than raw problem instances
Personalization	Clients maintain personalized models adapting to local distribution	Agents adopt personalized reasoning strategies based on the task at hand

more efficient and effective self-improvement. In federated learning, clients may adopt different optimization methods to improve training performance and efficiency. Analogously, in FoT, each agent may leverage distinct local reasoning strategies and prompt designs to generate insights.

For example, in federated learning, personalized federated learning addresses challenges such as uneven data distribution across clients, heterogeneous memory and computational capabilities, and diverse task requirements. Similarly, in FoT, agents can be personalized on each client, which is often easier to implement than in FL. For example, personalized FL uses techniques such as clustering to group clients with similar data distributions. Analogously, in FoT, we can cluster agents handling similar tasks to better identify relationships between their reasoning traces. In FoT, each agent may employ different language models or even different reasoning systems. Another aspect is evaluation. In federated learning, we can evaluate in two scenarios: performing inference on previous clients with new data, or letting new clients use the global model for inference. Similarly, in FoT, we can evaluate scenarios where we either let previous agents use the insight library to resolve new tasks, or send the insight library to new agents and let them complete tasks; see the next section for details. Future work should continue to fully explore the rich design space of FoT inspired by further connections with FL, such as personalization strategies, controlling distribution drift across rounds, and optimizing communication efficiency between agents and the server.

3 Applications

In this section, we study two applications where the insight library (that captures a set of shared, non-trivial principles of a type of problems) improves downstream performance and reasoning efficiency.

3.1 Mathematical Problems

In this application, our goal is to solve math problems at various difficulty levels, and we assume that solving them can inform each other. There are eight agents, each working on a benchmark dataset from AIME24, AIME25, AMC, CCEE, CNMO, WLPMC, V202412 Hard, and V202505 Hard from LiveMathBench. In each round, each agent uses the current insight library as reference and reasons over their local benchmark again. We experiment with DeepSeek-R1-Distill-Qwen-7B (DeepSeek) and Gemini 3 Pro as the base LLM models. In FoT, each agent and the server can use different models, but in this experiment, we use on the same LLMs in each row of the table below.

From Table 2, we observe that for challenging tasks such as AIME25 and V202412 Hard, FoT constructs useful insights that improve accuracies. Moreover, insights produced by stronger LLMs transfer effectively: when DeepSeek agents use a library generated by Gemini 3 Pro, performance improves substantially (comparing the first panel with the last row), demonstrating that FoT effectively distills and replays successful reasoning experience across agents. From Figure 2, we see that FoT constructs an insight library that leads to fewer loops and fewer generated tokens in the answers.

Table 2: Mathematical problem-solving accuracies across benchmarks and settings. With the FoT-generated library, agents can solve problems better. Stronger LLMs yield greater benefits, and with a library generated by Gemini in FoT, local DeepSeek agent achieves a much higher accuracy.

		AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard
DeepSeek	Single Agent	0.500	0.414	0.682	0.850	0.730	0.270	0.526	0.359
	FoT (Round 2)	0.500	0.414	0.711	0.850	0.730	0.300	0.526	0.306
	FoT (Round 3)	0.500	0.414	0.711	0.850	0.730	0.300	0.526	0.359
Gemini 3 Pro	Single Agent	**1.000**	0.933	0.935	**0.897**	**0.889**	0.909	0.762	**0.690**
	FoT (Round 2)	**1.000**	**0.966**	**0.935**	**0.897**	**0.889**	**0.909**	**0.905**	**0.690**
DeepSeek + Gemini 3 Pro Insight Library		0.816	0.588	0.682	0.850	0.730	0.316	0.526	0.359

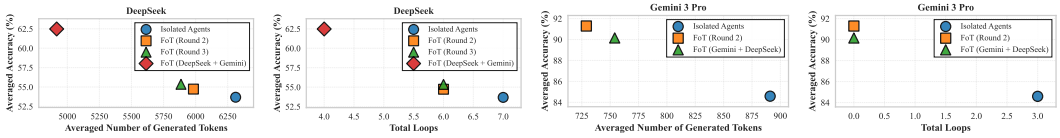

Figure 2: Comparison of reasoning efficiency and accuracy across different settings. With the presence of an insight library, the system generates fewer tokens during the reasoning processes while achieving better final performance. Additionally, with FoT, agents produce fewer loops, reducing the occurrence of repeated content (Pipis et al., 2025).

3.2 Machine Learning Insight Discovery

In this application, we aim to collectively discover insights from ML papers, which is different from prior works that build engineered workflows for paper writing. Each agent is assigned to one paper and tasked with identifying its main non-incremental novel contribution. We then evaluate how many future papers could be guided by insights in the library as their core contribution. Specifically, we input a paper and the insight library to an LLM, which judges on four aspects: concrete methodology usage, how methods are presented, whether the paper's core contribution would fundamentally differ without the insight, and whether the insight is novel, with details in Prompt 5.

We consider two settings. In the first, 281 agents each read one accepted ICLR 2023 paper, and we evaluate on ICLR 2024 accepted papers. All agents, the server, and the evaluation LLM use Gemini 3 Pro Preview. However, this LLM may already contain knowledge of ICLR 2024 papers. Therefore, in the second setting, we use Gemini 2.0 Flash: 453 agents read oral and spotlight papers from ICLR 2024, and we evaluate on ICLR 2025 papers. Gemini 2.0 Flash is designed to have no access to 2025 papers, as its cutoff date is August 2024, before the ICLR 2025 submission deadline.

We evaluate the percentage of subsequent ML papers whose core technical contribution can be covered by principles from FoT generated insight library (22 insights total). We compare against two baselines. First, we let a standalone Gemini model generate the top 200 insights based on the same insight definition and the keywords of ICLR 2023 papers (setting 1) or ICLR 2024 oral and spotlight papers (setting 2), then evaluate whether the insights guide papers from the following year. Second, we use Vertex AI retrieval-augmented generation (Lewis et al., 2020) (RAG) to build a datastore of all selected papers from the previous year, ask the LLM to generate top-200 insights with retrieved content as reference, and apply the same criteria. These baselines assess whether FoT improves insight discovery beyond the LLM's standalone capabilities and beyond simply storing old papers for retrieval. We also verify the validity of generated insights and demonstrate a subset in Appendix E. In summary, FoT generates insights corresponding to cutting-edge research topics from ICLR 2023 and 2024 papers, though under different terminology.

Table 3: Percentage of papers guided by insights generated from different methods. In both models, FoT improves over the baseline without the library ('Single Agent') and the baseline that simply stores previous tasks in RAG for retrieval ('RAG').

	Single Agent (Gemini)				**RAG (Gemini + Vertex AI)**				**FoT (Gemini + Insight Library)**			
	Overall	Oral	Spotlight	Poster	Overall	Oral	Spotlight	Poster	Overall	Oral	Spotlight	Poster
Gemini 3 Pro	**67.2**	41.3	**70.9**	**73.8**	33.3	66.5	58.1	69.2	66.4	**67.4**	67.4	68.7
Gemini 2.0 Flash	79.0	0.7	0.5	0.8	0.7	77.4	63.4	67.4	**79.6**	**78.5**	**76.1**	**75.5**

REFERENCES

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, et al. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, volume 33, pp. 9459–9474, 2020.

Chuan Li, Qianyi Zhao, Fengran Mo, and Cen Chen. Fedcot: Communication-efficient federated reasoning enhancement for large language models. *arXiv preprint arXiv:2508.10020*, 2025.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. Pmlr, 2017.

Amirkeivan Mohtashami, Florian Hartmann, Sian Gooding, Lukas Zilka, Matt Sharifi, et al. Social learning: Towards collaborative learning with large language models. *arXiv preprint arXiv:2312.11441*, 2023.

Charilaos Pipis, Shivam Garg, Vasilis Kontonis, Vaishnavi Shrivastava, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Wait, wait, wait... why do reasoning models loop? *arXiv preprint arXiv:2512.12895*, 2025.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*, 2025.

Haofei Yu, Zhaochen Hong, Zirui Cheng, Kunlun Zhu, Keyang Xuan, Jinwei Yao, Tao Feng, and Jiaxuan You. Researchtown: Simulator of human research community. In *International Conference on Machine Learning*, pp. 73051–73096. PMLR, 2025.

Yufan Zhuang, Chandan Singh, Liyuan Liu, Yelong Shen, Dinghuai Zhang, Jingbo Shang, Jianfeng Gao, and Weizhu Chen. Test-time recursive thinking: Self-improvement without external feedback. *arXiv preprint arXiv:2602.03094*, 2026.

A USAGE OF AI IN WRITING

We employ large language models (LLMs) primarily to improve the grammar and clarity of our writing and assist coding. All research ideas, directions, and decisions, however, are independently conceived and carried out by the authors.

B Related Work

Multi-Agent Collaboration. Recent agent systems explore collaboration mechanisms for solving complex tasks. A common paradigm decomposes a complex task into smaller subtasks, each solved by an individual agent, followed by aggregation of partial results (Tran et al., 2025). Related efforts aggregate knowledge at scale: Schmidgall et al. (2025) propose agent laboratory to aggregate human research artifacts via an LLM server, while ResearchTown (Yu et al., 2025) simulates a community of LLM agents for writing and reviewing papers. In contrast, Federation over Text (FoT) operates at a higher level of abstraction, focusing on discovering shared underlying principles or insights for solving related problems rather than coordinating end-to-end task execution.

Collaborative and Federated Learning. Federated learning (FL) (McMahan et al., 2017) aggregates model parameters from multiple clients while keeping data local. Existing FL multi-agent systems primarily focus on training a reinforcement learning model or agent model via classic parameter- or gradient-level aggregation (Li et al., 2025), which is insufficient for capturing abstract, high-level reasoning strategies. Social learning (Mohtashami et al., 2023) allows teacher agents to generate synthetic examples or abstract prompts that are aggregated to train a student model. However, the student solves the same task as the server, since social learning transfers knowledge from teachers to a student. FoT and FL share interesting connections (Section 2.1); and FoT operates in the semantic space while being compatible with heterogeneous agent models and domains.

Reusable Agent Experience. To embed past experience into LLM-based agents, existing approaches such as `Skills` and `Agents.md`[1] rely on human-authored or prompt-engineered artifacts to explicitly package domain knowledge and agent instructions. Hence, researchers further study metacognition (Didolkar et al., 2025) and self-improvement (Zhuang et al., 2026) strategies, aiming to automatically capture and reuse previously successful behaviors by storing them as structured handbooks that guide future decision-making. FoT can plug in any existing self-improvement method as the agent local thinking approach, and collaboratively construct and refine insights given agent reflection traces in a federated manner.

C Complete Algorithms

Algorithm 1: Federation over Gradients

Input: Initial global model, client datasets
for *each iteration* **do**
  for *each **Client*** **do**
    Local optimization starting from the current global model
    Upload new model weights
  end
  Server:
  Collect all local model weights
  Aggregation over model weights
  Broadcast the global model to clients
end
Output: Global model

Algorithm 2: Federation over Text

Input: Empty insight library, agent tasks, LLM
for *each iteration* **do**
  for *each **Agent*** **do**
    Local reasoning given the LLM and current insight library
    Upload reasoning traces to the server
  end
  Server:
  Gather all reasoning traces
  Update the insight library
  Broadcast the new library to agents
end
Output: Insight library

D Complete Prompts

Prompt 1 is used for the reflection step before an agent generates reasoning traces during Step 2: Local reasoning and reflection in Figure 1.

Prompt 1: Definition of reflection on solution

[1] `https://github.com/anthropics/skills`; `https://agents.md/`

```
Analyze the solution below to extract procedural knowledge that can be
  turned into reusable insights.
Insights are instructions that teach how to complete specific tasks in
  a repeatable way.

Problem:
{problem}

Step-by-Step Solution:
{solution}

Your task: Extract procedural knowledge and reusable patterns that can
  be packaged as insights. Focus on:

1. **Procedural Patterns**: What step-by-step procedures were used?
  How can these be repeated?
2. **Decision Points**: What conditions determined which approach to
  use? When should each technique apply?
3. **Reusable Techniques**: What methods, strategies, or workflows can
  be applied to similar problems?
4. **Key Insights**: What made this approach effective? What should
  someone know to use it correctly?
5. **Applicability**: What types of problems would benefit from these
  techniques?

Output your analysis covering:

### I. Procedural Knowledge
- Break down the solution into clear, repeatable procedures
- Identify the sequence of steps that led to success
- Note any decision-making criteria or conditions

### II. Reusable Techniques and Methods
- List specific techniques, strategies, or workflows used
- For each technique, identify:
* When it should be used (conditions/triggers)
* How it was applied (concrete steps)
* Why it was effective (insights)
* What problems it could solve (applicability)

### III. Critical Insights and Guidelines
- What key insights made this solution work?
- What common pitfalls should be avoided?
- What variations or edge cases should be considered?

Focus on extracting actionable, procedural knowledge that can be
  packaged as reusable insights for similar problems.
```

Prompt 2 is used for the generation of reasoning traces during Step 2: Local reasoning and self-reflection in Figure 1. The description includes our definitions of reasoning traces.

Prompt 2: Reasoning Trace Prompt

```
Extract reusable reasoning traces from the solution below. Analyze the
  solution and reflection to identify concrete, actionable insights
  that can be applied to similar problems.

Problem: {problem}

Solution: {solution}
```

```
Reflection: {reflection}

**Your Task:**
Identify and extract all reusable reasoning trace, techniques, and
  methods used in the solution. Each trace should be a concrete
  procedure that can guide someone to solve similar problems.

**What Makes a Good Reasoning:**
- A specific technique or method that was used in the solution
- Something that can be applied to similar problems, not just this one
- Has clear steps that can be followed
- Includes guidance on when to use it

**Description Must Include:**

1. **When to use**: Explain when this skill should be applied. What
  types of problems? What conditions must be met? What situations
  trigger this skill?

2. **Step-by-step**: Provide detailed, numbered steps (1) 2) 3) ...)
  that explain exactly how to apply this skill. Include specific
  techniques, formulas, methods, or approaches. Be concrete and
  actionable.

3. **Key insights** (optional): Important considerations, common
  pitfalls, tips, or why this approach works.

**Output Format (Simple JSON):**
Output a simple JSON object with skill names as keys and descriptions
  as string values:

{{"skill_name": "description"}}

Format Rules:
- Use valid JSON format
- Each insight name must start with "skill_"
- Each description is a single string containing: "When to use:" and
  "Step-by-step:" sections
- Steps must be numbered: 1) 2) 3)
- Keep JSON simple - no nested objects, just key-value pairs
- Escape quotes in descriptions with backslash: \\"
```

Prompt 3 is used for clustering and identifying connections between reasoning traces before updating the insight library during Step 4: Aggregate traces to update the library in Figure 1.

Prompt 3: Prompt for constructing relationship between reasoning traces

```
You are analyzing a collection of problem-solving insights to
  understand their relationships and structure, following principles
  from hierarchical insight learning and knowledge graph construction.

**Collected Insights ({len(insight_store)} total):**
{insights_text}

**Your Task:**
Analyze these insights and build a profiling of their relationships:

1. **Identify Insight Clusters**:
Group related insights that share:
- Common themes or problem-solving patterns
- Same domain or application area
- Similar approaches or techniques
```

```
Insights in the same cluster should be related. Individual insights
  can be in separate clusters if they don't fit with others.

2. **Map Insight Relationships**:
Build a relationship graph that records:
- **Prerequisite relationships**: Insights that must be learned/used
  before others
- **Composition relationships**: Insights that can be chained/composed
  together
- **Alternative relationships**: Different approaches to the same
  problem
- **Complementary relationships**: Insights that work well together
- **Derivation relationships**: Insights derived from or based on
  others
- **Similar relationships**: Insights that are similar but not
  identical

**Important Guidelines:**
- **Clusters**: Create meaningful clusters based on shared themes.
  Individual insights can be in separate clusters.
- **Relationships**: Record all important relationships - insights
  don't exist in isolation
- **Be thorough**: Map relationships between insights within clusters
  and across clusters
- **Preserve Original Insights**: Keep original insight names in
  clusters and relationships for traceability

# Output Format:
{{
    "clusters": [
    {{
        "cluster_id": 0,
        "cluster_name": "Domain/Theme Name",
        "insights": ["insight_name1", "insight_name2",
  "insight_name3"],
        "common_theme": "What these insights share in common",
        "domain": "mathematics/algebra/geometry/etc"
    }}
    ],
    "relationships": [
    {{
        "insight_a": "insight_name1",
        "insight_b": "insight_name2",
        "relationship_type":
  "prerequisite/complementary/alternative/similar/
        derived_from/composes_with",
        "description": "How these insights relate to each other"
    }}
    ]
}}

**Output your analysis as JSON only:**
```

Prompt 4 is used for updating the insight library during Step 4: Aggregate traces to update the library in Figure 1.

Prompt 4: Definition of Insight Library

```
You are extracting fundamental knowledge from a collection of
  problem-solving insights.

**Context:**
```

```
- Total insights collected from clients: {len(insight_store)}
- These insights were derived from solving specific problems
  (bottom-up approach)
- Your task is to extract multi-disciplinary, fundamental knowledge
  (top-down approach) which can be generalized to multi-domain
  problem-solving.
- The extracted knowledge should be able to DERIVE and GUIDE the use
  of the collected insights
- Original insights should NOT be forgotten - they remain available
  for direct use

**Existing library (if any):**
{existing_lirary if existing_lirary else "None"}

**All Client insights (New insights to Integrate):**
{all_insights_text}

**insight Clusters (from Step 2):**
{clusters_text if clusters_text else "None identified"}

**Relationships (from Step 2):**
{json.dumps(profiling.get("relationships", []), indent=2) if
  isinstance(profiling, dict) and "relationships" in profiling else
  "None identified"}

**Your Task:**
You have:
1. Previous library (existing general insights)
2. All client insights (new specific insights from problem-solving) -
  {len(insight_store)} insights total

**CRITICAL: Extract Many Fundamental and Cross-Domain insights**

Your goal is to extract a comprehensive set of fundamental,
  cross-domain insights that can be derived and applied beyond their
  original domain. DO NOT over-merge insights.

1. **Knowledge Distillation - Extract Reusable insight Primitives**:
Following the principle of creating object-centric, transferable
  insight primitives:
- For EACH insight cluster, extract multiple fundamental insights that
  capture the core essence and variations
- For EACH domain area, extract cross-domain insights that can be
  applied to other fields
- Extract reusable components that can be applied across contexts
- Create insight primitives that serve as building blocks for
  composition
- General enough to transfer to other domains, specific enough to be
  actionable
- **Extract insights from ALL major clusters and domains
  comprehensively, not just merge everything into a few insights**

2. **Hierarchical insight Learning - Multi-Level Organization**:
Following GSL framework principles:
- **Fundamental Level**: Core principles and theories that underlie
  multiple domains - extract many of these
- **General Level**: Broad techniques applicable across related
  problem types - extract many of these
- **Cross-Domain insights**: insights that can be derived/applied to
  fields beyond their origin - extract many of these
- Each level should guide the level below it
- insights should be organized in a hierarchy where higher levels
  subsume lower levels
- **Extract insights at all levels comprehensively**
```

```
3. **Cross-Domain Transfer - Extract Universal Patterns**:
- For each insight cluster, identify patterns that appear across
  different fields
- Extract insights that can be DERIVED into other domains (e.g., an
  insight from one field that applies to another)
- Create insights that capture universal principles (e.g., attention
  mechanisms, optimization strategies, regularization techniques)
- Document which insights transfer well across domains in the "When to
  use" section
- **Extract many cross-domain insights that can be applied beyond
  their original field**

4. **Preserve and Generalize Original insights**:
- DO NOT simply merge all insights into a few generic ones
- Instead, extract fundamental versions of original insights that are
  more general but preserve their essence
- For similar insights, create a more fundamental version that can
  guide both, but also preserve important variations
- Original insights remain available - general insights should be able
  to derive/guide them
- **Extract fundamental versions of important original insights
  comprehensively, not just merge them**

5. **Better insights - More Fundamental and Cross-Domain**:
- More fundamental and general than original insights, but still
  specific enough to be actionable
- Can guide the use of original insights and be derived into other
  fields
- Include cross-domain insights and patterns in the description
- Still actionable with clear step-by-step instructions
- **Each insight should specify which domains it can be applied to
  beyond its origin**

**Description Format (Must match client.py exactly):**
Each description must be a single string containing all the following
  sections, separated naturally in the text:

1. **When to use**: Detailed explanation of when to apply this
  insight, what problem types trigger it, and what conditions must be
  met. Be comprehensive and specific.

2. **Step-by-step**: Detailed, numbered steps (1) 2) 3) ...) that
  explain exactly how to apply this insight. Include specific
  techniques, formulas, methods, or approaches. Be concrete and
  actionable.

3. **Cross-Domain Transfer - Identify Universal Patterns**:
Following research on transfer learning and domain adaptation:
- **Cross-domain insights**: Patterns that appear across different
  fields (universal principles)
- **In-domain patterns**: Patterns specific to a domain but general
  within it
- **Shared insights**: Common principles that appear in multiple
  domains
- Document which insights transfer well across domains

**Output your extracted knowledge as JSON only:**

**Format Rules:**
- Use valid JSON format
- Each insight name must start with "insight_"
- Each description is a single string containing: "When to use:" and
  "Step-by-step:" sections (and optionally "Key insights:" and
  "Example:")
- Steps must be numbered: 1) 2) 3)
```

```
  - Keep JSON simple - no nested objects, just key-value pairs
  - Escape quotes in descriptions with backslash: \\"
```

Prompt 5 is used for determining whether a paper can be covered by insights from the library in Section 3.2.

Prompt 5: Definition of criteria for matching insights and the library

```
  Evaluation Criteria - An insight guides the paper ONLY IF ALL of the
    following are true:

  1. CONCRETE METHODOLOGY USAGE: The insight's methodology or approach
    is concretely used in the paper's methods/approach section, not just
    theoretically relevant or mentioned in motivation.

  2. METHODS SECTION PRESENCE: The insight must be related to how the
    paper actually implements its solution (methods, algorithms,
    techniques), not just in problem statement or related work.

  3. COUNTERFACTUAL TEST: The paper's core contribution would
    fundamentally differ or fail without this insight. Ask: "If the
    authors didn't know this insight, could they still arrive at the
    same core solution?"

  4. SPECIFICITY: The insight must specifically address a key challenge
    or component of the paper's solution, not just be generally
    applicable background knowledge.
```

E REPRESENTATIVE INSIGHTS

In this preliminary work, we show the three most representative insights from the library in Setting 1 (derived from ICLR 2023 top 25% papers) and Setting 2 (derived from ICLR 2024 Oral and Spotlight papers) here.

With Gemini 3 Pro Preview and ICLR 2023 top 25% papers, we obtain three insights: structured output prediction (Prompt 1), causal representation learning (Prompt 2), and physics-informed deep learning (Prompt 3), which provide interesting ideas and instructions on how to approach related research problems. These topics remain cutting-edge research questions today.

Insight 1: insight_structuredOutputPrediction (setting 1)

```
When to use: When the model output is not a simple class or scalar, but
    a complex structured object (e.g., a sequence, a tree, a matching, a
    code program) where dependencies between output elements must be
    modeled. Step-by-step: 1) Define the output space constraints and
    dependencies (e.g., grammar rules, bi-partite matching constraints).
    2) Use autoregressive models (like Transformers) to generate
    elements sequentially conditioned on history, or non-autoregressive
    models with iterative refinement. 3) Apply structural losses (e.g.,
    connectionist temporal classification, graph edit distance, optimal
    transport cost) rather than simple pointwise error. 4) Use inference
    techniques like beam search or constrained decoding to find the most
    likely valid structure. Key insights: This covers skills like
    hierarchical bipartite matching, permutation equivalent modeling,
    and program synthesis.
```

Insight 2: insight_causalRepresentationLearning (setting 1)

```
When to use: When building models that must robustly identify
    cause-and-effect relationships rather than spurious correlations,
    enabling interventions and counterfactual reasoning. Step-by-step:
    1) Model the system using a Structural Causal Model (SCM) or
    Directed Acyclic Graph (DAG). 2) Distinguish between invariant
    causal mechanisms (stable across environments) and spurious
    associations. 3) Use interventions (do-calculus) or environmental
    diversity to discover the causal graph. 4) Learn representations
    that satisfy conditional independence constraints implied by the
    causal structure (disentanglement). 5) Optimize for Invariant Risk
    Minimization (IRM) to find predictors that are optimal across all
    interventional distributions. Key insights: This unifies skills like
    causal mediation analysis, front-door prompting, and causal shift
    diagnosis.
```

Insight 3: insight_physicsInformedDeepLearning (setting 1)

```
When to use: When modeling physical systems governed by partial
    differential equations (PDEs), conservation laws, or geometric
    constraints (e.g., fluid dynamics, structural mechanics, climate
    modeling) where purely data-driven models violate physical validity
    or require excessive data. Use this to integrate physical knowledge
    directly into the learning process. Step-by-step: 1) Formulate the
    governing physical equations (residuals) of the system (e.g.,
    Navier-Stokes, Eikonal equation). 2) Design the neural network to
    output the physical state variables (velocity, pressure) given
    spatiotemporal coordinates. 3) Compute derivatives of the outputs
    with respect to inputs using automatic differentiation (Autograd) to
    evaluate the PDE residuals. 4) Construct a composite loss function:
    Data Loss (fitting observations) + Physics Loss (minimizing PDE
    residuals) + Boundary/Initial Condition Loss. 5) Optionally, use
    hard constraints (ansatz) or coordinate transformations to enforce
    boundary conditions by construction. Key insights: This guides
    specific skills like PDE-constrained latent dynamics, soft contact
    modeling, and neural operator learning.
```

In Setting 2, insights generated by Gemini 2.0 Flash are generally shorter than those by Gemini 3 Pro. However, even without knowledge of future papers, we still obtain interesting insights such as neuro-symbolic reasoning (Prompt 4) and parameter-efficient model adaptation (Prompt 5), which corresponds to what is currently known as parameter-efficient fine-tuning. It also produces the zero-to-one capability testing insight (Prompt 6), which can be adapted to systems research.

Insight 4: insight_neuroSymbolicReasoning (setting 2)

```
When to use: When a problem requires both the perceptual flexibility of
    neural networks (handling noisy, unstructured data) and the
    precision/interpretability of symbolic logic (handling rules, math,
    constraints). Step-by-step: 1) Decompose the system into a Neural
    Module (perception/pattern matching) and a Symbolic Module
    (reasoning/execution). 2) Design an interface layer: converting
    neural outputs into discrete symbols (concepts) or differentiable
    approximations (soft logic). 3) Use the Symbolic Module to execute
    logical rules, programs, or arithmetic on the extracted symbols. 4)
    Train end-to-end (via gradients through soft logic) or separately
    (via REINFORCE or supervision) to optimize both perception and
    reasoning accuracy. Key insights: This abstracts skills like
```

```
    neuro-symbolic prompt construction, logic constraint integration,
    and program synthesis.
```

Insight 5: insight_parameterEfficientModelAdaptation (setting 2)

```
When to use: When adapting large pre-trained 'foundation' models (LLMs,
    ViTs) to specific downstream tasks without the computational cost of
    full fine-tuning. Step-by-step: 1) Freeze the bulk of the
    pre-trained model parameters. 2) Introduce a small set of trainable
    parameters: Adapters (bottleneck layers between existing layers),
    Prompt/Prefix Tuning (learnable input tokens), or Low-Rank
    Adaptation (LoRA - decomposing weight updates into low-rank
    matrices). 3) Train only these new parameters on the task-specific
    dataset. 4) Optionally, use techniques like weight averaging (model
    soup) or dynamic routing to combine multiple adaptations. Key
    insights: This covers skills like minimalist adapter co-training,
    prompt engineering, and efficient fine-tuning protocols.
```

Insight 6: insight_zeroToOneCapabilityTesting (setting 2)

```
When to use: When assessing the functional value of a new technology or
    method to distinguish between efficiency gains and capability
    unlocks. This applies to product management, feature prioritization,
    and evaluating 'breakthrough' research. Step-by-step: 1) Ignore
    quantitative metrics (speed, cost, error rate) initially 2) Define
    the 'Capability Boundary' of the previous
    state-of-the-art---specifically what tasks it physically could not
    perform 3) Ask the binary question: 'Does the new solution enable a
    task that was previously impossible/intractable?' 4) If the answer
    is 'No' (it just does existing tasks better), classify as '1-to-N'
    (Incremental) 5) If the answer is 'Yes' (it opens a new action
    space), classify as '0-to-1' (Non-Incremental) 6) Prioritize 0-to-1
    capabilities for strategic innovation as they create new markets.
```