

Learning Optimal Message Representations for Agentic Communication

Shashwat Gupta¹, Anson Bastos¹, Mayukh Das¹, Supriyo Ghosh²,
Nagarajan Natarajan³, Chetan Bansal¹, Saravan Rajmohan¹

¹M365 Research, Microsoft ²Inception, G42, UAE ³Microsoft Research, India
guptashashwatme@gmail.com, {ansonbastos, mayukhdas}@microsoft.com

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities in multi-agent collaborative problem-solving, albeit a gap exists. Existing frameworks predominantly rely on natural language as a primary representation for agentic communication. However, natural language could be ambiguous, biased and verbose. Furthermore, recent works have shown that alternative representations can enhance performance in LLMs on certain tasks. However, current approaches lack the intelligence necessary to understand, learn or apply optimal communication representations adaptively. In this paper, we propose to dynamically learn the optimal message representations to enhance agentic performance. We model the optimization problem as an Expanding Markov Decision Process (EMDP) and propose our method named OPTi-MACS. We evaluate our system across benchmark datasets of collaborative problem-solving. The results show significant performance improvements while maintaining efficiency. Our work bridges the gap between rigid communication protocols and open-ended natural language by providing an adaptive framework that learns task-aware structural representations.

1 Introduction

Communication is essential among humans and forms the very fabric of society. Along with natural language, we convey information using gestures, mathematical expressions, multimedia, etc. Depending on the task, different forms of communication are important to accomplish the end goal in an effective and efficient manner. For example, in a driving scenario, the vehicles and the traffic system efficiently coordinate primarily by visual means using colored lights, signs, markers, etc. rather than relying solely on natural language (NL). In the realm of human-computer interaction, traditional forms of communication primarily consisted of user commands (using text, touch, gestures) that

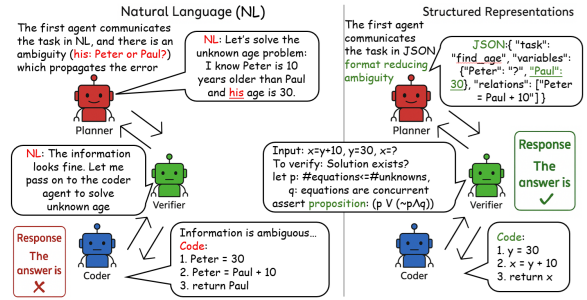


Figure 1: In agentic systems, optimal message representations (JSON, proposition, code etc.) enable agents to collaborate more effectively, reduce *ambiguity*, *bias*, prevent error propagation and improve task performance.

perform desired action through APIs encoded in high or low level machine instructions. Lately, with the advent of Large Language Models (LLMs), natural language has become the dominant mode of communication, between LLM and human agents, and between LLM agents. This prevalence extends to multi-agent systems, where specialized agents collaborate through natural language exchanges to solve complex tasks (Li et al., 2023; Wu et al., 2023; Park et al., 2023). But one question naturally arises, even among text-based agents, whether natural language in general is the best representation of communication to solve a given task of varying complexity and nature (cf. fig. 1)?

Recent advances in LLM reasoning have challenged the assumption that natural language represents the optimal intermediate representation for cognitive processes. Many works such as Program-of-Thought (Chen et al., 2022), X-of-Thought (Ding et al., 2024), PLaG (Lin et al., 2024) have demonstrated that alternative structured representations can enhance reasoning and decision-making capabilities. This ineffectiveness of using NL is compounded in multi-agent settings where multiple LLM agents are involved and pass messages using natural language as the de facto mode. In this vein, recent research (Chen et al., 2024) suggests that

employing non-natural language as intermediate formats can help achieve improvements in reasoning performance during agentic communication.

Despite the promising developments in literature, current single and multi-agent frameworks lack a principled and data-driven approach to adaptively select the optimal format of the intermediate representation based on task requirements. Existing approaches rely on static format selection or on hand-crafted prompts that let the LLM choose the appropriate format for the task based on its inductive biases. Both the static formats and the ones suggested by the LLM may not be optimal. Moreover, the prior works rely on a fixed set of structures, whereas new optimal representations could dynamically emerge as the agent interacts with the environment. The fundamental challenge lies in developing systems that can *learn* optimal message formatting strategies *dynamically* from historical data or from the *environment* while maintaining *seamless integration* with existing systems.

In this paper, we address the challenge of learning the optimal representation to solve a given sub task of a problem. We propose to learn the optimal representations in a *data-driven* manner, from the observations/rewards as the agents interact with the environment. Since novel optimal representations could emerge as the agent learns, we further propose to dynamically adapt to the set of tasks and communication representations rather than optimizing for a fixed set of representations.

We aim to *learn* optimal representations (actions) *dynamically* for a given task (state) as the agents interact with the environment, e.g. what is the best format for an equation solving task. This naturally reduces to a sequential decision-making problem which can be modeled as a Markov Decision Process (MDP). As the representations could evolve dynamically, we work in a setting where the states and actions are evolving/expanding. Thus, we pose the optimal representation selection problem as an Expanding Markov Decision Process (EMDP) and introduce our method called OPTiMACS- *Optimal Prompt Transformation in Multi-Agent Communication System*. Considering the agents’ communication trajectory of length L_T and our desire to learn the optimal message representation at each step/turn, a naive approach to learn the optimal representation for a task from the historical data would incur a complexity of $\mathcal{O}((|\mathcal{T}||\mathcal{F}|)^{L_T})$, where \mathcal{T}, \mathcal{F} (cf. § A.7) are the set of tasks and representations/formats respectively.

We thus employ methods from RL to efficiently learn the state action space in linear complexity.

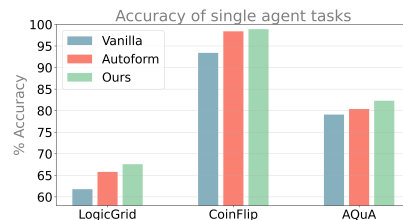


Figure 2: Task performance improves when prompt is formatted as per discovered optimal representation.

Motivating the need to learn the optimal representations in a data-driven manner, consider the following problem assigned to the system (cf. fig. 3): “Gerald works at a daycare that pays him \$30 every day. He worked for an *entire week*,...he spent \$100 on... How much money does he have left?”. The problem states the person works for 7 days and saves \$110. Plain text (NL) based solution arrives at the answer of \$50, which stems from the NL bias that a working week consists of 5 days. Prompting the LLM to think in code (suggested by our method) enables the model to explicitly define the number of days as 7, thereby eliminating *common biases associated with NL reasoning*, arriving at the expected answer. Further examples are in the Appendix § A.6. To make the motivation concrete, we perform a study on single agent (LLM) tasks where we compare results of plain text, optimal representation decided by LLM and optimal representation derived by exploring the data (our method). We present the findings in fig. 2 where we see that our method consistently outperforms the baselines (p-value<0.05 using Student’s t-test, std. dev. < 2%, LLM: GPT-4). The errors induced by natural language ambiguity get amplified in a multi-agent scenario with many turns, necessitating the discovery of task-specific optimal message representations to input to the LLM agent.

Contributions: To summarize, we make the following contributions: (1) We formalize the problem of selecting the optimal communication representation in a multi-agent system as an evolving MDP (2) We propose our method OPTiMACS, to learn the optimal policy for task and message representation in a *data-driven* and *dynamic manner*. (3) Extensive experiments are conducted on 7 benchmark datasets to show the efficacy of our method. For reproducibility, we provide pseudo-code and prompts in the Appendix.

2 Related Works

Here we briefly compare our work with related works for LLM agents and defer detailed discussion to the Appendix.

2.1 Agentic Communication Protocols

Traditional agent communication has relied on standardized protocols like KQML (Finin et al., 1994), FIPA-ACL (FIPA, 1997), MASIF (Milojicic et al., 1999), Web Services (Berners-Lee et al., 2001), Event Stream Processing (Cugola and Margara, 2012) etc. In the era of LLMs, researchers have explored communication using function calling mechanisms (Schick et al., 2023) and tool usage (Schick et al., 2023). Recent standardization efforts include protocols such as MCP (Anthropic, 2024), ACP (Consortium, 2024a), A2A (Group, 2024) and ANP (Consortium, 2024b). While these establish agent communication infrastructure, they provide static, rule-based schemas that cannot adapt to varying task requirements or agent capabilities.

2.2 Structured Communication among LLM Agents

Recent work on structured communication focused on leveraging the inductive biases in LLM for format generation (Chen et al., 2022), though without mechanisms for dynamically learning or domain adaptation. These methods suffer from over-reliance on static LLM biases without learning from environment feedback.

Our work differs from existing research in that, unlike existing approaches that focus on either agent coordination infrastructure or static / LLM prompted format generation, we focus on the optimization of message representations/structures/formats in multi-agent communication by *dynamically learning* from the environment.

3 Problem Formulation

In this section, we formulate the problem setup. For readers unaware of Markov Decision Process (MDP) and related techniques, we kindly refer to the Appendix (A.0.1) or (Sutton and Barto, 2018).

We operate in a multi-agentic setting where multiple specialized LLM agents collaborate *sequentially* to solve a complex task. Let there be N LLM agents $\{\mathcal{A}_i\}_{i=1}^N$, passing messages to each other over T turns. In any turn $t, 1 \leq t \leq T$, agent \mathcal{A}_i sends a message $m_{(i,j)}^t$ to next agent \mathcal{A}_j in the pipeline to share information and context.

We aim to develop a module $F(m_{(i,j)}^t) \rightarrow m'$ that transforms message from \mathcal{A}_i to \mathcal{A}_j . We optimize $m' = F(m_{(i,j)}^t)$ such that m' is in a (near) optimal structure/representation (defined by goal success) given the complex task and the nature of \mathcal{A}_i and \mathcal{A}_j . Considering we desire to learn the optimal structures (action) for a given task (state), we naturally pose the problem as a Markov decision process (MDP) (Puterman, 1994) and aim to learn the optimal policy $\pi^*(F|\text{Task}_i, m, t, \mathcal{A}_i, \mathcal{A}_j)$ which maps a given task to the optimal representation. Simpler alternatives like contextual bandits¹ would not solve for the sequential nature of the problem and A/B testing to find the best format is not practical (cf. § A.1). Formally, we aim to learn the policy π^* such that the end goal of the multi-agent communication is achieved. This can be represented as:

$$\operatorname{argmax}_{\pi \in \Pi} \sum_{\text{task}} f_s \left(\text{task}, F(m) \middle| \begin{array}{l} F = \operatorname{argmax}_{f \in \mathcal{F}} \pi(f|\text{context}) \end{array} \right)$$

where Π is the space of all possible policies, $f_s(\cdot) \in \mathcal{R}$ is a functional indicating the degree of task (function) success.

We define the MDP by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where \mathcal{S} is the true state space, \mathcal{A} is the action space, \mathcal{P} represents transition probabilities, \mathcal{R} is the reward function. We thus formulate the MDP as: (i) **State Space** \mathcal{S} : Task types $t \in \mathcal{T}$ representing generalizable abstraction of messages (e.g., information_request, logical_reasoning, decision_strategy) (ii) **Action Space** \mathcal{A} : Message structures/representations/formats $f \in \mathcal{F}$ including structures like JSON, XML, tabular data, decision trees, etc. Note here we use the term structures to mean representations (formats) and do not limit the search space to only include “structured” objects (e.g., JSON). (iii) **Reward Function** $R(t, f)$: A Success metric that measures goal attainment (e.g., +1 for success, else -1) (iv) **Policy** $\pi(f|t)$: Probability distribution over formats/representations given task type.

In our problem setup, we allow the tasks and the representations to evolve as the learning progresses. This is needed because, in real-world problems, the agents encounter new information and we may not

¹In the single agent (1-step) setup the MDP simplifies to a Contextual Bandit. Refer Appendix A.1 for details.

always know the tasks and structures beforehand. In later sections (cf. fig. 4), we show empirically that the evolving/dynamic state/action formulation benefits the performance over fixed state/action space i.e. standard MDP. Thus, we propose to evolve the MDP as the agent encounters new states (tasks) and actions (structure formats) while learning. Formally, we expand our state space \mathcal{T}_{old} to $\mathcal{T}_{new} = \mathcal{T}_{old} \cup \{t_{new}\}$ on encountering new task t_{new} . Similarly the action space \mathcal{F}_{old} is updated as $\mathcal{F}_{new} = \mathcal{F}_{old} \cup \{f_{new}\}$ on encountering new structure format f_{new} . As the states and actions evolve during learning, we work in the setting of an Evolving Markov Decision Process (EMDP). One point to note is that the tasks themselves are not observed directly and our method estimates the task based on the observations. As such the assumed MDP, in theory is a partially observable one, where one might maintain some belief states from the actual environment observations, which can be intractable in real multi-agentic problem spaces. However, we observe in our experiments that the probability distribution of the states given an observation has a large mass around a state, with high accuracy (cf. § 6.1). Thus, we can approximately know the state and the problem, at a given time, can be approximated as an MDP to enable tractable policy learning. In the below proposition, we show the convergence of learning the proposed EMDP (proof in Appendix § A.2):

Proposition 1 (Convergence of the proposed EMDP based algorithm). *Consider the EMDP = $(S_e, A_e, P_e, R, \gamma)$, with value function V . Let T denote the update operator (including expanding S_e, A_e) s.t. $V_{k+1} = T(V_k)$, then $\lim_{k \rightarrow \infty} \|V_k - V^*\| = 0$, where V^* is the optimal value function for the states and actions explored till k iterations.*

4 Learning Optimal Representations for Agentic Communication

Having seen the problem setup in the previous section, this section outlines our method, OPTiMACS which consists of the following components: 1) Task classification 2) Policy for optimal representation selection. The message from the source agent \mathcal{A}_i is categorized into the appropriate task. This is done using an LLM, since LLMs have exhibited capabilities for few/zero-shot text classification tasks (Brown et al., 2020). Once the task is decided, the learnt policy then selects the optimal structure using which the message is formatted for commu-

nication with destination agent \mathcal{A}_j . The policy is learnt using an adapted policy learning framework as we shall see in the next sections. The system overview for OPTiMACS is in figure 3.

1) Initialization: In order to initialize the task and representations, we sample and process the data (if available) to build a knowledge base (KB), by prompting the LLM. In case of a pure online setting, we bootstrap the KB from prior knowledge of data from similar domain. The novel tasks and representations are then updated as the learning progresses. Note, we do not aim for OOD generalization but adaptation to the concerned task/domain.

2) Task categorization: Tasks serve as an abstraction to help generalize to novel messages. Prior to selecting the optimal representation using the policy, the system needs to classify the incoming message into one of the known tasks or a new one. For this categorization, we use an LLM. We warm start the task taxonomy with a few known categories and adopt a chain of thought style prompting, where the LLM is prompted to first review the message and decide whether the message fits into one of the existing tasks or a new task category needs to be added to the set, followed by the categorization of the message into the task. Following are the inputs to the LLM: (i) Source agent name and description (e.g., “problem_solver: mathematical reasoning agent”) (ii) Target agent name and description (e.g., “code_executor: computational verification agent”) (iii) Raw message content requiring formatting (iv) Current task taxonomy with descriptions.

Thus, the module considers the message as well as the agent context in order to decide the task category. If the existing task categories do not fit the current message context, the module (LLM) creates a new task category. The prompt for selection of task is provided in the Appendix § A.11. Once the task is decided, we then provide the task to the policy which learns and decides the optimal representation for that task and agent pair.

3) Policy for Representation Selection: We have seen the formulation of the problem-setting as expanding MDP. This module learns a policy (π) to sample from the action (structure) distribution given the state (task). In order to learn the optimal policy π^* , we design a policy learning method for our problem setting. The algorithm is an off-policy method to allow for exploration using a behavior policy $\mu(a|s)$. At inference we sample greedily from the learnt optimal policy. Exact methods would require an exponential complexity to com-

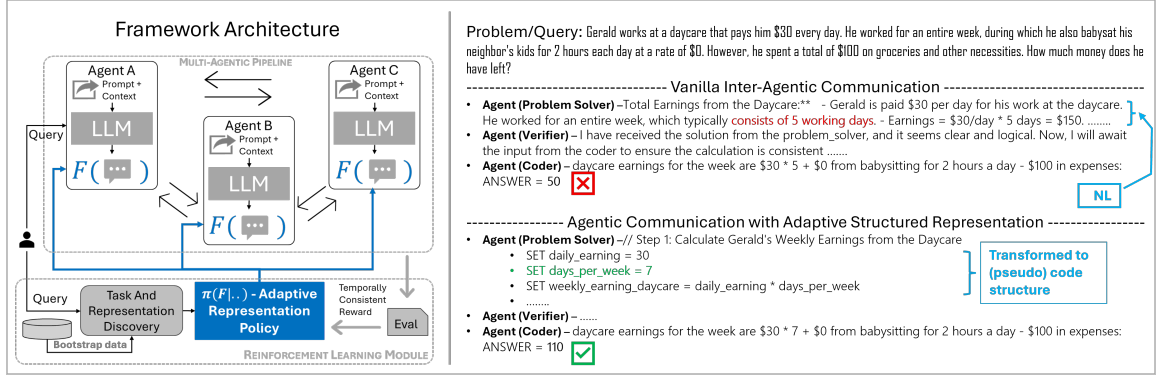


Figure 3: Left figure shows the overview of our method which consists of two components namely the task categorization and the policy learning module (in blue). The policy module learns the policy for the chosen task using historical data or online from the reward on achieving the end goal. Right figure compares the outcomes on a sample problem, using base and our method. The vanilla system chooses natural language and arrives at the wrong answer whereas our method selects code as the message representation to deduce the correct answer.

pute the system dynamics (cf. § 1). Thus, we approach the solution using trajectories as the agents explore the environment and employ Monte Carlo (MC) based methods (Sutton and Barto, 2018).

(3.1) Discovery of Representations: In our method, the behavior policy explores the tasks and representations followed by policy optimization. We first describe the behavior policy before we see how we use it in our algorithm to find the optimal policy ($\pi^*(a|s)$). The behavior policy $\mu(a|s)$ consists of sampling using three strategies namely: 1) Sampling from Q values (Q) 2) LLMs as a sampler (L) and 3) Diverse Exploration (D). The motivation for using these policies comes from the seminal work by Freud (Freud, 1989), which suggests that human behavior is influenced by unconscious memories/patterns and rational thoughts. Inspired by human behavior, the behavior policy samples actions from its “unconscious” learnt patterns (Q-values), “conscious” rational module (LLM) and random exploration (diversity policy). In order to give differential preference to each of the samplers, we use strategy-wise weights $\mathbf{w} = [w_Q, w_L, w_D]$.

Sampling from Q values (Q): This policy samples structures from the learnt Q values, akin to a scoreboard, in an ϵ -greedy manner. Specifically, the probability of the action (message structure) is computed as: $\pi_Q(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a'} \exp(Q(s,a')/\tau)}$ if $\text{random}() > \epsilon$ else $\frac{1}{|F|}$. where τ is the sampling temperature. Thus this policy encourages message structures that are known to work well for a task.

LLMs as a sampler (L): This policy selects the action (message structure) by prompting an LLM

with comprehensive context: 1) Source and target agent context descriptions 2) Message content and categorized task type 3) Historical pass rates (% success) for each structure previously used with this task 4) Currently overused formats to encourage diversity 5) High-level optimization goals (e.g. enhance clarity, efficiency etc.). In doing so, we look for representations that are specifically designed with the current query in mind. This phase also allows for addition of new message structures as needed and decided by the LLM. The prompt for structure selection via LLM is given in §A.11.

Diverse Exploration (D): Here, we explore diverse formats that did not appear for concerned task but appeared for other tasks. To give a fair chance to the under-sampled formats, we sample inversely by frequency. Thus, we design this policy which samples inversely to usage frequency for exploration across tasks from the distribution $p_i = \frac{(1-n_i/N)}{(N-1)}$, where p_i, n_i are the sampling probability and count for task i and N is the total steps.

(3.2) Learning the Optimal Policy: Having described the behavior policy, we now explain our algorithm. As seen, we adopt an off-policy method in which the states and actions are not fixed. We adopt a Monte Carlo (MC) based learning since we obtain feedback at the end of the trajectory. MC methods are known to be unbiased, stable and avoid spurious noisy rewards. Specifically, the method can be described as an off-policy MC learning with environment expansion. The target policy $\pi(a|s)$ (for

inference) greedily selects structures. Formally,

$$\pi(a|s) = \begin{cases} 1 & \text{if } a = \arg \max_{a'} Q(s, a') \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The behavior policy combines the exploration strategies as follows:

$$\mu(a|s) = w_Q \pi_Q(a|s) + w_L \pi_L(a|s) + w_D \pi_D(a|s)$$

The Q-values are updated using importance sampling, on every visit as per the below expression:

$$Q_{t+1}(s, a) \leftarrow Q_t(s, a) + \frac{W_{t+1}}{C_t(s, a)} (G_t - Q_t(s, a))$$

$$G_{t+1} = \gamma G_t + R_{t+1}, \quad W_{t+1} = \frac{W_t}{\mu(a|s)}$$

where $Q \in \mathcal{R}^{|\mathcal{T}| \times |\mathcal{F}|}$ is the Q table, W is the importance sampling ratio, R_{t+1} is the reward at time t for taking action $a \in \mathcal{R}^{\mathcal{F}}$ (message structure) in state $s \in \mathcal{R}^{\mathcal{T}}$ (task), $C_t(s, a)$ is a normalizing factor (generally count of the times s, a is visited).

In the above, the tasks (state) and structures (action) could evolve as the learning progresses. Since new state-action pairs are discovered during learning, we employ the below two phases in learning:

Expansion Phase: Allow discovery of new (s, a) pairs up to threshold α fraction of interactions. In this phase the LLM generates new task categories and structures as seen previously.

Convergence Phase: Once new tasks and formats become rare, we fix the state-action space and allow for Q-value convergence.

The system learns from the complete multi-agent conversation trajectories of length L_T ending with outcome $R \in \{0, 1\}$. Algorithm 1 summarizes the method. The complexity of the method is $\mathcal{O}((|\mathcal{T}| + |\mathcal{F}|)|\mathcal{D}|L_T)$ (cf. § A.7), which is linear in number of tasks (\mathcal{T}), formats (\mathcal{F}), dataset size (\mathcal{D}) and trajectory length (L_T).

(3.3) Inferring Representations: At inference, the policy module takes a task (given by the task categorization module) and decides the structure in which to format the message for downstream communication. We adopt a greedy policy (eq. 1) that selects the best action $\pi^*(a|s)$.

5 Experiment Setup

We evaluate our method OPTiMACS, following the approach of the main baseline (Chen et al., 2024). Specifically, we seek to answer the following research questions: (i) **RQ1:** Does OPTiMACS

Algorithm 1 OPTiMACS Algorithm

```

1: Initialize, for all  $s \in \mathcal{T}, a \in \mathcal{F}(s)$ :
2:    $Q(s, a) \leftarrow$  arbitrary
3:    $C(s, a) \leftarrow 0$ 
4:    $\mu(a|s) \leftarrow w_Q \pi_Q(a|s) + w_L \pi_L(a|s) + w_D \pi_D(a|s)$  (behavior policy)
5:    $\pi(s) \leftarrow$  a deterministic policy that is greedy wrt  $Q$ 
6: repeat
7:   msg  $\leftarrow$  Sample from the set of queries/problems
8:   Generate an episode using behavior policy  $\mu(a|s)$ :
9:   repeat
10:     $\mathcal{T}_t \leftarrow$  TaskCategorization(msg, context)
11:     $\mathcal{F}_t \leftarrow \mu(\mathcal{T}_t, context)$ 
12:    prompt  $\leftarrow$  ApplyStructure(msg,  $\mathcal{F}_t$ )
13:    msg,  $R_{t+1} \leftarrow \mathcal{A}_t(\text{prompt})$ 
14:    if Expansion Phase then
15:       $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{T}_t\}, \mathcal{F} \leftarrow \mathcal{F} \cup \{\mathcal{F}_t\}$ 
16:    end if
17:     $t \leftarrow t + 1$ 
18:  until Episode end
19:   $G \leftarrow 0$ 
20:   $W \leftarrow 1$ 
21:  for  $t = T - 1, T - 2, \dots$  down to 0 do
22:     $G \leftarrow \gamma G + R_{t+1}$ 
23:     $C(\mathcal{T}_t, \mathcal{F}_t) \leftarrow C(\mathcal{T}_t, \mathcal{F}_t) + W$ 
24:     $Q(\mathcal{T}_t, \mathcal{F}_t) \leftarrow Q(\mathcal{T}_t, \mathcal{F}_t) + \frac{W}{C(\mathcal{T}_t, \mathcal{F}_t)} [G - Q(\mathcal{T}_t, \mathcal{F}_t)]$ 
25:     $\pi(\mathcal{F}_t|\mathcal{T}_t) \leftarrow \begin{cases} 1 & \text{if } \mathcal{F}_t = \arg \max_a Q(\mathcal{T}_t, a) \\ 0 & \text{else (with ties broken arbitrarily)} \end{cases}$ 
26:     $W \leftarrow W \cdot \frac{\pi(\mathcal{F}_t|\mathcal{T}_t)}{\mu(\mathcal{F}_t|\mathcal{T}_t)}$ 
27:    if  $W = 0$  then
28:      Exit For Loop
29:    end if
30:  end for
31: until Convergence or max iterations

```

enhance the performance over static and fixed structures in multi-agent settings? (ii) **RQ2:** Do we need to optimize for representations or are there fixed structures that work across datasets? (iii) **RQ3:** How is the efficiency of the system affected? (iv) **RQ4:** What is the effectiveness of each component? (v) **RQ5:** What is the distribution of the dynamically evolving tasks and representations? We detail the experiment setup below.

5.1 Datasets:

Single LLM Reasoning: In the motivation study (cf. fig. 1), we employ the datasets of Logic Grid, Coin Flip and AQuA adopting the same experiment setting as (Chen et al., 2024).

Multi-Agent: (i) Collaborative Problem Solving: We use the GSM+ (GSM) benchmark (Li et al., 2024) to evaluate mathematical problem solving, as the adversarial examples in the dataset could benefit from a multi-agent setup and various message structures. Correctness is measured by matching answers up to precision. **(ii) Information Exchange:** In order to evaluate the ability to find optimal structures, we experiment with information exchange using multi-hop QA using the NarrativeQA (NQA) (Kočický et al., 2018), WikiHopQA (WQA) (Welbl et al., 2018), and HotpotQA (HQA) (Yang et al., 2018) datasets. Correctness is evaluated as follows: (i) when answer choices are provided, we use exact match, (ii) for open-ended responses, we employ an NLI-based metric where we use a RoBERTa model fine-tuned for NLI (Tarunesh et al., 2021), treating the question and model-generated answer as the “premise,” and the annotation as the “hypothesis,” scoring correct if entailment score > 0.7 (set empirically).

5.2 Baselines:

For all the experiments, we choose the vanilla multi-agent system (Vanilla MAS) and Autoform (prompts LLM for best structure from predefined ones) as the baselines. Please note that the Vanilla MAS already uses prompting techniques like CoT (Wei et al., 2023), ReAct (Yao et al., 2023) etc. and our method is orthogonal to these. Since these methods involve directly prompting the LLMs and there is no learning involved, we directly evaluate on the test-set. For OPTiMACS, we first learn the policy by running trajectories over a small sample of data points ($\sim 500 - 2000$). The testing set is obtained keeping a similar setting as baseline Autoform (Chen et al., 2024). In order to gauge the benefits of dynamically learning the message structures from data/environment, we experiment with LLMs such as GPT4o (OpenAI, 2024) and O3 (OpenAI, 2025). To further understand the effect of learning structures on smaller models, we also experiment with some small language models such as Phi4-mini (Abdin et al., 2024), llama3-8B (et al., 2024) and Qwen2.5-Math-7B (Team, 2024).

Table 1: OPTiMACS outperforms the vanilla MAS using NL and Autoform (fixed formats by the LLM).

LLM	Method	GSM	WQA	HQA	NQA	Avg
GPT-4o	Vanilla	86.5	19.5	65.1	51.3	55.6
	Autoform	87.5	18.2	48.5	35.9	47.5
	OPTiMACS	89.5	21.4	69.1	57.2	59.3
o3	Vanilla	93.5	20.5	80.2	70.5	66.2
	Autoform	92.1	21.9	81.7	59.8	63.9
	OPTiMACS	94.7	22.8	88.9	71.9	69.6
Qwen2.5-7B (Math)	Vanilla	79.1	11.8	29.3	26.2	36.6
	OPTiMACS	82.5	17.2	37.4	29.7	41.7
Phi4-mini	Vanilla	64.5	6.3	24.5	22.4	29.4
	OPTiMACS	70.5	11.6	31.8	23.9	34.4
Llama3-8B	Vanilla	77.9	9.7	28.3	25.1	35.2
	OPTiMACS	83.5	12.8	33.6	27.2	39.3
Efficiency	Δ Tokens (%)	-8.7	-5.4	-18.8	19.3	-3.4

5.3 Implementation details

We design multi-agent scenarios with three agents - Planner, Verifier, and Coder - for GSM+ dataset and five agents (each with different context/information) for the multi-hop setting, using the AutoGen framework (Wu et al., 2023). For the policy learning, we initialize the Q values for newly discovered structures and sampling temperature as 0. We allow the trajectories to roll out for a maximum of 25 steps (as we empirically find the goal is achieved by then). For the behavior policy, we set the weights as $w_Q = 0.5$, $w_L = 0.25$, $w_D = 0.25$. We set the $\epsilon = 0.1$. The expansion phase is allowed for around 20% of the epoch after which we allow the Q-values to converge. The trajectory is terminated when any agent thinks the goal is achieved or the maximum number of iterations (25) have been reached. If the goal/answer matches the ground truth a reward of +1 is provided else a penalty of -1 is incurred, for every step. All experiments are conducted on a linux machine with a single A100 GPU. For SLM-inference, we use the vLLM library (Kwon et al., 2023) and official open source models as on huggingface (Hugging Face, 2025). Each experiment is run three times and we report the average over runs for brevity (variance $\sim \pm 2\%$).

6 Results

6.1 Main Results (RQ1)

In this section, we present the results of our method w.r.t. baselines on the benchmark datasets. Table 1 shows the accuracy results on GSMPlus and the MultiHop datasets, against the respective LLMs. We observe that our method outperforms the baselines and the enhancements are statistically significant (using Wilcoxon signed-rank test, p-value

< 0.05). This indicates that learning the optimal structures dynamically enhances performance in multi-agent setting, answering RQ1. Where Roberta based NLI is used for evaluation, we perform a human evaluation on a sample (~ 100) set and find a high Cohen’s Kappa (Cohen et al., 2007) ($>90\%$). We also find our method (vanilla), on average, needs around 3.5 (3.5), 2.5 (3) hops using GPT4o, o3 respectively. From the SLM results, we can observe that our method demonstrates consistent improvements compared to the vanilla method, confirming the benefit of communicating using optimal message structures even in small models. We further report the percent difference of tokens consumed by our method (w.r.t. vanilla system) across methods and datasets and find that the communication cost is on par or even reduced in some cases.

Task Classification: One of the important modules in our method is the task classification module, where the LLM samples a state from its observation (input message). Here we quantify the accuracy of the task classification, by a double-blind human review. We employ two human experts to label the tasks (~ 100 per dataset) as classified by the module. The accuracy (across datasets) was 99.5% and the Cohen’s Kappa (Cohen et al., 2007) was near 100% showing complete inter-rater agreement.

6.2 Performance with Fixed Structures (RQ2)

In this study, we analyze if finding the optimal structure is necessary by comparing with some largely adopted fixed structures in multi-agent communication literature. From table 2 we can see that among fixed methods, Pseudo-code & List (Sequential Thinking) generally perform well due to the mathematical nature of the tasks. However, there exists no structure that performs consistently better across all datasets. This shows that the best structure is task-dependent, answering RQ2. Further, our method performs better than the fixed structures indicating that OPTiMACS attempts to find the optimal structure for a task. We also report average tokens and time per message and across datasets. We find that alternative formats (e.g. KQML) have higher efficiency than NL and OPTiMACS is able to maintain high accuracy while also finding formats that have low token count and latency.

6.3 Token Efficiency (RQ3)

In this study, we observe how efficient our method is compared to the vanilla multi-agent system without optimizing the message structures. This would

Table 2: OPTiMACS outperforms fixed representations indicating the importance of dynamic learning.

Structure	GSM	WQA	HQA	NQA	Avg	Avg Tokens	Avg Time (s)
NL	86.5	19.5	65.1	51.3	55.6	87.8	1.58
Ours	89.5	21.4	69.1	57.2	59.3	79.8	1.23
JSON	71.5	19.5	66.9	52.9	52.7	101	1.5
XML	89.1	17.1	68.1	52.2	56.6	99.8	1.48
KQML	89.5	14.6	63.9	<u>57.1</u>	56.3	70.5	1.28
YML	87.6	18.3	66.8	50.0	55.7	77.3	1.20
Code	<u>89.2</u>	<u>21.1</u>	68.0	54.0	58.1	201.7	2.41
Equation	87.9	18.3	<u>69.0</u>	47.4	55.7	88.8	1.58
Table	87.0	14.6	62.2	53.1	54.2	117.3	1.54
List	88.1	20.8	67.9	56.3	<u>58.3</u>	106.1	1.34

help inform practitioners of the tradeoffs in tokens/-cost with enhanced performance. We observe from table 3 that across all the datasets on which we experimented, the maximum increase in token usage is not more than 20%. In some cases OPTiMACS is more efficient compared to the vanilla multi-agent system. This is because certain formal structures require fewer tokens compared to natural language. Thus, OPTiMACS enhances performance while keeping the cost bounded, answering RQ3.

Table 3: Token efficiency of OPTiMACS.

Dataset	#Tokens (Vanilla)	#Tokens (Ours)	Δ Tokens (%)
GSM+	2667	2434	-8.7
WikiHop	1023	968	-5.4
HotPotQA	671	545	-18.8
NarrativeQA	925	1104	+19.3

6.4 Ablation Study (RQ4)

We study the impact of fixed set of structures/tasks (w/o dynamic) and dynamic but not learning the policy from the data (w/o learning). We further study how the performance evolves with converging Q values as the learning progresses. From figure 4 we find that the results drop without the policy learning ($\sim 42\%$) or dynamically evolving modules. On the WikiHop dataset, we find the drop is drastic ($\sim 42\%$) without the policy learning module, whereas in the HotpotQA dataset we find that dynamically learning structures is more important to the task. Thus, the importance of the modules is task/dataset dependent; nevertheless, both modules are important in OPTiMACS to achieve optimal performance. From the Q-plot we can see that the Q-values/accuracy converge with time. Further, the average slope ($\Delta Q/\text{step}$) in the expansion phase is 3.08×10^{-2} vs 1.5×10^{-3} in the convergence phase, indicating convergence once the EMDP is frozen (supporting prop. 1). Further studies are provided in the Appendix (A.4.3, A.5).

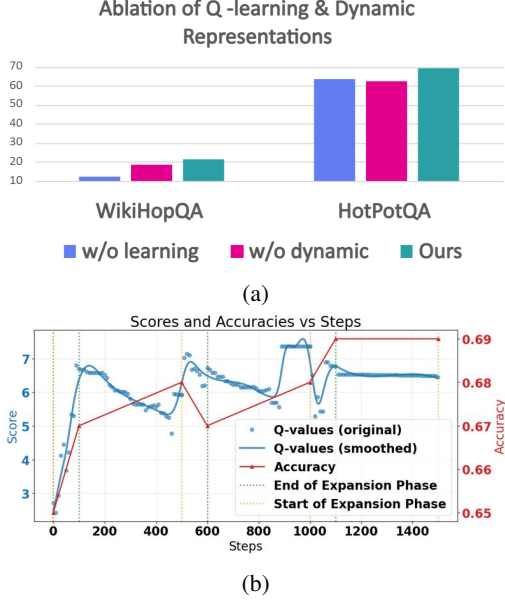


Figure 4: a) Dynamic representation evolving and policy learning module are important for performance b) The Q-values/accuracy converge as learning progresses.

6.5 Structure Distribution (RQ5)

In this section, we study the distribution of the tasks and structures uncovered by OPTiMACS during the expansion phase of learning. In figure 5, we see the structure distribution where we display 10/(64) high-level structures including seed and discovered structures. In addition to the seeded structures of JSON, XML, YAML, List, Code etc., our method is able to discover some new structures ($\sim 50\%$) within the categories of Trees, Graphs, Math etc. From the task wise distribution of representations, we see the hard category of the comparison task benefits from “Math” representation due to the nature of the task, whereas the bridge task benefits from Graph and Tree structures as expected. Thus OPTiMACS is able to select the pertinent structure for the task. For brevity, we show evolution of formats in the Appendix A.4.2.

7 Conclusion

In this paper, we present a method to dynamically learn the optimal structure for effective communication in a multi-agent system. We pose the problem of finding the optimal message structure as an Expanding Markov Decision Process (EMDP). This allows us to dynamically evolve to new representations as the agents interact with the environment. We further learn the optimal policy to decide the structure for the task at hand, rather than relying

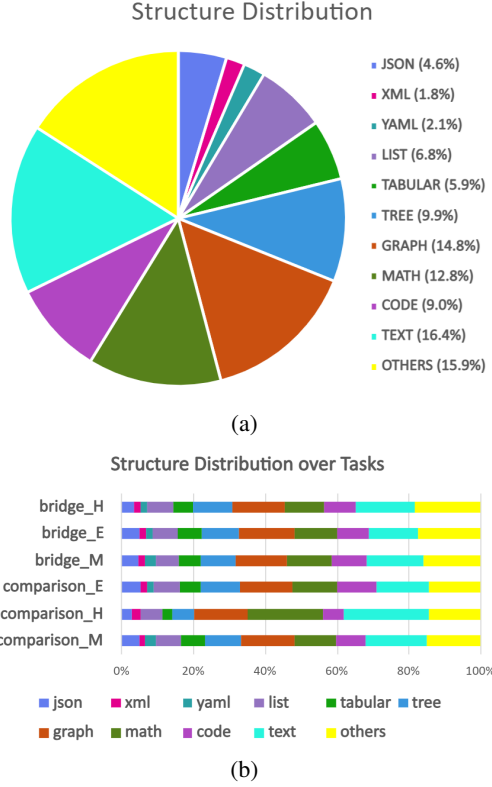


Figure 5: Fig. a) shows the overall structure distribution and b) is the task-wise distribution found for HotPotQA.

on fixed structures or the inductive biases from the LLM. Our work is the first to provide a learnable framework for optimizing message representations in agentic (LLM) communication. Future works could extend this research by employing composite optimal representations and latent search spaces to expand beyond finite tabular methods to handle infinite tasks and representations.

8 Limitations

While OPTiMACS demonstrates strong empirical results on standard benchmarks, we encourage practitioners to apply the techniques proposed in the paper to their real-world scenarios to see whether structured representations benefit the task. In our study, we optimize the policy for trajectories sampled from a given dataset. However, it remains to be studied whether this policy can be transferred to another related dataset. While we observe improvements in task performance, we also note slight increase in token usage in certain cases and future works can look at prompt optimization to further enhance efficiency. Future research could also explore a multi-dimensional abstraction of tasks.

References

- Marah Abidin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. [Phi-4 technical report](#). *Preprint*, arXiv:2412.08905.
- Saeed Afshar, Ying Xu, Jonathan Tapson, André van Schaik, and Gregory Cohen. 2019. [Event-based feature extraction using adaptive selection thresholds](#). *Preprint*, arXiv:1907.07853.
- Anthropic. 2024. Model context protocol. *Technical Report*.
- Jon Barwise and John Perry. 1983. *Situations and attitudes*. MIT Press.
- R. Bellman, R.E. Bellman, and Rand Corporation. 1957. [Dynamic Programming](#). Rand Corporation research study. Princeton University Press.
- Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific American*, 284(5):34–43.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). *Preprint*, arXiv:2005.14165.
- Weize Chen, Chenfei Yuan, Jiarui Yuan, Yusheng Su, Chen Qian, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2024. [Beyond natural language: LLMs leveraging alternative formats for enhanced reasoning and communication](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10626–10641, Miami, Florida, USA. Association for Computational Linguistics.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Sarah Cohen, Werner Nutt, and Yehoshua Sagiv. 2007. [Deciding equivalences among conjunctive aggregate queries](#). *Journal of the ACM*, 54(2).
- ACP Consortium. 2024a. Agent communication protocol. *Technical Specification*.
- ANP Consortium. 2024b. Agent network protocol. *Technical Specification*.
- Thomas M Cover and Joy A Thomas. 1999. *Elements of information theory*. John Wiley & Sons.
- Gianpaolo Cugola and Alessandro Margara. 2012. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):1–62.
- Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. [Everything of thoughts: Defying the law of penrose triangle for thought generation](#). *Preprint*, arXiv:2311.04254.
- Aaron Grattafiori et. al. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. 1994. Kqml as an agent communication language. In *Proceedings of the third international conference on Information and knowledge management*, pages 456–463. ACM.
- FIPA. 1997. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents*.
- Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*.
- Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. 2024. [Magenti-one: A generalist multi-agent system for solving complex tasks](#). *Preprint*, arXiv:2411.04468.
- Sigmund Freud. 1989. The ego and the id (1923). *Tacd Journal*, 17(1):5–22.
- Google. 2024. [Google whitepaper on ai agents](#). Technical report, Google. Whitepaper, accessed 17 Jul 2025.
- Google. 2025. [Dive into google’s agent development kit \(adk\) to build production-ready ai agents](#). Published July 2025.
- A2A Working Group. 2024. Agent-to-agent protocol. *Technical Specification*.
- Yuchen Han, Yucheng Wu, and Jeffrey Willard. 2025. [Effect of selection format on llm performance](#). *Preprint*, arXiv:2503.06926.
- Hugging Face. 2025. Hugging face. <https://huggingface.co/>.
- Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205.

- Hans Kamp and Uwe Reyle. 1993. *From discourse to logic*. Springer.
- Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. [The narrativeqa reading comprehension challenge](#). *Transactions of the Association for Computational Linguistics*, 6:317–328.
- Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. *arXiv preprint arXiv:2010.05700*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with pagedattention](#). *Preprint*, arXiv:2309.06180.
- Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *arXiv preprint arXiv:2303.17760*.
- Juncen Li, Robin Jia, He He, and Percy Liang. 2018. [Delete, retrieve, generate: A simple approach to sentiment and style transfer](#). *Preprint*, arXiv:1804.06437.
- Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. [GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.
- Fangru Lin, Emanuele La Malfa, Valentin Hofmann, Elle Michelle Yang, Anthony G. Cohn, and Janet B. Pierrehumbert. 2024. Graph-enhanced large language models in asynchronous plan reasoning. In *Proceedings of the 41st International Conference on Machine Learning, ICML'24*. JMLR.org.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Yuanye Liu, Jiahang Xu, Li Lina Zhang, Qi Chen, Xuan Feng, Yang Chen, Zhongxin Guo, Yuqing Yang, and Peng Cheng. 2024a. Beyond prompt content: Enhancing llm performance via content-format integrated prompt optimization. *arXiv preprint arXiv:2402.04295*.
- Ziheng Liu, Jiayi Zhang, Enyu Shi, Zhilong Liu, Dusit Niyato, Bo Ai, and Xuemin Shen. 2024b. [Graph neural network meets multi-agent reinforcement learning: Fundamentals, applications, and future directions](#). *Preprint*, arXiv:2404.04898.
- Dejan Milošević, Markus Breugst, Ingo Busse, John Campbell, Stefan Covaci, Benny Friedman, Kazuhiko Kosaka, Danny Lange, Kouji Ono, Masahiro Oshima, and 1 others. 1999. Masif: The omg mobile agent system interoperability facility. *Personal Technologies*, 2(2):117–129.
- Richard Montague. 1974. *Formal philosophy: selected papers of Richard Montague*. Yale University Press.
- J. C. Olamendy. 2023. Understanding early stopping: A key to preventing overfitting in machine learning. *Medium*.
- OpenAI. 2024. Gpt-4o: Omni-modal large language model. *OpenAI Technical Report*.
- OpenAI. 2025. [Competitive programming with large reasoning models](#). *Preprint*, arXiv:2502.06807.
- Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.
- John R Pierce. 1992. *An introduction to information theory*. Dover Publications.
- Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York.
- Yujia Qin, Shihao Liang, Yining Ye, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.
- Claude E Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: Language agents with verbal reinforcement learning](#). *Preprint*, arXiv:2303.11366. NeurIPS 2023. URL: <https://github.com/noahshinn024/reflexion>.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*.

- Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, 2 edition. MIT Press. See Figure 5.10, p. 137: Off-policy every-visit MC control algorithm using weighted importance sampling.
- Rinyoichi Takezoe, Yaqian Li, Zi-Hao Bo, Anzhou Hou, Mo Guang, and Kaiwen Long. 2026. [Learnpruner: Rethinking attention-based token pruning in vision language models](#). In *The Fourteenth International Conference on Learning Representations*.
- Ishan Tarunesh, Somak Aditya, and Monojit Choudhury. 2021. [Trusting roberta over bert: Insights from checklisting the natural language inference task](#). *Preprint*, arXiv:2107.07229.
- Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Y. Tian and et al. 2024. Can LLMs reason in the wild with programs? *arXiv preprint arXiv:2406.13764*.
- Christopher J.C.H. Watkins and Peter Dayan. 1992. [Q-learning](#). *Machine Learning*, 8(3-4):279–292.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. [Chain-of-thought prompting elicits reasoning in large language models](#). *Preprint*, arXiv:2201.11903.
- Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. [Constructing datasets for multi-hop reading comprehension across documents](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2416–2426.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- K. Wu, E. Wu, and J. Zou. 2024. How faithful are rag models? quantifying the tug-of-war between rag and llms’ internal prior. *arXiv preprint arXiv:2404.10198*.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [Hotpotqa: A dataset for diverse, explainable multi-hop question answering](#). *Preprint*, arXiv:1809.09600.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). *Preprint*, arXiv:2210.03629.
- Susan Zhang and et. al. 2023. Emergent abilities of large language models. *Transactions on Machine Learning Research*.
- Lei Zhao, Wenyang Yu, Zhibin Zheng, Yifan Wang, Yifan Feng, Dapeng Tan, and Enjie Yu. 2023. A survey of large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

A Appendix

In this supplementary, we provide results and discussion that could not be included in the main paper due to space constraints. Specifically, we elaborate on some of the preliminaries in §A.0.1 followed by some additional results and analysis. The additional details are:

1. Preliminaries (§A.0.1) and justification of MDP setting over Contextual bandits (§A.1)
2. Theory of convergence of the proposed EMDP learning (§A.2)
3. Related works (§A.3)
4. Details of experiments in main paper (§A.4)
5. Further model ablation (§A.5)
6. Example case studies (§A.6) to show the flow of communication between the agents in different datasets
7. Computational complexity (§A.7)
8. Distribution of Structures (§A.8)
9. Novelty and Diversity of Structures/Formats Detected (§A.9)
10. Style Transfer Metrics (§A.10): to study the loss of information from natural language to structured representations and vice versa
11. Prompts used in the LLMs (§A.11)
12. Design of the single agent setting (§A.12)

A.0.1 Preliminaries

A.0.2 Markov Decision Process (MDP)

A Markov Decision Process (MDP) provides a mathematical framework for modeling decision-making in environments where outcomes are partly random and partly under the control of an agent. Formally, an MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- \mathcal{S} is the set of states,
- \mathcal{A} is the set of actions,
- $P(s'|s, a)$ is the transition probability function, representing the probability of moving to state s' from state s after taking action a ,

- $R(s, a)$ is the reward function, giving the expected immediate reward received after taking action a in state s ,
- $\gamma \in [0, 1]$ is the discount factor, which balances the importance of immediate and future rewards.

The objective in an MDP is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected cumulative discounted reward, defined as:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s \right],$$

where $V^\pi(s)$ is the value function under policy π . MDPs assume the *Markov property*, meaning the future state depends only on the current state and action, not on the sequence of events that preceded it. This framework underpins many reinforcement learning algorithms and is foundational to understanding agent-environment interactions (Puterman, 1994).

A.0.3 Policy Learning (Q-learning)

Q-learning is a model-free reinforcement learning algorithm that aims to learn the optimal action-selection policy by estimating the action-value function, denoted as $Q(s, a)$. The action-value function represents the expected cumulative discounted reward of taking action a in state s and following the optimal policy thereafter. The core idea of Q-learning is to iteratively update the Q-values using the Bellman equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

where $\alpha \in (0, 1]$ is the learning rate, $\gamma \in [0, 1]$ is the discount factor, r_t is the reward received after taking action a_t in state s_t , and s_{t+1} is the resulting next state. Over time, and under certain conditions such as sufficient exploration and decaying learning rate, Q-learning converges to the optimal action-value function $Q^*(s, a)$, enabling the derivation of the optimal policy $\pi^*(s) = \arg \max_a Q^*(s, a)$ (Watkins and Dayan, 1992).

A.0.4 Monte Carlo Learning

The Monte Carlo (MC) method for Q-learning, is a model-free approach that estimates the action-value function $Q(s, a)$ using complete episodes of experience. Unlike temporal-difference methods,

Monte Carlo methods update Q-values only at the end of an episode, using the actual returns observed. For each state-action pair (s, a) encountered in an episode, the Q-value is updated as:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [G_t - Q(s, a)],$$

where α is the learning rate and $G_t = \sum_{k=t}^T \gamma^{k-t} r_k$ is the return from time step t until the end of the episode T . Monte Carlo methods require episodes to terminate and assume that the environment is episodic. They are particularly useful when the model dynamics are unknown and when bootstrapping is not desired. While slower to converge than TD methods, MC-based Q-learning can provide unbiased estimates of the expected return under sufficient exploration (Sutton and Barto, 2018).

A.1 Justification for MDP based setting

We briefly formally introduce the key concepts used in our framework.

Policy (π): A policy is a strategy, mapping states (contexts) to actions (decisions), denoted as $\pi(a|s)$, representing the probability of taking action a in state s .

A/B Testing: A randomized control experiment used to compare two versions of a variable (e.g., prompts) to determine which performs better on average across a dataset. This can be used for example if we have much offline data to test which representation generally works well across a dataset. However it requires much offline evaluation and may not be practical in an online setting. Further it lacks the ability to make fine-grained decisions for example if there are different tasks in a dataset where different representations would be beneficial.

Contextual Bandits: A simplified reinforcement learning setting where an agent observes a context, selects an action, and receives an immediate reward, without state transitions. We use this validation for single-step prompting where the action (format) does not impact future states. However in a multi-agent setting where the output of one agent feeds into another sequentially we would need to track the history of states and in this scenario the contextual bandit would not be effective.

Markov Decision Processes (MDPs): A framework for sequential decision-making characterized

by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where \mathcal{S} is the state space, \mathcal{A} is the action space, \mathcal{P} defines transition probabilities between states, \mathcal{R} is the reward function, and γ is theoretically the discount factor. We employ this for multi-agent coordination where current decisions fundamentally alter the dialogue trajectory.

Q-values ($Q(s, a)$): The expected cumulative future reward an agent will receive by taking action a in state s and following a specific policy thereafter.

Off-Policy Learning: An off-policy algorithm learns the value of a target policy π (e.g., optimal formats) while following a different behavior policy μ for exploration. Both our algorithms are off-policy as they separate the learning of optimal strategies from the diverse exploration required to discover them.

A.1.1 Formulation for 1-step LLM Calls

Here we formalize the problem of LLM prompting and justify several key design choices as well as need for Reinforcement Learning (RL). While A/B testing can identify a globally superior static prompt, it fails to capture instance-specific preferences where different queries require different formats. A contextual-bandit formulation addresses this by learning a policy that maps specific contexts to optimal actions. Our formulation aligns with the standard Contextual Bandit framework—mapping contexts to actions to maximize reward—but is distinguished by two key differences: (1) **Tabular State Space:** Unlike vector-based methods (e.g., LinUCB) that generalize across infinite states, we discretize the context into semantic task categories, defining valid clusters as proposed in Autoform (Chen et al., 2024) and program-based reasoning studies (Tian and et al., 2024); and (2) **Ensemble Exploration:** We utilize a union of diverse strategies rather than a single policy. The benefit of (1) is interpretable, data-efficient learning that avoids complex function approximation, while (2) ensures robust coverage of the search space.

A.2 Convergence of Proposed EMDP

In this section, we discuss and prove the convergence of the proposed EMDP for learning optimal message representations. First, we state the known convergence of the standard MDP.

Lemma 1 (Convergence of Value Iteration for Finite MDPs (Bellman et al., 1957)). *Let $M =$*

(S, A, P, R, γ) be a finite Markov Decision Process, where:

- S is a finite set of states,
- A is a finite set of actions,
- $P(s'|s, a)$ is the transition probability,
- $R(s, a)$ is a bounded reward function,
- $0 < \gamma < 1$ is the discount factor.

Define the value iteration operator T as:

$$(TV)(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a)V(s') \right]$$

for any value function V .

Then, starting from any initial value function V_0 , the sequence $V_{k+1} = TV_k$ converges uniformly to the unique fixed point V^* , which is the optimal value function for the MDP:

$$\lim_{k \rightarrow \infty} \|V_k - V^*\|_{\infty} = 0$$

Moreover, the greedy policy with respect to V^* is an optimal policy.

In our setting the states and actions are not static but keep evolving with time. Moreover the state space is not known but the agent only observes the environment and maintains a belief \mathcal{B} which is the set of all probability distributions over the state space (S). While sampling the state from the belief, we convert the belief to a dirac distribution (1 at sampled state and 0 elsewhere) which simplifies the problem to the standard MDP. However, since the states and action spaces are evolving (evolving phase) we need to allow for some iterations for convergence of sampled states and actions. This happens in the cooling phase of our algorithm. Moreover the LLM itself could update its belief if the parameters change. However since we use a pretrained LLM we relax the assumption that the LLM's belief is fixed. To reiterate, we operate under the following mild assumptions:

1. The state (task) sampling prompt uses a temperature of 0 to sample the best task for the concerned message as per the LLM's belief
2. The LLM's parameters (and thus belief given observation) are kept fixed during learning and inference i.e. we use a pretrained LLM

Having seen the assumptions we are now ready to look at the proof of the convergence of our EMDP based formulation.

Proposition 2 (Convergence of the proposed EMDP based algorithm). *Consider the EMDP = $(S_e, A_e, P_e, R, \gamma)$, with value function V . S_e, A_e are the evolving states and actions respectively. Let T denote the update operator s.t. $V_{k+1} = T(V_k)$, then $\lim_{k \rightarrow \infty} \|V_k - V^*\| = 0$, where V^* is the optimal value function for the states and actions explored.*

Proof. Let \mathcal{P} be a finite POMDP with state space S , action space A , observation space O , bounded reward function R , and discount factor $0 < \gamma < 1$. Let V_0 be any initial value function over the belief space \mathcal{B} (the set of all probability distributions over S). Define the value iteration operator T as:

$$(TV)(b) = \max_{a \in A} \left[\sum_{s \in S} R(s, a)b(s) + \gamma \sum_{o \in O} P(o|b, a)V(\tau(b, a, o)) \right]$$

where b is a belief state and $\tau(b, a, o)$ is the updated belief after action a and observation o .

Now, the LLM samples the state (task) from its belief. Assuming the state with maximum probability is sampled, we have

$$b^*(s) = \begin{cases} 1 & \text{if } s = \underset{s}{\operatorname{argmax}} b(s) \\ 0 & \text{otherwise} \end{cases}$$

This gives us the operator (for the cooling phase

where states and actions are fixed),

$$\begin{aligned}
(TV)(b) &= \max_{a \in A} \left[\sum_{s \in S} R(s, a) b(s) \right. \\
&\quad \left. + \gamma \sum_{o \in O} P(o|b, a) V(\tau(b, a, o)) \right] \\
&= \max_{a \in A} \left[\sum_{s \in S} R(s, a) b^*(s) \right. \\
&\quad \left. + \gamma \sum_{o \in O} P(o|b, a) V(\tau(b, a, o)) \right] \\
&= \max_{a \in A} \left[R(s^*, a) + \gamma \sum_{o \in O} P(o|b, a) V(\tau(b, a, o)) \right] \\
&= \max_{a \in A} \left[R(s^*, a) + \gamma \sum_{s' \in S} P(s' | s^*, a) V(s') \right] \\
&\quad \dots \text{injective since observation maps to a belief} \\
&\quad \text{from which argmax state is sampled} \\
(TV)(s^*) &= \max_{a \in A} \left[R(s^*, a) + \gamma \sum_{s' \in S} P(s' | s^*, a) V(s') \right]
\end{aligned}$$

This converts to the standard MDP, which from lemma 1 converges to optimal value function V^* (and thus policy) i.e. the sequence $V_{k+1} = TV_k$ converges to V^* which is the optimal value function for the EMDP (till this point), as $k \rightarrow \infty$: $\lim_{k \rightarrow \infty} \|V_k - V^*\|_\infty = 0$. After every expansion finding new states and actions, we could follow this procedure to guarantee convergence. \square

Consider a phase p_i of an EMDP (S_i, A_i, P_i, R_i) where we perform the expansion of states and actions and allow convergence before exploring new states and actions. Now in the new phase p_{i+1} 's expansion, let's say the agent has discovered new states ΔS and actions ΔA . The above proposition essentially states that at every phase, the learning of the EMDP $\Delta S + S_i, \Delta A + A_i, P_{i+1}, R_{i+1}$ converges.

A.3 Related Works

A.3.1 Multi-Agent Systems (MAS)

Recent advances in MAS have evolved from single-shot LLM interactions to structured communication (Zhao et al., 2023). The framework proposed in (Google, 2024, 2025) formalizes the model-orchestration-tools stack with persistent memory systems. (Fourney et al., 2024) demonstrates practical coordination through specialized agents managed by a central Orchestrator. However, existing frameworks focus primarily on agent coordination

mechanisms rather than optimizing communication structures between agents.

A.3.2 Multi-Agent Communication Protocols

Traditional agent communication has relied on standardized protocols like KQML (Finin et al., 1994), FIPA-ACL (FIPA, 1997), MASIF (Milojicic et al., 1999), Web Services (Berners-Lee et al., 2001), Event Stream Processing (Cugola and Margara, 2012) etc. In the era of LLMs, researchers have explored communication using function calling mechanisms (Schick et al., 2023) and tool usage (Schick et al., 2023). Recent standardization efforts include protocols such as MCP (Anthropic, 2024), ACP (Consortium, 2024a), A2A (Group, 2024) and ANP (Consortium, 2024b). While these establish communication infrastructure, they provide static, rule-based schemas that cannot adapt to varying task requirements or agent capabilities.

A.3.3 Learning in Multi-Agent Systems

Multi-Agent Reinforcement Learning (RL) has explored communication optimization through various approaches. Message pruning (Takezoe et al., 2026) and graph-based (Liu et al., 2024b) methods focus on reducing communication overhead and optimizing information flow between agents. Task-oriented communication protocols have been developed using RL to enable agents to discover efficient communication strategies between classical RL agents (Foerster et al., 2016). However, LLMs differ in linguistically mediated communication (Sumers et al., 2023). Existing multi-agent LLM frameworks leverage world knowledge for coordination through open-loop and closed-loop methods (Shinn et al., 2023), but do not adapt message structures to optimize task performance.

A.3.4 Structured Communication among LLM Agents

Recent work on structured communication focused on leveraging the inductive biases in LLM for format generation (Chen et al., 2022), though without mechanisms for dynamically learning or domain adaptation. These methods suffer from over-reliance on static LLM biases without adaptive learning mechanisms.

Our work differs from existing research in that, unlike existing approaches that focus on either agent coordination infrastructure or static / LLM prompted format generation, we focus on the optimization of message representations/structures/format

mats in multi-agent communication by *dynamically learning* from the environment.

A.4 Details of Experiments in Main Paper

A.4.1 Token Efficiency (RQ3)

In this study, we observe how efficient our method is compared to the vanilla multi-agent system without optimizing the message structures. This would help inform practitioners of the tradeoffs in tokens/cost with enhanced performance. We observe from table 4 that across all the datasets on which we experimented, the maximum increase in token usage is not more than 20%. In some cases OPTiMACS is more efficient compared to the vanilla multi-agent system. This is because certain formal structures require fewer tokens compared to natural language. Thus this experiment shows that OPTiMACS enhances performance while keeping the cost bounded.

Table 4: Token efficiency of OPTiMACS.

Dataset	#Tokens (Vanilla)	#Tokens (Ours)	Δ Tokens (%)
GSM+	2667	2434	-8.7
WikiHop	1023	968	-5.4
HotPotQA	671	545	-18.8
NarrativeQA	925	1104	+19.3

A.4.2 Evolution of Formats

Figure 6 illustrates quantitatively the discovery of new formats and task performance in OPTiMACS across datasets and LLMs. SA indicates single agent datasets and MA are the multi-agent datasets. Static denotes, no evolution of formats and dynamic denotes format evolution using our method. "Top" in the figure indicates the top 3 frequently used formats for the concerned dataset and setting. As training progresses, the bar plots show a steady and substantial increase in the cumulative number of newly discovered communication formats, indicating that agents continue to explore and adopt diverse structured representations rather than converging prematurely. In parallel, the line plots reveal consistent improvements in the performance metrics (success rate and normalized Q-values), reflecting gains in decision quality as training advances. Notably, phases with sharper increases in discovered formats (mid to late training) align closely with pronounced improvements in Q-values, suggesting that richer and more expressive format spaces directly contribute to higher-value action selection. We find that static formats (where there is no evolution of formats) have lower perfor-

mance in general compared to their evolving counterparts. For example in AQuA dataset we see the static method (first blue bar) has poor performance (10%) compared to its evolving counterpart (green bar) which achieves over 90% success rate (green dotted line with triangle marker). We also observe (for AQuA dataset) that the static formats predominantly prefer natural language whereas the evolving method learns new math representations that enhance the performance. While dataset-specific trends vary, the overall pattern remains consistent: continued format discovery correlates with sustained or improved task performance, demonstrating that OPTiMACS effectively balances exploration of novel communication formats with exploitation for higher downstream reward.

A.4.3 OPTiMACS Ablation Study (RQ4)

In this section, we perform an ablation study to analyze the importance of each component of OPTiMACS. Specifically, we study the impact of having a fixed set of structures/tasks and keeping the structures/tasks dynamic but not learning the policy from the environment/data. We further study how the performance evolves with converging Q values as the learning progresses. From figure 7 we find that the results drop without the policy learning or dynamically evolving modules. On the WikiHop dataset we find the drop is drastic ($\sim 42\%$) without the policy learning module whereas in the HotpotQA dataset we find that dynamically learning structures is more important to the task. Thus the importance of the modules is task/dataset dependent but nevertheless both the modules are important in OPTiMACS to achieve optimal performance. From the Q-plot we can see that the Q-values/accuracy converge with time.

A.4.4 Structure Distribution (RQ5)

In this section, we study the distribution of the tasks and structures uncovered by OPTiMACS during the expansion phase of learning. In figure 8 we see the structure distribution where we display 10 high-level structures including seed and discovered structures. In total the method discovered 64 structures and we club the fine-grained structures into these high level types. In addition to the seeded structures of JSON, XML, YAML, List, Code etc., our method is able to discover some new structures within the categories of Trees, Graphs, Math etc. The newly discovered structures account for more than 50% of the structures. We also analyze the

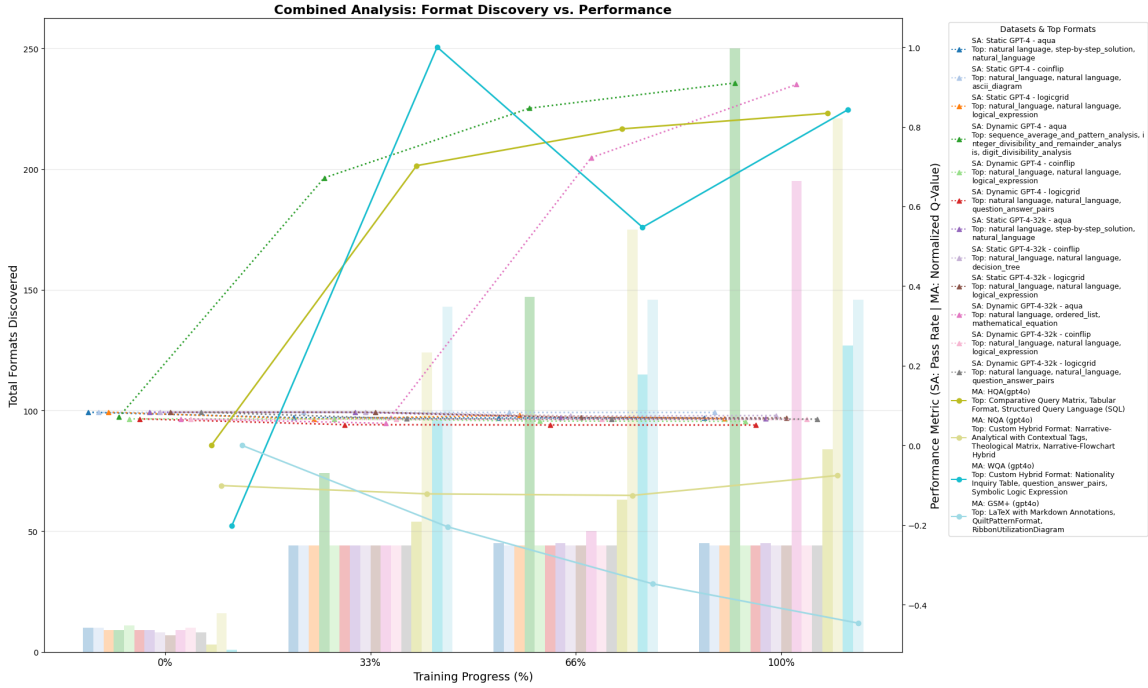


Figure 6: Bar plots show evolution of structures (formats) as the learning progresses. The color indicates different datasets and methods. The line plots show the performance (accuracy) and normalized Q-values.

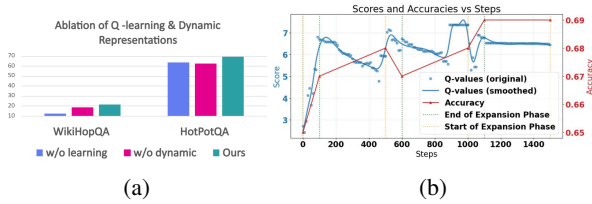


Figure 7: a) Dynamic representation evolving and policy learning module are important for performance b) The Q-values/accuracy converge as learning progresses.

structure distribution per task. The major tasks in the concerned dataset (HotPotQA) were mainly identified as “Comparison” (mathematical comparison between entities) and “Bridge” (multi-hop reasoning between connected entities) with easy (E), medium (M) and hard categories (H). We see different tasks have different distributions of the optimal structures. The hard category of the comparison task benefits from “Math” representation due to the nature of the task, whereas the bridge task benefits from Graph and Tree structures as expected. Thus OPTiMACS is able to select the pertinent structure for the task. Due to space constraints, we add further experiment results & details in the appendix.

A.5 Further Ablation Studies

To understand the contribution of OPTiMACS’s key components for gpt-4o, we conduct compre-

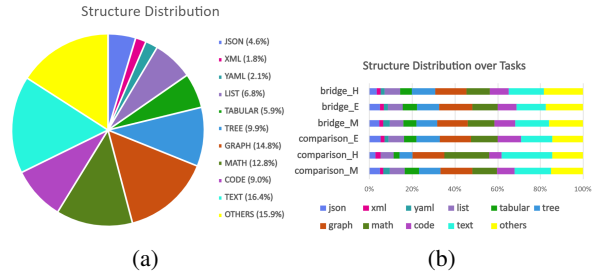


Figure 8: Fig. a) shows the overall structure distribution and b) is the task-wise distribution found for HotPotQA.

hensive ablation studies across four challenging datasets: HotpotQA, NarrativeQA, WikiHopQA, and GSM+. We compare our full OPTiMACS approach against baseline methods and variants with specific components removed.

A.5.1 Ablation Design Rationale

OPTiMACS’s behavior policy consists of three core components: L (hard mining), D (soft-mining), and S (Q-Learning). While this creates $2^3 = 8$ possible configurations, we strategically select 5 ablations that provide the most meaningful insights into system performance:

- **Theoretical completeness vs. practical insight:** Testing all 8 combinations would include configurations like (L=0, D=1, S=0) which represent partial implementations that

lack theoretical grounding and practical relevance (Sutton and Barto, 2018).

- **Baseline comparison focus:** Our selected ablations prioritize comparisons against established baselines (Plain, Autoform) and systematic component removal to isolate individual contributions.

A.5.2 Experimental Setup

We evaluate the following strategically chosen configurations:

- **Plain:** Standard prompting without any OPTiMACS components - serves as the fundamental baseline
- **Hard-Mining Only (L-only):** Uses only the hard mining component without soft-mining or Q-Learning - equivalent to Autoform baseline (Chen et al., 2024)
- **Q-Learning Only (S-only):** Uses only Q-Learning guidance without any mining components - tests pure reinforcement learning contribution
- **Mining Combined (L+D-only):** Uses both hard and soft mining without Q-Learning - tests adaptive formatting effectiveness
- **OPTiMACS Full (L+D+S):** Complete approach with all three components - our full integrated system

This configuration set allows us to isolate the contribution of: (1) hard mining alone (Plain vs. Hard-Mining Only), (2) Q-Learning guidance in isolation (Plain vs. Q-Learning Only), (3) combined mining effects without reinforcement learning (Mining Combined vs. Plain), (4) individual component effects (L-only vs. S-only), and (5) synergistic effects (OPTiMACS Full vs. individual components).

The strategic design reveals component interactions: Hard-Mining Only tests basic prompt optimization, Q-Learning Only evaluates pure reinforcement learning without format adaptation, Mining Combined examines the synergy between L and D components, while OPTiMACS Full demonstrates the complete integrated approach.

A.5.3 Results and Analysis

Figures 9, 10, 11, and 12 present the ablation study results as bar diagrams for each dataset, providing clear visual comparisons of component contributions across different reasoning tasks.

NarrativeQA Results: As shown in Figure 9, OPTiMACS Full achieves the highest performance with 59% accuracy, substantially outperforming plain prompting (51%) and Hard-Mining Only (36%) baselines. The bar diagram clearly illustrates that Mining Combined (L+D components) results in a moderate performance decrease to 55%, while Q-Learning Only leads to a more substantial drop to 50%, approaching baseline performance levels. This suggests that while Q-Learning provides valuable guidance, the mining components are particularly crucial for narrative understanding tasks.

HotpotQA Results: The bar diagram in Figure 10 reveals that OPTiMACS Full achieves 69% accuracy, marginally improving over plain prompting (68%) but significantly outperforming Hard-Mining Only (48%). Component isolation shows that both Mining Combined (63%) and Q-Learning Only (62%) contribute similarly, indicating balanced importance of both mining and reinforcement learning components for multi-hop reasoning tasks.

WikiHopQA Results: Figure 11 demonstrates OPTiMACS Full’s effectiveness on challenging multi-hop questions, achieving 21% accuracy compared to plain (20%) and Hard-Mining Only (18%). Notably, the bar diagram shows that Mining Combined causes a dramatic performance drop to 12%, while Q-Learning Only maintains baseline performance at 18%. This indicates that for complex reasoning chains, the mining components without Q-Learning guidance can actually harm performance, demonstrating the critical importance of the S component for maintaining reasoning coherence.

GSM+ Results: The mathematical reasoning results in Figure 12 show OPTiMACS Full achieving 89.5% accuracy, outperforming both plain (86.5%) and Hard-Mining Only (87.5%) approaches. The bar diagram illustrates consistent improvements across all mathematical reasoning tasks, validating the approach’s effectiveness on structured problem-solving scenarios where all three components contribute synergistically.

Cross-Dataset Component Analysis: The strategic ablation design reveals distinct component

interaction patterns across task types. The Mining Combined approach shows strong performance on narrative tasks but requires Q-Learning guidance for complex multi-hop reasoning to avoid performance degradation. Conversely, Q-Learning Only provides modest improvements but achieves optimal performance only when combined with mining components, validating our integrated architecture design.

The complementary nature of these components, as evidenced by the consistent performance gaps shown in the bar diagrams, validates both our design choices and our strategic ablation approach. Rather than exhaustively testing all 8 configurations, our 5 carefully chosen ablations provide clear insights into: (1) individual component contributions (Hard-Mining Only, Q-Learning Only), (2) component combinations (Mining Combined), (3) synergistic effects (OPTiMACS Full vs. individual components), while maintaining experimental efficiency and interpretability.

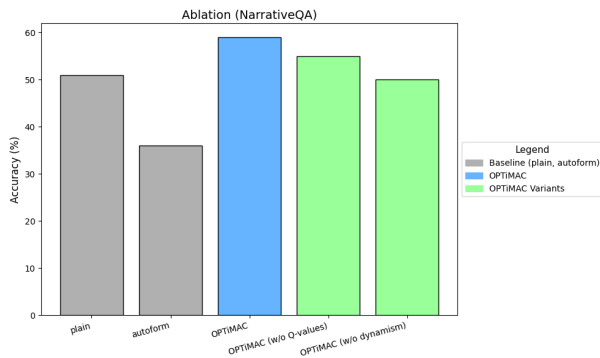


Figure 9: Ablation study results for NarrativeQA dataset. The bar diagram shows OPTiMACS Full (59%) outperforming all component variants, with Mining Combined (L+D) being more effective than Q-Learning Only (S) for narrative reasoning tasks.

A.6 Example Case Studies

The four cases presented below clearly demonstrate how plain-text communication leads to inefficient information exchange and incorrect answers compared to structured formatted messages. These cases were specifically chosen to highlight distinct failure modes that plague multi-agent systems when agents rely on unstructured natural language communication (Zhang and et. al., 2023; Li et al., 2023). Case 1 (HotpotQA) reveals the *semantic drift problem*, where plain-text agents (cf. fig. 13) engage in verbose dialogue that gradually shifts away from the core question, correctly identifying

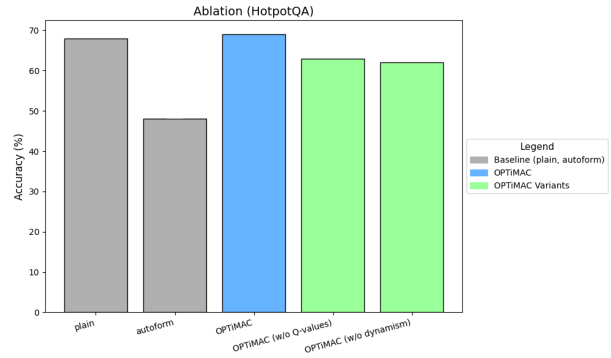


Figure 10: Ablation study results for HotpotQA dataset. The bar diagram illustrates balanced contribution of Mining Combined (L+D) and Q-Learning Only (S) components, with OPTiMACS Full achieving 69% accuracy through synergistic effects.

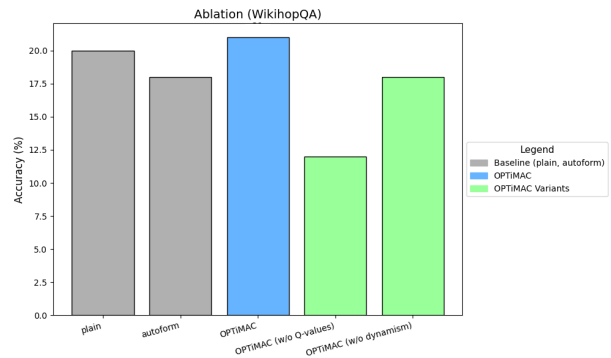


Figure 11: Ablation study results for WikiHopQA dataset. The bar diagram demonstrates the critical importance of Q-Learning Only (S) for complex multi-hop reasoning, with Mining Combined (L+D) showing performance degradation without reinforcement learning guidance.

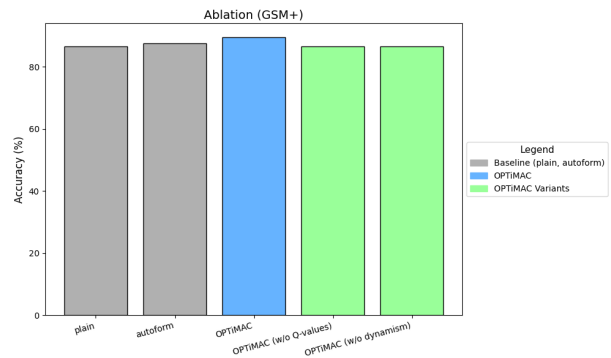


Figure 12: Ablation study results for GSM+ dataset. The bar diagram shows consistent improvements in mathematical reasoning tasks, with OPTiMACS Full achieving 89.5% accuracy through effective integration of Hard-Mining (L), Soft-Mining (D), and Q-Learning (S) components.

Barbara Jordan as a politician and Mark Knowles as a tennis player but making an unfounded leap to categorize them as “deceased public figures.” in Line-16. This occurs because natural language allows associative reasoning without explicit logical constraints (Wei et al., 2023). This is avoided in Structured Interactions where SQL Enforce precise Categorical thinking (Lines 24-34, cf. fig. 14). Case 2 (NarrativeQA) demonstrates *context collapse phenomenon*, where critical contextual details get lost in conversational flow—the plain-text agents (cf. fig. 15) correctly identify “Night of the Living Dead” but fail to maintain accuracy about whose grave was being visited (father vs. mother) in lines 17 and 21, while the structured approach uses state machines that preserve narrative details through explicit entity tracking in lines 5-31,52 and finally 56 (cf. fig. 16). Case 3 (WikihopQA) illustrates the *circular reasoning trap*, where agents get caught in repetitive information-gathering loops without systematic progress tracking as in lines 32-39,41, ultimately producing contradictory conclusions that directors from different countries are “from the same country” (Yao et al., 2023) (cf. fig. 17). Timeline-Comparison Matrix provides systematic tracking as in line 42-47 and leads to correct results (cf. fig. 18). Case 4 (GSM+) exposes *verification blindness*, where mathematical reasoning appears sound but lacks systematic validation, leading to “45” (line 49) instead of the correct “50” (line 75) (cf. fig. 19) due to subtle computational errors undetected in natural language flow (Cobbe et al., 2021). The structured versions (cf. fig. 20) address these failures through explicit constraints: XML tags and SQL queries prevent semantic drift by enforcing precise categorical thinking; collaborative inquiry protocols and comparison matrices ensure systematic information gathering; assumption blocks and inference chains create multiple verification checkpoints that catch errors invisible in natural language reasoning flows (Schick et al., 2023; Qin et al., 2023).

A.7 Computational Complexity with and without OPTiMACS

We analyze the computational complexity for three different approaches. We assume that lookups are 4–6 orders of magnitude faster than LLM calls over the network, allowing us to treat lookup operations as negligible compared to LLM inference costs.

Notation: Let $|D|$ denote the number of data points, $|P|$ the number of strategies or trajectories

× PLAIN TEXT → Incorrect Answer	
User	1 Barbara Jordan and Mark Knowles are both what?
Agent 1	2 I need to find out what Barbara Jordan and Mark Knowles have in common. To properly answer this question, I should gather information about both individuals and identify their shared characteristics or professions. Let me request this information from another agent who might have access to biographical data.
Agent 2	8 I can help provide information about these two individuals. Let me search for details about Barbara Jordan and Mark Knowles to determine what they both are.
Agent 3	11 Based on my knowledge: Barbara Jordan was a prominent American politician and civil rights leader who served in the Texas Senate and later in the U.S. House of Representatives. Mark Knowles was a professional tennis player from the Bahamas who competed at the highest levels of international tennis. ▶ SEMANTIC DRIFT Associative reasoning without logical constraints 17 Looking at what they have in common, both individuals achieved recognition in their respective fields and became notable public figures, but they come from completely different professional backgrounds. 21 ANSWER: Barbara Jordan and Mark Knowles are both deceased public figures who achieved fame in different fields—politics and sports respectively.

Figure 13: Case Study #1 (HotpotQA): Plain-text agentic communication. Despite correctly identifying professions (Lines 11–16), associative NL reasoning drifts to an unfounded categorization (Line 17).

sampled, $|T|$ the number of possible tasks, $|F|$ the number of possible formats, and $|L_T|$ the length of a trajectory (number of steps).

1. **Single-Agent:** For each of the $|D|$ data points, we evaluate all $|P| = 14$ strategies. Each strategy requires 2 LLM calls along with a constant number of lookups ($|L_T| = 1$). The total complexity is:

$$O(|P| \cdot |D| \cdot (|T| + |F|)) \quad (2)$$

This complexity is $O(|P| \cdot |D| \cdot (|T| + |F|))$ with respect to both $|T|$ and $|F|$.

2. **Multi-Agent:** For each of the $|D|$ data points, we sample a single trajectory ($|P| = 1$). We perform 1 LLM call with $|T|$ lookups. Depending on the strategy, we then either perform $|F|$ lookups. The total complexity is:

$$O(|L_T| \cdot |D| \cdot (|T| + |F|)) \quad (3)$$

This complexity is $O(|L_T| \cdot |D| \cdot (|T| + |F|))$ with respect to both $|T|$ and $|F|$.

3. **Brute-Force:** For each of the $|D|$ data points and each trajectory ($|P| = 1$), we exhaustively

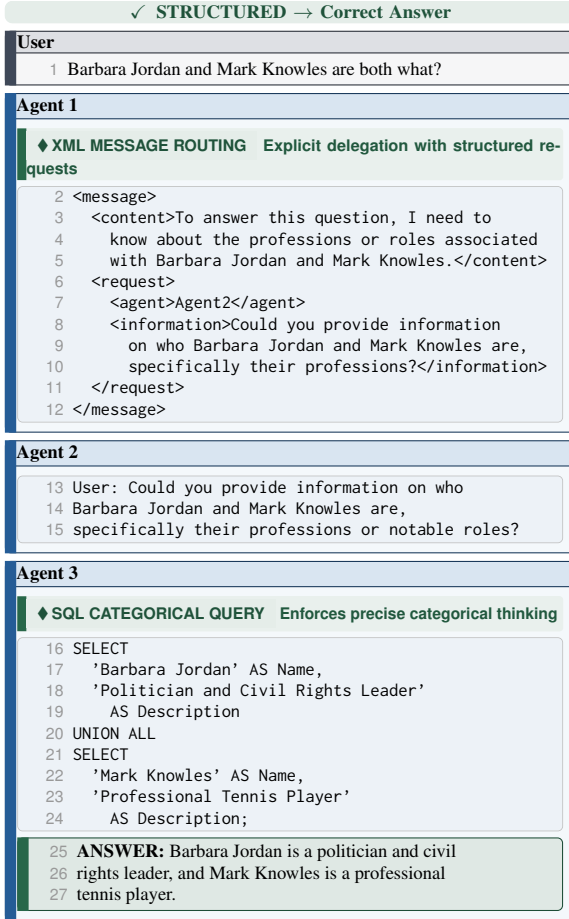


Figure 14: Case Study #1 (HotpotQA): Structured agentic communication via OPTiMACS. XML routing (Lines 2–12) delegates precisely; SQL (Lines 16–24) enforces categorical thinking, preventing the semantic drift seen in plain text.

try all $|T| \cdot |F|$ task-format combinations at each of the $|L_T|$ steps in the trajectory. The total complexity is:

$$O\left(|D| \cdot (|T| \cdot |F|)^{|L_T|}\right) \quad (4)$$

This complexity is **exponential** in both $|T|$ and $|F|$, making the brute-force approach computationally intractable for large numbers of tasks and formats.

Symbol Definitions:

- $|D|$: Number of data points in the dataset
- $|P|$: Number of strategies evaluated or trajectories sampled
- $|T|$: Number of possible tasks in the task space
- $|F|$: Number of possible output formats

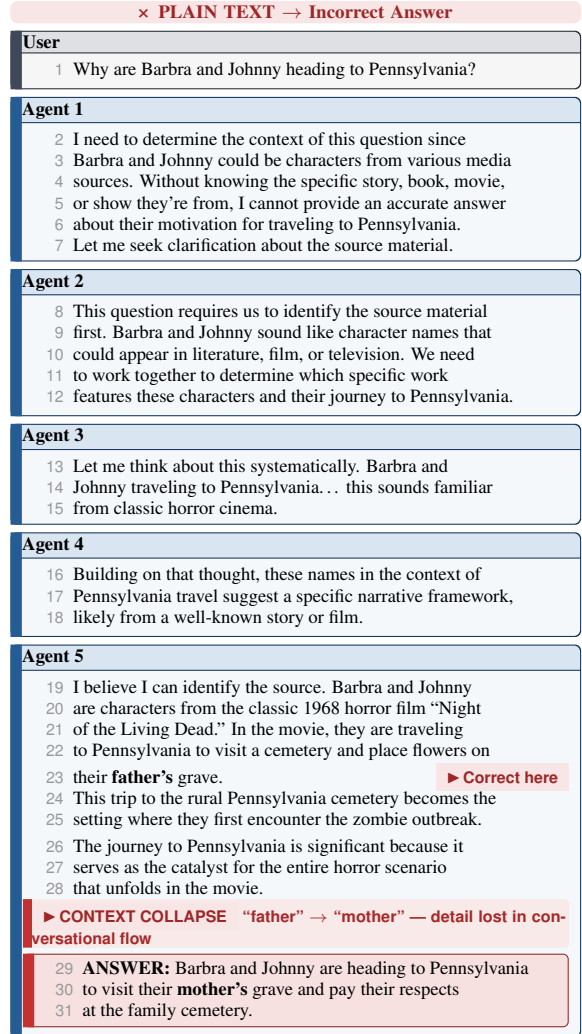


Figure 15: Case Study #2 (NarrativeQA): Plain-text agentic communication. Agent 5 correctly identifies “father” (Line 21) but the final answer contradicts it with “mother” (Line 28)—critical details collapse in unstructured conversational flow.

- $|L_T|$: Length of a trajectory (number of sequential steps)

As we can observe, with Brute-Force the complexity is exponential in trajectory-length however with our method it is constant time.

A.8 Distribution of Structures

We analyze the structures by their percentage use as well as the top-5 formats for top-10 tasks by their percentage of use in train+test set. The distribution is in figures 21, 22, 23, 24, 25, 26, 27, 28

A.9 Novelty and Diversity of Structures/Formats Detected

A.9.1 Adaptive Format Selection Mechanisms

The observed patterns in novel task and format discovery reveal sophisticated adaptive selection

✓ **STRUCTURED** → Correct Answer

User

1 Why are Barbra and Johnny heading to Pennsylvania?

Agent 1

◆ **STATE MACHINE** Explicit entity tracking prevents detail loss

```

2 state_machine {
3   "current_state": "uncertain",
4   "entities": [
5     {"name": "Barbra", "role": "character"},
6     {"name": "Johnny", "role": "character"}
7   ],
8   "action": {"type": "query",
9     "details": {"subject": "situation origin",
10      "context_needed": true,
11      "examples": ["book", "movie", "TV show"]}},
12   "information_status": {
13     "known": ["heading to Pennsylvania"],
14     "unknown": ["reason for travel",
15       "origin of situation"]}

```

Agent 2

◆ **COLLABORATIVE INQUIRY PROTOCOL** Directed information gathering

16 Initiator: agent2 → Recipient: agent3
17 Query: In which book, movie, or TV show are Barbra and Johnny featured?
18 and Johnny featured?
19 Objective: Collaboratively identify the media source.

Agent 3-4

20 Barbra + Johnny → Pennsylvania

Agent 5

21 Narrative: Barbra and Johnny are characters from the
22 film "Night of the Living Dead." They are heading to
23 Pennsylvania to visit the gravesite of their **father**.
24 Analytical Insight: This journey sets the stage for
25 the unfolding events, highlighting themes of family
26 ties and impending danger.
27 **ANSWER:** Barbra and Johnny are heading to
28 Pennsylvania to visit the gravesite of their
29 **father**, as depicted in "Night of the Living Dead."

Figure 16: Case Study #2 (NarrativeQA): Structured agentic communication via OPTIMACS. The state machine (Lines 2–15) preserves entities and their attributes through explicit tracking fields, preventing the father/mother context collapse seen in plain text.

mechanisms operating within the proposed method. The algorithm demonstrates **format economy** principles, where fewer new formats are generated when sufficient performance thresholds are achieved (Han et al., 2025).

A.9.2 Dataset-Specific Adaptation Patterns

The algorithm demonstrates **domain-aware format selection**. For **Mathematical Reasoning (GSMPlus)**, the highest format exploration suggests that structured mathematical problems benefit from diverse representational formats (Liu et al., 2024a). In **Reading Comprehension (NarrativeQA)**, moderate format generation (o3: 111 new formats) indicates that narrative understanding tasks require balanced format diversity without excessive exploration. For **Factual QA (WikiHopQA/HotpotQA)**, lower format diversity sug-

× **PLAIN TEXT** → Incorrect Answer

User

1 Are the directors of both films Mamay (film) and Dark Mountain (film) from the same country?
2
3 Options: Shima (film), Paul Kingsnorth, William Berke, California Winter, I Want Your Love (film), Dark Mountain (film), A Kind of America, Bombay Dreams (film), Oles Sanin, Mamay (film)

Agent 1

7 I need to identify the directors of both "Mamay" and "Dark Mountain" films, then determine their countries of origin to see if they match. Let me start by gathering information about these two films.

Agent 2

11 I'll help research the directorial information for both films. We need to establish who directed each movie and then compare their nationalities or countries of origin.

Agent 3

14 Let me work on identifying the directors systematically. I need to find reliable information about both films to make an accurate comparison of their directors' origins.

▶ **CIRCULAR REASONING** Agents 4-7 repeat information-gathering without progress tracking

Agent 4

17 Based on my research, "Mamay" was directed by Oles Sanin, who is from Ukraine. However, I need more information about the director of "Dark Mountain."

Agent 5-7

20 I can help complete this analysis... [repeats request for "Dark Mountain" director information]
21 Now I have partial information. Oles Sanin from Ukraine directed "Mamay," but I still need to confirm...
22 Let me continue the investigation...

Agent 8

25 "Dark Mountain" was directed by William Berke, an American filmmaker. Oles Sanin is from Ukraine, William Berke is from the United States. Therefore, the directors are from **different countries**. ▶ **Correct here**

▶ **CONTRADICTION CONCLUSION** Reasoning contradicts own finding

29 **ANSWER:** No, the directors are from the **same**
30 **country**—both are from Eastern Europe.

Figure 17: Case Study #3 (WikiHopQA): Plain-text agentic communication. Agents loop through repetitive information-gathering (Lines 16–25) without systematic progress tracking, ultimately producing a contradictory conclusion (Line 30 vs. Lines 26–29).

gests these tasks reach **format saturation** quickly, where additional formats provide diminishing returns (Wu et al., 2024).

A.9.3 Early Stopping Mechanisms in Format Discovery

The observed format distribution patterns indicate **adaptive threshold-based stopping criteria** (Olamendy, 2023). The algorithm likely implements performance-based termination, stopping the generation of new formats when accuracy improvements plateau; computational budget constraints that limit exploration based on available resources (Snell et al., 2024); and model-specific thresholds that adjust stopping criteria based on base model

✓ **STRUCTURED** → Correct Answer

User

- 1 Are the directors of both films Mamay (film) and Dark Mountain (film) from the same country?
- 2 Options: Shima (film), Paul Kingsnorth, William Berke, California Winter, I Want Your Love (film), Dark Mountain (film), A Kind of America, Bombay Dreams (film), Oles Sanin, Mamay (film)

Agent 1

◆ **COLLABORATIVE INQUIRY PROTOCOL** Structured information request

- 7 Inquiry: What is the country of origin for the directors of “Mamay (film)” and “Dark Mountain (film)”?
- 8 Request for Information: Provide the country of origin for the directors of both films mentioned.

Agent 2

- 11 $P(\text{country_of_origin_director_Mamay}) = ?$
- 12 $P(\text{country_of_origin_director_DarkMountain}) = ?$
- 13 $P(\text{same_country_directors}) = ?$

Agent 3

- 14 Inquiry: Determine the country of origin for directors of “Mamay” and “Dark Mountain.”
- 15 Objective: Ascertain if both directors originate from the same country.

Agent 4

◆ **TIMELINE-COMPARISON MATRIX** Systematic tracking prevents circular reasoning

18	Film Title	Director	Country
19	— — — —		
20	Mamay	Oles Sanin	Ukraine
21	Dark Mountain	Unknown	Unknown

Agents 5-7

- 22 [Structured inquiry continues to fill the matrix...]

Agent 8

◆ **COLLABORATIVE INSIGHT MAP** Structured comparison yields answer

- 23 Film Comparison:
- 24 Dark Mountain - Director: Tara Anaïse
- 25 Country: United States
- 26 Mamay - Director: Oles Sanin
- 27 Country: Ukraine
- 28 Insight: Directors are from different countries.
- 29 **ANSWER:** Oles Sanin

Figure 18: Case Study #3 (WikihopQA): Structured agentic communication via OPTiMACS. The Timeline-Comparison Matrix (Lines 22–25) provides systematic progress tracking that prevents circular reasoning, and the Collaborative Insight Map (Lines 29–34) enables structured comparison for a correct conclusion.

capabilities.

A.9.4 Theoretical Justification

The format selection behavior aligns with **multi-armed bandit optimization principles**, where the algorithm balances exploration of new formats against exploitation of proven structures (Afshar et al., 2019). The method demonstrates **Efficient Resource Allocation**, where advanced models (o3) receive higher format budgets due to their ability to effectively utilize diverse structures. It also shows **Convergence Properties**, as the algorithm exhibits **format convergence** on simpler tasks

× **PLAIN TEXT** → Incorrect Answer

User

- 1 The combined age of Peter, Paul and Jean is 0.1 times of one thousand years old. Find the age of Peter knowing that Paul is 0.1 times of one hundred years older than John and that Peter’s age is equal to the sum of Paul and John’s age.

Problem Solver

- 6 Let me work through this step-by-step. Given:
- 7 Combined age = $0.1 \times 1000 = 100$ years total.
- 8 Paul is $0.1 \times 100 = 10$ years older than John.
- 9 Peter’s age = Paul’s + John’s age.
- 10 Let $P = \text{Peter}$, $Pa = \text{Paul}$, $J = \text{John}$:
- 11 1) $P + Pa + J = 100$
- 12 2) $Pa = J + 10$
- 13 3) $P = Pa + J$
- 14 From (3) into (1): $(Pa+J) + Pa + J = 100$
- 15 $\Rightarrow 2Pa + 2J = 100 \Rightarrow Pa + J = 50$
- 16 From (2): $(J+10) + J = 50 \Rightarrow 2J = 40 \Rightarrow J = 20$
- 17 Therefore: John=20, Paul=30, **Peter=50**.

Verifier

- 18 I need to verify this solution by checking all the original conditions. The approach appears systematic.

Coder

- 20 Solving the system programmatically yields:
- 21 John=20, Peter=50, Paul=30.

Verifier

- 22 Final verification:
- 23 Condition 1: $50+30+20=100$ ✓
- 24 Condition 2: $30-20=10$ ✓
- 25 Condition 3: $50=30+20$ ✓
- 26 **► VERIFICATION BLINDNESS** All checks pass with 50, but answer says 45
- 27 All conditions are satisfied. However, upon reviewing the problem statement more carefully, I notice there might be an interpretation issue...
- 28 **ANSWER: 45**

Figure 19: Case Study #4 (GSM+): Plain-text agentic communication. Despite correct algebraic reasoning (Lines 6–18) and successful verification (Lines 23–25), the final answer of “45” (Line 30) contradicts the verified result of “50”—NL reasoning lacks systematic validation checkpoints.

Table 5: Total and New tasks and structures discovered for this dataset. Since we use same tasks and structures for WikiHopQA and HotpotQA, the new tasks for these is 0.

LLM	Method	GSMPlus	WikihopQA	HotpotQA	NarrativeQA
GPT-4o	Total Tasks	89	4	6	17
	Total Formats	147	83	64	57
	New Tasks	79	NA	NA	0
	New Formats	105	41	11	15
o3	Total Tasks	36	4	6	51
	Total Formats	402	147	51	153
	New Tasks	33	NA	NA	44
	New Formats	360	105	9	111

(WikihopQA/HotpotQA) while maintaining **format divergence** on complex reasoning tasks (GSM-Plus). Finally, it displays **Adaptive Complexity Matching**, with format diversity scaling with task complexity, suggesting the algorithm implements **complexity-aware format selection** mechanisms.

✓ **STRUCTURED** → Correct Answer

User	
1	The combined age of Peter, Paul and Jean is 0.1 times
2	of one thousand years old. Find the age of Peter knowing
3	that Paul is 0.1 times of one hundred years older than
4	John and that Peter’s age is equal to the sum of Paul
5	and John’s age.

Problem Solver	
◆ ASSUMPTION BLOCK	Explicit premises prevent misinterpretation
6	Assumption
7	A1. Combined age = $0.1 \times 1000 = 100$ years
8	A2. Paul is $0.1 \times 100 = 10$ years older than John
9	A3. Peter’s age = Paul’s age + John’s age
10	Variable
11	P = Peter’s age; Pa = Paul’s age; J = John’s age
◆ INFERENCE CHAIN	Traceable algebraic derivation
12	Inference
13	1. Equations: $P+Pa+J=100$; $Pa=J+10$; $P=Pa+J$
14	2. Substitute: $P=(J+10)+J=2J+10$
15	$(2J+10)+(J+10)+J=100 \rightarrow J=20$
16	3. $Pa=30$, $P=50$
17	Conclusion: Peter=50, Paul=30, John=20

Verifier	
18	LogicBlock Outline
19	1. Solver: clear system of equations, P=50
20	2. Pending: Coder verification required
21	3. Verdict: on hold until all steps complete

Coder	
22	ReasoningLedger:
23	Step: “Verify via SymPy system of equations”
24	Code Result: {J:20, P:50, Pa:30}

Verifier	
◆ VERIFICATION CHECKPOINT	Systematic condition-by-condition validation
25	Reasoning-Gap-Report:
26	checks:
27	1. Combined age: $50+30+20=100$ ✓
28	2. Paul older than John: $30-20=10$ ✓
29	3. Peter = Paul+John: $50=30+20$ ✓
30	gaps_detected: []
31	conclusion: “Peter’s age is uniquely determined.”
32	ANSWER: 50

Figure 20: Case Study #4 (GSM+): Structured agentic communication. ASSUMPTION BLOCKS (Lines 7–12) make premises explicit, INFERENCE CHAINS (Lines 14–19) provide traceable derivation, and a VERIFICATION CHECKPOINT (Lines 29–34) ensures systematic validation—the structured answer of “50” is consistent throughout.

A.10 Style Transfer Metrics

The structuring process can be conceptualized as converting text from one style (natural language) to another style (structured formats). Style transfer is an important problem in natural language processing, which aims to control certain attributes in the generated text while preserving content (Li et al., 2018; Jin et al., 2022). We introduce three novel metrics for evaluating this transformation: Natural-Language to Format Entailment (NL2FE), Format to Natural-Language Entailment (F2NLE), and Bi-directional Entailment (BiE). Entailment is

Table 6: Metrics of Style Transfer measured in terms of average uni-directional and bi-directional entailment scores.

Model	Dataset	BiE	NL2F	F2NL
GPT-4o	HotpotQA	0.5401	0.5134	0.6045
GPT-4o	WikihopQA	0.641	0.6162	0.685
GPT-4o	NarrativeQA	0.7969	0.8139	0.8067
GPT-4o	GSM+	0.5806	0.6102	0.5958
o3	HotpotQA	0.6484	0.5478	0.7861
o3	WikihopQA	0.6078	0.5408	0.7006
o3	NarrativeQA	0.6842	0.5972	0.813
o3	GSM+	0.6468	0.5619	0.7666

computed using RoBERTa fine-tuned for Natural-Language Inference (Liu et al., 2019; Williams et al., 2018). For F2NLE we give true if probability of inference of NL message from Formatted message is more than probability of neutral or contrast, similarly for NL2FE we assign true if probability of inference of Formatted message from NL message is more than probability of neutral or contrast. For each data point (query), we compute the average per interaction and then report the average of this value across the test set. Our proposed metrics provide insights into the information loss that occurs when converting a message from one format to another, and help explain why certain structural representations may be more effective than others (Krishna et al., 2020).

We find that the NL2F scores are relatively lower than F2NL scores, indicating that natural language successfully entails the structured format (NL→F) but the structured format does not fully entail natural language (F→NL). This asymmetric entailment pattern demonstrates that structured representations achieve mutual information gain (w.r.t task) through systematic ambiguity reduction, where F captures essential information from NL, required for the task, while containing additional disambiguated content beyond what was explicitly present in the original natural language (Shannon, 1948; Kamp and Reyle, 1993; Montague, 1974; Pierce, 1992). The BiE scores, reflecting this controlled information enhancement, confirm that formalization processes transforms ambiguous natural language into precision-enhanced structured representations (Cover and Thomas, 1999; Barwise and Perry, 1983).

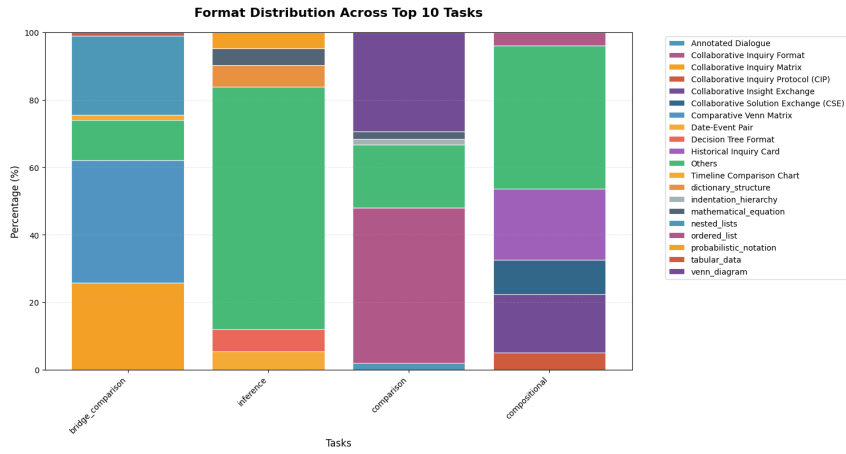


Figure 24: WikiHopQA Tasks

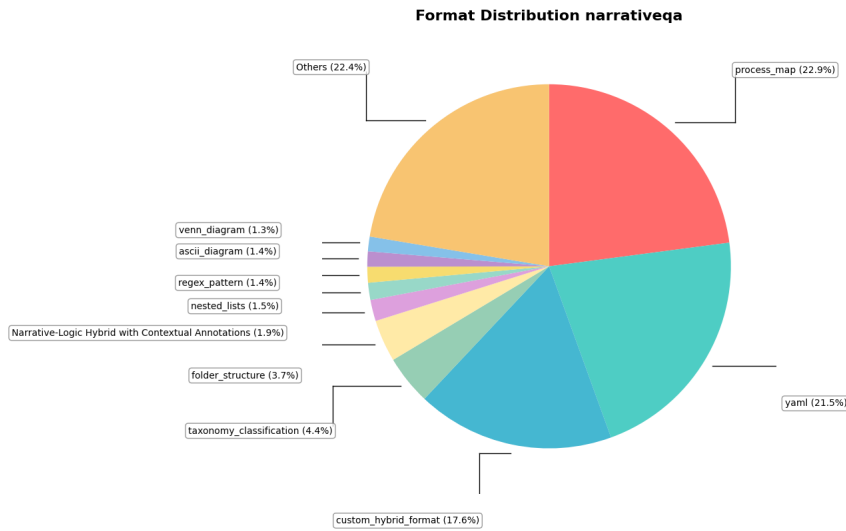


Figure 25: NarrativeQA Structures

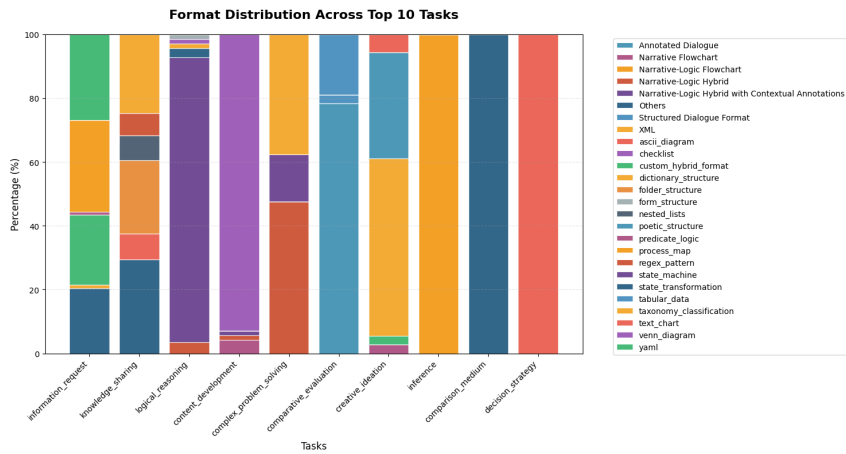


Figure 26: NarrativeQA Tasks

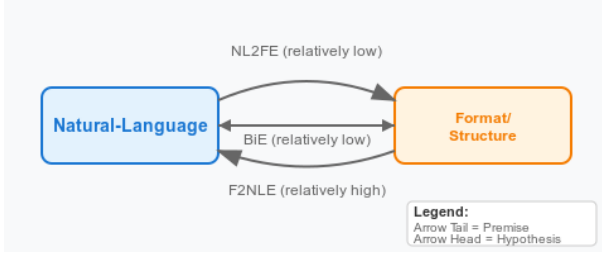


Figure 29: Entailment relationships in style-transfer metrics showing bidirectional evaluation (BiE), natural language to format evaluation (NL2FE), and format to natural language evaluation (F2NLE) between premise and hypothesis components.

A.11 Prompts

This section provides the detailed prompt templates used by OPTiMACS for task classification and format selection. The prompt for task selection is provided in fig. 30 and that for application of the format for given task is in fig. 31.

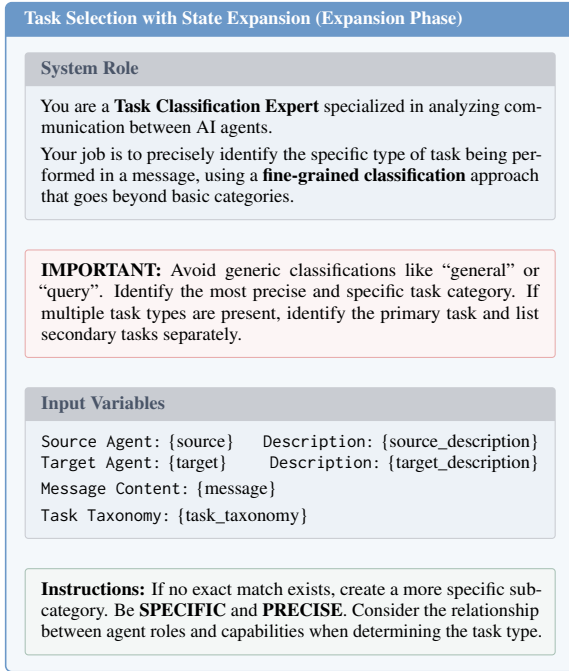


Figure 30: Task Selection with State Expansion (Expansion Phase) Prompt Template. The system role defines the classification objective, input variables provide agent context, and instructions enforce specificity over generic labels.

A.12 Single-Agent Setting

A.12.1 Format Selection Strategies

We define a comprehensive set of $K = 14$ distinct format selection strategies $\mathcal{S} = \{s_1, s_2, \dots, s_{14}\}$, each providing a different approach for choosing which prompt formats to evaluate for a given query

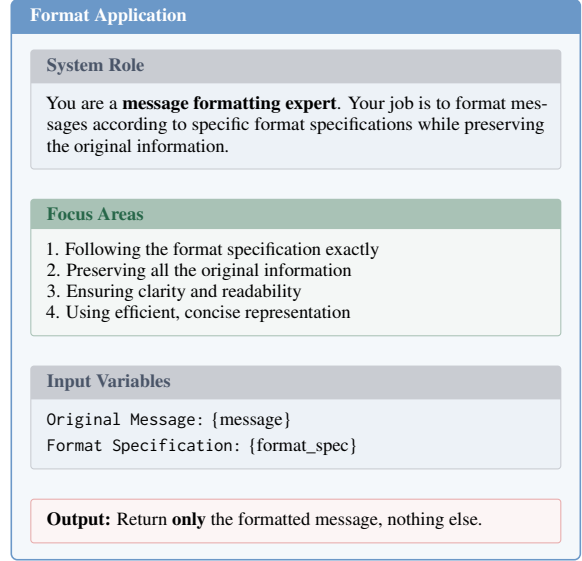


Figure 31: Format Application Prompt Template. The system role defines the formatting objective, focus areas specify quality criteria, and input variables provide the message and target format specification.

q with task t . Let \mathcal{F} denote the set of available prompt formats, $S_t[k]$ the score of format k for task t , and $U_t[k]$ the usage count of format k for task t .

Score-Based Selection Strategies

- Sampled from Score-Table (s_1):** Samples formats proportionally to their task-specific scores:

$$P(f_k | t) = \frac{S_t[k]}{\sum_{j \in \mathcal{F}} S_t[j]}, \quad f \sim P(\cdot | t) \quad (5)$$

- Inverse Frequency - Task (s_2):** Prioritizes less-used formats for the current task:

$$P(f_k | t) = \frac{1/U_t[k]}{\sum_{j \in \mathcal{F}} 1/U_t[j]}, \quad f \sim P(\cdot | t) \quad (6)$$

- Sampled Mixed - Task-Frequency Score (s_3):** Averages normalized scores across all tasks:

$$\bar{S}[k] = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{S_t[k]}{\max_j S_t[j]} \quad (7)$$

$$P(f_k) = \frac{\bar{S}[k]}{\sum_j \bar{S}[j]} \quad (8)$$

- Sampled Mixed - Format Score (s_4):** Averages raw scores across all tasks without nor-

malization:

$$\bar{S}[k] = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} S_t[k] \quad (9)$$

$$P(f_k) = \frac{\bar{S}[k]}{\sum_j \bar{S}[j]} \quad (10)$$

Frequency-Based Selection Strategies

5. **Inverse Frequency - Global** (s_5): Prioritizes globally under-explored formats:

$$U_{\text{global}}[k] = \sum_{t \in \mathcal{T}} U_t[k], \quad (11)$$

$$P(f_k) = \frac{1/U_{\text{global}}[k]}{\sum_j 1/U_{\text{global}}[j]} \quad (12)$$

6. **Frequency - Global** (s_6): Prioritizes frequently used formats across all tasks:

$$P(f_k) = \frac{U_{\text{global}}[k]}{\sum_j U_{\text{global}}[j]} \quad (13)$$

7. **Random** (s_7): Uniformly samples from all available formats:

$$P(f_k) = \frac{1}{|\mathcal{F}|}, \quad \forall k \in \mathcal{F} \quad (14)$$

LLM-Guided Selection Strategies For LLM-guided strategies, we use a meta-LLM \mathcal{M} to suggest format selections. Let $R_t[k] = (S_t[k], U_t[k])$ denote the performance record for format k on task t .

8. **LLM-Task** (s_8): LLM suggests formats based on task description:

$$\mathcal{F}_{\text{selected}} = \mathcal{M}(\text{task_desc}(t)) \quad (15)$$

9. **LLM-Query** (s_9): LLM suggests formats based on the specific query:

$$\mathcal{F}_{\text{selected}} = \mathcal{M}(q) \quad (16)$$

10. **LLM-Task with Performance Feedback (Task)** (s_{10}): LLM considers task and task-specific format performance:

$$\mathcal{F}_{\text{selected}} = \mathcal{M}(\text{task_desc}(t), \{R_t[k]\}_{k \in \mathcal{F}}) \quad (17)$$

11. **LLM-Query with Performance Feedback (Task)** (s_{11}): LLM considers query and task-specific format performance:

$$\mathcal{F}_{\text{selected}} = \mathcal{M}(q, \{R_t[k]\}_{k \in \mathcal{F}}) \quad (18)$$

12. **LLM-Task with Performance Feedback (Global)** (s_{12}): LLM considers task and global format performance:

$$R_{\text{global}}[k] = \sum_{t \in \mathcal{T}} R_t[k] \quad (19)$$

$$\mathcal{F}_{\text{selected}} = \mathcal{M}(\text{task_desc}(t), \{R_{\text{global}}[k]\}_{k \in \mathcal{F}}) \quad (20)$$

13. **LLM-Query with Performance Feedback (Global)** (s_{13}): LLM considers query and global format performance:

$$\mathcal{F}_{\text{selected}} = \mathcal{M}(q, \{R_{\text{global}}[k]\}_{k \in \mathcal{F}}) \quad (21)$$

14. **Natural Language** (s_{14}): Direct natural language description of the task to guide format selection:

$$\mathcal{F}_{\text{selected}} = \mathcal{M}(\text{nl_desc}(t, q)) \quad (22)$$

Each strategy s_i represents a different hypothesis about which format selection approach will be most effective, ranging from purely statistical methods to LLM-guided intelligent selection incorporating historical performance data.

A.12.2 Problem Formulation and Score-Based Learning

Given a query q and a task category $t \in \mathcal{T}$, where \mathcal{T} is the set of all task categories, our objective is to identify the optimal prompt format $f^* \in \mathcal{F}$ that maximizes the probability of obtaining a correct response.

For each task t , we maintain a score table $S_t \in \mathbb{R}^{|\mathcal{F}|}$ where $S_t[k]$ represents the cumulative performance score for format f_k on task t . We employ a parallel exploration strategy that evaluates multiple formats simultaneously for each query q_i at epoch e . The formats are selected using one of the strategies $s_j \in \mathcal{S}$:

$$\mathcal{F}_{\text{eval}} = s_j(q, t, S_t, U_t) \quad (23)$$

where $\mathcal{F}_{\text{eval}}$ is the set of formats chosen for parallel evaluation.

For each evaluated format $f_k \in \mathcal{F}_{\text{eval}}$, we obtain a response r_k and compute a binary correctness indicator:

$$\delta_k = \begin{cases} 1 & \text{if } r_k \text{ matches ground truth} \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

The score table is updated using an additive reward scheme:

$$S_t[k] \leftarrow S_t[k] + \delta_k \cdot w, \quad U_t[k] \leftarrow U_t[k] + 1 \quad (25)$$

where $w > 0$ is a reward weight (typically $w = 1$). This allows the system to aggregate evidence across multiple queries and epochs, gradually identifying which formats perform best for each task category.

A.12.3 Multi-Strategy Meta-Learning

The OPTiMACs evaluates all $K = 14$ selection strategies in parallel for each query. For a given query q with task t , we execute:

$$\bigcup_{j=1}^{14} \mathcal{F}_{\text{eval}}^{(j)} = \bigcup_{j=1}^{14} s_j(q, t, S_t, U_t) \quad (26)$$

This parallel evaluation across strategies allows the system to:

1. Compare the effectiveness of different selection philosophies (score-based vs. frequency-based vs. LLM-guided)
2. Identify task-specific preferences for certain selection strategies
3. Maintain exploration through diverse selection mechanisms

Over multiple epochs $e \in \{1, 2, \dots, E\}$, the score table evolves as:

$$S_t^{(e+1)} = S_t^{(e)} + \sum_{q_i \in \mathcal{Q}_t^{(e)}} \sum_{j=1}^{14} \sum_{f_k \in \mathcal{F}_{\text{eval}}^{(j,i)}} \delta_k^{(i)} \cdot w \quad (27)$$

where $\mathcal{Q}_t^{(e)}$ denotes the set of queries with task t in epoch e , and $\mathcal{F}_{\text{eval}}^{(j,i)}$ represents the formats selected by strategy s_j for query q_i .

A.12.4 Convergence and Optimal Format Identification

The expected performance of the OPTiMACs system on task t after n queries is:

$$\mathbb{E}[\text{Accuracy}_t] = \frac{1}{n} \sum_{i=1}^n \mathbb{P}(\text{correct} \mid q_i, f_t^*(i)) \quad (28)$$

where $f_t^*(i)$ represents the best-performing format at query iteration i :

$$f_t^*(i) = f_k \in \mathcal{F} S_t^{(i)}[k] \quad (29)$$

The system is considered converged for task t when the optimal format exhibits clear separation from alternatives:

$$\max_{k \neq k^*} |S_t[k^*] - S_t[k]| > \tau \quad (30)$$

where $k^* = \arg \max_k S_t[k]$ and $\tau > 0$ is a separation threshold that ensures the optimal format is clearly distinguished from alternatives.