

CoCoMaMa: Contextual Combinatorial Multi-Armed Bandit Router for Multi-Agent Systems with Volatile Arms

Jonathan Rau^{1,*}, Jonathan Bader¹, Philipp Wiesner¹ and Odej Kao¹

¹Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany

Abstract

Agentic Large Language Models (LLMs) are designed for specialized objectives using fine-tuning, prompting techniques, and tool calling to outperform general-purpose models in their expert domains. Standardization efforts like the Agent2Agent Protocol could drastically increase the number and heterogeneity of experts available via the Web. A router is required to find the best agent for any given task. However, existing LLM routing methods use a fixed-sized pool of models and often rely on offline training data such as benchmarks.

We propose CoCoMaMa and Neural-CoCoMaMa, a combinatorial contextual volatile multi-armed bandit approach that leverages similarities between tasks and agents by learning on online feedback. It can handle volatile arms by incorporating agent cards as defined by the Agent2Agent Protocol without requiring changes to the internal structures or retraining. Our experimental evaluation shows that CoCoMaMa and Neural-CoCoMaMa achieve better results than respective state-of-the-art algorithms using the LLM routing dataset SPROUT and a novel extended version of SPROUT with synthetic specialized agents.

Keywords

Multi-Agent Systems, Multi-Armed Bandit, Large Language Models, Agent routing, Online learning

1. Introduction

Large Language Models (LLMs) are a popular foundation to create specialized LLM agents by integrating tools or memory systems [1, 2]. The application of those agents is diverse, such as acting as personal assistants to retrieve weather forecasts, make travel arrangements [3], develop software [4, 5] or perform research [6]. The original work on Semantic Web [7] envisions many specialized agents integrated within the web with different access to resources. The Mixture of Experts concept, stating that training multiple specialized experts and using a sparse gating function to route requests, has been proven to be more efficient than training a single general-purpose model [8]. However, it is required to discover the capabilities of available agents first to be able to route a given task to the best-matching agent. Emerging AI Agent Protocols [9] such as the Agent2Agent (A2A) Protocol [10] propose describing agents, their interfaces and their skills in standardized agent-cards. An agent might then be able to call another agent over the web as a tool [11, 12]. The size of available WebAgents and resources via the web, openness of the web and governance aspects of such hypermedia Multi-Agent Systems (hMAS) [13] pose challenges for routers, as new agents can easily enter and leave the web and the performance of each agent might vary based on the context and the permissions of the client that interacts with it.

Multi-armed bandits (MAB) provide a mathematical framework for sequential decision-making under uncertainty. The classical MAB problem [14] involves an agent repeatedly choosing from a set of actions (arms) to maximize cumulative reward over time while balancing exploration of unknown options with exploitation of known good choices. Lu et al. [15] extend the basic MAB framework by incorporating contextual information affecting the payoff. A set of arms played in the same round forms a super arm in combinatorial MAB [16]. Settings in which the available arms can freely join and leave the pool of

The Second International Workshop on Hypermedia Multi-Agent Systems (HyperAgents 2025), in conjunction with the 28th European Conference on Artificial Intelligence (ECAI 2025); October 26, 2025, Bologna, Italy

*Corresponding author.

✉ j.rau.1@tu-berlin.de (J. Rau); jonathan.bader@tu-berlin.de (J. Bader); wiesner@tu-berlin.de (P. Wiesner); odej.kao@tu-berlin.de (O. Kao)

ORCID 0009-0004-0391-3590 (J. Rau); 0000-0003-0391-728X (J. Bader); 0000-0001-5352-7525 (P. Wiesner); 0000-0001-6454-6799 (O. Kao)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

options are usually referred to as sleeping [17] or volatile bandits [18]. Applying such MAB algorithms to the routing problem in hMAS has not been examined to the best of our knowledge.

In this paper, we propose using Contextual Combinatorial MAB with volatile arms to route tasks to agents based on their agent card, theoretically enabling an infinite number of volatile agents to enter and leave the pool without retraining a router.

Contributions. This paper makes the following contributions:

- We present CoCoMaMa, a novel MAB approach that learns from online feedback and efficiently explores and exploits similarities between tasks and agents in high-dimensional context spaces by adaptively discretizing the context space following statistically informed decisions.
- We propose Neural-CoCoMaMa, which improves the CoCoMaMa method by leveraging the benefits of neural networks while maintaining exploration behavior.
- We evaluate our approaches using the LLM routing dataset SPROUT [19] and a novel setup with synthetic specialized agents and compare them to three state-of-the-art contextual combinatorial volatile MAB algorithms.
- We provide an open-source implementation of our CoCoMaMa methods ¹.

2. Related Work

We outline related work focusing on LLM ensemble methods. Chen et al. [20] provide a classification with three groups: route to an expert before the inference step, combine multiple models during inference within the model architecture, and combine the results of different models after inference. Treating Web Agents as black boxes rules out the possibility of ensemble methods applied during inference. Thus, that path is neglected in the remainder of this work.

Before inference: Shnitzer et al. [21] propose repurposing benchmark datasets to learn router models for LLM selection by training a classifier for each candidate LLM. Many similar approaches are proposed to route to an expert from a fixed size of candidate models [22, 23, 24, 25, 26, 27, 28], while some of them also aim to balance cost and performance. Online algorithms could also be used to train the router. Sikeridis et al. [29] propose using reinforcement learning to train a router based on online user or AI feedback [30, 31]. There is also recent work looking into various bandits for online LLM routing [32, 19, 25, 33, 34]. Many of them are creating a task requirement vector using an embedding model like [35] and formulate the routing problem as a contextual bandit [15].

After inference: Cascading [36, 37, 38] can be used to escalate a task to a model with higher costs and higher expected quality, in case the answer of the initial model does not meet quality requirements. This requires feedback on the quality, which could be obtained by asking users. Though using Large Reasoning Models as a Judge is also viable [30] with limitations [31]. Majority voting to select the best answer is presented in Agent Forest [39]. Regenerating an answer after querying and ranking multiple agents was shown by Lv et al. [40]

Contrary to the solutions above, our solution considers metadata from agent cards and is built with high and volatile amounts of agents as routing targets in mind.

3. Problem Formulation

Consider a sequence of tasks indexed by time steps $t \in \{1, 2, \dots, T\}$. A task is a natural-language user query or intent that requires at least one agent response, but more responses do not hurt, e.g., weather retrieval or booking assistance. For each task t , there is a set of available agents M_t . Each agent $m \in M_t$ has distinct capabilities described by its agent card v_m . Agents may appear in multiple rounds, but can only be selected once per round. To ensure distinguishability, all agents in a round must have unique agent cards. If an agent’s capabilities or metadata change (e.g., through an update), it receives a

¹<https://github.com/dos-group/CoCoMaMa>

new agent card and is treated as a distinct agent. However, similar capabilities yield similar embeddings, allowing the router to transfer prior knowledge through semantic similarity in the context space.

Both the task t and an agent card v_m can be mapped into a multi-dimensional context space. By combining the context of task t with that of v_m , we obtain the context for the arm $x_{t,m}$. The true expected performance of agent m on task t , denoted $\mu_{t,m}$, is initially unknown. After the agent provides an answer, a “judge” infers $\mu_{t,m}$ by assigning a continuous score in $[0, 1]$. This feedback can come from users or other evaluation methods [30].

Because invoking and scoring agents typically incurs cost, we impose a fixed budget b that limits how many agents may be selected at each round. Following MAB terminology, we call the subset of chosen agents a super arm, denoted $S_t \subseteq M_t$, with $|S_t| = b$. The reward on task t is given by

$$r(S_t) = \max_{m \in S_t} \mu_{t,m},$$

reflecting the requester’s interest in only the best individual performance among the selected agents. The regret at task t is then the difference between the maximum achievable reward, i.e., $\max_{m \in M_t} \mu_{t,m}$, and the actual reward $r(S_t)$. The router’s objective is to select each S_t in order to minimize the cumulative regret over all T rounds, i.e.,

$$\min \sum_{t=1}^T \left[\max_{m \in M_t} \mu_{t,m} - r(S_t) \right].$$

4. Approach

The core idea of contextual, combinatorial, volatile MAB algorithms applied on hypermedia Multi-Agent Systems (hMAS) is to continuously learn and refine the understanding of the conceptual requirements per task and capabilities of each agent based on feedback. E.g., we might have observed that weather agent A provided a good result for the task “What is the weather going to be like in Bologna tomorrow?”. Then, the weather agent A might also perform well on the task “What is the weather going to be like in Rome tomorrow?”, because the tasks are very similar to each other. Later, we observe that weather agent A performs badly on the task “What is the weather going to be like in Berlin?”, but we also tried weather agent B and it provides a good result. Conclusively, a router might learn that requests for weather information in Italy should be routed to weather agent A, and requests for locations in Germany should be routed to weather agent B.

Therefore, we need to extract features describing each task and agent that allow us to exploit semantic similarities. This is described in 4.1. Next, algorithms that are capable of exploring the capabilities of agents and exploiting good task-agent assignments are covered in 4.2.

4.1. Constructing the Context Space

To apply contextual bandit algorithms effectively, both the task and the agent must be mapped and combined into suitable feature vectors that semantically describe how a specific task is assigned to a specific agent. Using pre-trained Sentence Transformers [35, 41] to produce compact embeddings out of a task is an established practice in LLM-routing (e.g. [24, 22]).

We propose creating feature vectors for the agent cards using the same method. This yields a pair of vectors for each task-agent combination, where semantically similar tasks and agent cards have similar embeddings, e.g., a high cosine similarity. The two embedding vectors are concatenated to form the unified context $x_{t,m}$ for the task-arm pair. This preserves all the available information, contrary to adding or multiplying the vectors or applying similarity metrics such as the Euclidean distance.

4.2. The Contextual, Combinatorial, Volatile Multi-Armed Bandits

Three state-of-the-art algorithms that support the contextual, combinatorial and volatile setting were identified and are presented briefly. This is followed by the introduction of our CoCoMaMa and Neural-CoCoMaMa algorithms.

4.2.1. CC-MAB

Chen et al. [42] propose splitting the context space into evenly sized non-overlapping regions and balancing exploration of unknown regions and exploitation of known regions with high expected reward in their CC-MAB algorithm. Whenever an arm is played, statistics for the respective context region are updated.

4.2.2. ACC-UCB

Nika et al. [43] introduce the Adaptive Contextual Combinatorial Upper Confidence Bound (ACC-UCB) algorithm, which uses a tree-based approach to iteratively partition the context space into non-overlapping regions of varying sizes using sets of hypercubes to define a region. A set containing a single hypercube is split by creating non-overlapping sets of hypercubes with half the side length. E.g., a context region containing a 2x2 chessboard could be split into sets based on rows, columns, or black and white tiles. Using sets of hypercubes to define regions consumes many resources in high-dimensional context spaces. E.g., using "all-MiniLM-L6-v2" [35] as an embedding model yields a 768-dimensional context space, which would be split into 2^{768} hypercubes. Initializing that many objects is not feasible on standard hardware (assuming 64GB memory as of 2025).

We change the implementation of ACC-UCB by using hyperrectangles, defined by a center vector and a length vector, to mark context regions. Nodes are split at the center along the dimension with the highest length (random selection to break ties). We term that variant High-Dimensional-ACC-UCB (HD-ACC-UCB) for the remainder of this work. It effectively just adds a small constraint to the core concept of Nika et al. [43]: splitting a region into "black and white tiles" is prohibited.

4.2.3. Neural-MAB

Lin et al. [44] follow a greedy selection strategy in their Neural-MAB algorithm using two neural networks with one hidden layer each to predict the reward of individual arms and the super arm.

4.2.4. CoCoMaMa

We hypothesize that making statistically informed decisions on the split condition and the split location can yield better results on high-dimensional context spaces. Especially in the field of hMAS, we expect many heterogeneous tasks and versatile agents, which require many dimensions to capture their nuances. Thus, we propose the CoCoMaMa Algorithm 1 as an improvement to HD-ACC-UCB. We maintain for each leaf node $x_{h,i} \in L^t$, the following metrics:

- $\bar{X}^t(x_{h,i}) \in \mathbb{R}^n$: running mean of the arms (i.e., the average context vector).
- $\bar{\mu}^t(x_{h,i}) \in \mathbb{R}$: running mean of the reward.
- $\text{Cov}^t(x_{h,i}) \in \mathbb{R}^n$: running covariances between each context dimension and the reward.
- $\text{Var}^t(\mu(x_{h,i})) \in \mathbb{R}$: running variance of the reward.
- $C^t(x_{h,i}) \in \mathbb{R}$: number of times the node has been played.
- $p(x_{h,i})$: parent node with all the associated metrics above at the state when it was split.

For each newly observed data point $(x_{t,m}, \mu_{t,m})$, the statistics for the node can be updated using Welford's Algorithm [45]. We introduce the combined confidence of a node and its parent node, defined as $c^t(x_{h,i}, p(x_{h,i})) := \sqrt{\frac{2 \log t}{C^t(x_{h,i}) + C^t(p(x_{h,i}))}}$. The node index is defined as:

$$g(x_{h,i}) := \max \begin{cases} \bar{\mu}^t(x_{h,i}) + c^t(x_{h,i}, p(x_{h,i})), \\ \bar{\mu}^t(p(x_{h,i})) + c^t(x_{h,i}, p(x_{h,i})) \end{cases} \quad (1)$$

A high variance of rewards for a node could indicate that splitting the node could yield a good and a bad performing region. Therefore, it seems desirable to split the nodes with the highest potential based

on the variance. A node is split if the variance of rewards of a node is θ times bigger than the weighted average variance of rewards of all leaf nodes:

$$\text{Var}^t(\mu(x_{h,i})) > \frac{\theta}{b \cdot t} \cdot \sum_x^{L^t} C^t(x) \cdot \text{Var}^t(\mu(x)) \quad (2)$$

With θ being defined as a hyperparameter. Frequent splitting could yield branches with relatively high confidence values $c^t(x_{h,i})$ for each individual node in a branch. Adding the following dampening condition circumvents overcommitment of the algorithm to explore and split such branches:

$$C^t(x_{h,i}) > C^t(p(x_{h,i})) \quad (3)$$

We propose selecting the dimension d^* with the highest running absolute covariance between context and reward:

$$d^* = \arg \max_{d \in [n]} |\text{Cov}^t(x_{h,i})_d| \quad (4)$$

and splitting the region at the running mean of the arms $\bar{X}^t(x_{h,i})$ in the respective dimension d^* .

Algorithm 1 CoCoMaMa-Algorithm

Require: budget b , split parameters θ, ρ, v_1 , root $x_{0,1}$

- 1: **Initialize:** $\bar{\mu}_0, \text{Var}_0, C^0 = 0, L^1 = \{x_{0,1}\}$
 - 2: **for** $t = 1, 2, \dots, T$ **do**
 - 3: Observe available agents M_t and the task t
 - 4: Construct arm contexts $x_{t,m}$ for each agent in M_t
 - 5: Compute indices according to (1) for each arm
 - 6: Select arm Set S^t based on indices and budget b
 - 7: Play arm Set S^t and observe rewards $\mu_{t,m}$
 - 8: Identify set of selected nodes \mathcal{P}^t
 - 9: **for** node $x_{h,i} \in \mathcal{P}^t$ **do**
 - 10: Update metrics for node
 - 11: **if** ((2) and (3)) or $c_\infty^t(x_{h,i}) \leq v_1 \rho^h$ **then**
 - 12: $L^{t+1} \leftarrow \text{split at } \bar{X}(x_{h,i}) \text{ on dimension } d^* \text{ (4)}$
 - 13: **end if**
 - 14: **end for**
 - 15: **end for**
-

4.2.5. Neural-CoCoMaMa

Instead of using $\bar{\mu}^t(x_{h,i})$ in the calculation of the index in Equation 1, we propose predicting the expected reward of an arm $\hat{\mu}^t(x_{t,m})$ using a neural net with a single hidden layer as in Neural-MAB [15], which learns each time an outcome is observed. The index is then defined per arm as $g(x_{t,m}) := \hat{\mu}^t(x_{t,m}) + c^t(x_{h,i}, p(x_{h,i}))$, where $x_{h,i}$ corresponds to the node the arm is in.

5. Evaluation

We evaluate the outlined algorithms on two datasets. The first dataset has been introduced by Somerstep et al. [19] and is enriched by agent cards in this work. It has a fixed size of general-purpose LLMs as agents and highlights the ability of the algorithms to identify and exploit the better-performing agents from a set of options with similar descriptions. For the second dataset, we add synthetic specialized agents and derive the performance based on a mathematical definition of a task-agent fit and the base-agent score from the first dataset [19]. The second experiment outlines the capabilities of the algorithms to identify and exploit specialized agents.

5.1. Routing on SPROUT

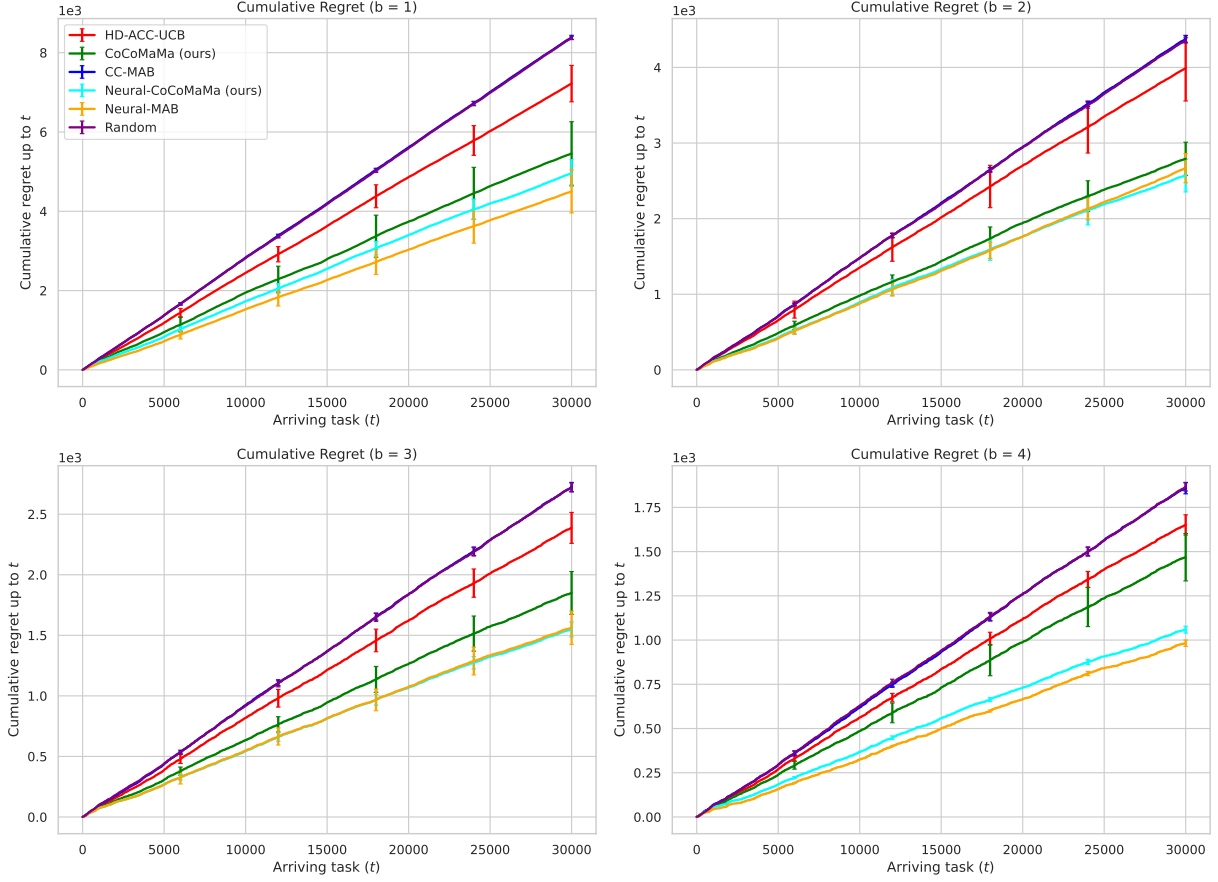


Figure 1: Cumulative regret over time on the SPROUT problem model for different budget (b) sizes. Displaying the average of 10 rounds with error bars displaying the standard deviation. Using $v_1 = \sqrt{5}$, $v_2 = 1$, $\rho = 0.9$, $\theta = 10$, for ACC-UCB and CoCoMaMa-based methods and "all-MiniLM-L6-v2" as the embedding model with 384 dimensions for task and agent card.

The SPROUT [19] dataset provides quality scores for answers from 13 different LLMs on over 40000 queries from 6 benchmarks. Agent cards for each model were created based on public announcements from the respective providers and are shown in Annex A. A random router and an oracle router, which always greedily selects the agents with the highest true mean in each round, are used as naive baselines. The HD-ACC-UCB, CC-MAB and Neural-MAB algorithms serve as the state-of-the-art baselines. The experiments are conducted 10 times each with the same sequential ordering of tasks for the budgets 1,2,3,4. The decisions made by the algorithms are compared with the optimal solution made by the oracle router. Making sub-optimal decisions yields regret and should be minimized.

The plots in Figure 1 show that CC-MAB yields the same regret as the random router. CoCoMaMa yields less cumulative regret than HD-ACC-UCB for all tested budgets. This supports our prior hypothesis that making statistically informed splitting decisions can increase performance. Neural-CoCoMaMa achieves even better results for all budgets, which could be attributed to a faster learning rate, as weights on all input dimensions of the neural net can be updated after each observation, while splitting of nodes only takes place under certain conditions. Furthermore, nodes are split on just one dimension. Neural-CoCoMaMa matches the performance of Neural-MAB for a budget of 2 and 3, and only shows a slightly higher regret for the other budgets.

The agent selection rates in Figure 2 show that Neural-MAB does not spend significant amounts to explore all agents and exploits the same 3 agents instead. This is indicated by the selection rates close to 1.0, where the sum of all selection rates should sum up to the budget 3. Always picking the

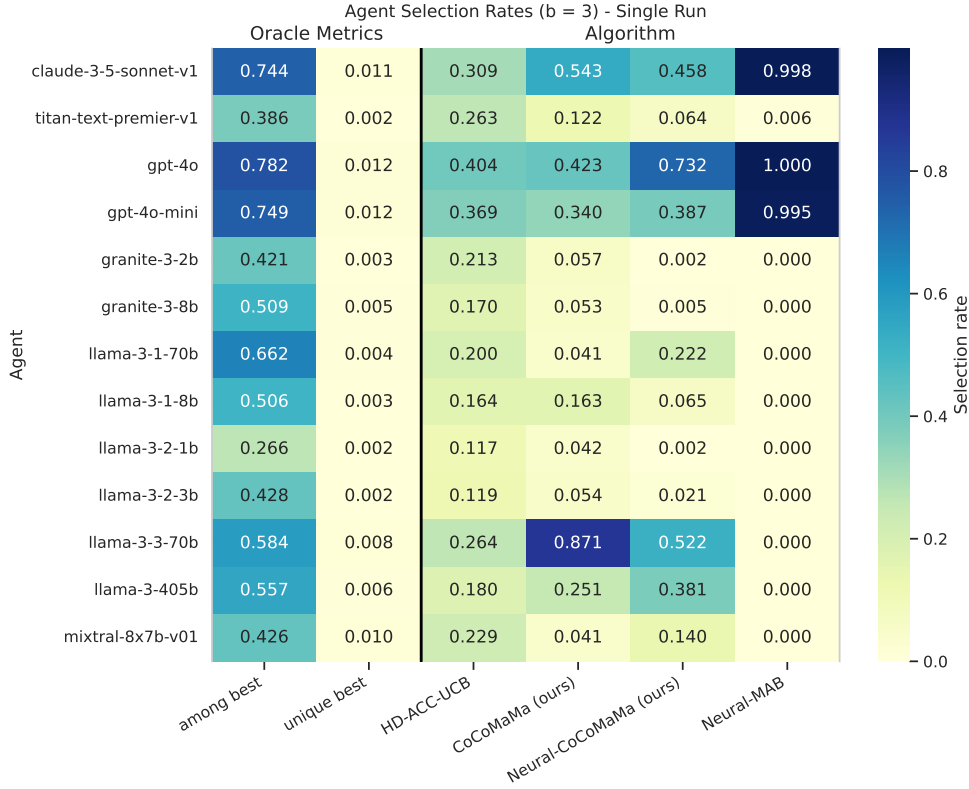


Figure 2: Agent selection rates for a single run on the SPROUT dataset using a budget of 3.

same 3 agents is not a bad strategy in this case, as each of them is among the best performing agents in over 70% of the tasks. In 1% of the tasks, Mixtral is the unique best-performing model, but it is never selected by Neural-MAB. All other algorithms follow a design where they are actively trying to explore cases where other models might perform better than their known best educated guesses. Our CoCoMaMa methods spend more effort on exploration than the greedy Neural-MAB, and provide a sharper distinction between good and bad performing agents compared to HD-ACC-UCB.

5.2. Routing on SPROUT with Specialized Experts

The SPROUT dataset does not contain many queries, where a unique best-performing agent can be identified and the average response quality of many models is high. This will most likely not be the case for highly specialized WebAgents. Therefore, we are adding synthetic specialized agents to the SPROUT dataset.

Let α denote the index of the task t , where we begin adding new specialized agents. If $t \geq \alpha$, a new agent is added every β rounds to all following sets of available agents M_t . If $t > \gamma$, the new agent is a strong expert, and a weak expert otherwise. A new expert is always based on a random base agent by copying their agent card embedding. The value at δ random dimensions is set to 1 for strong experts, and 0.9 for weak experts, to signal their specialization in certain areas. The possible expert dimensions are limited to 50% of the used dimensions from the embeddings. The true reward of an agent doing a task depends 80% on the task-agent fit and 20% on the base agent score. The task-agent fit $\mu_{t,m}^{taf}$ is computed based on matching the value q at the task embedding at the dimension with the highest value, with the respective value v at the same dimension at the agent card embedding using the following equation:

$$\mu_{t,m}^{taf} = \begin{cases} \sigma(5 \cdot q \cdot v^{100}) & \text{if } \sigma(5 \cdot q \cdot v^{100}) \geq 0.6 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

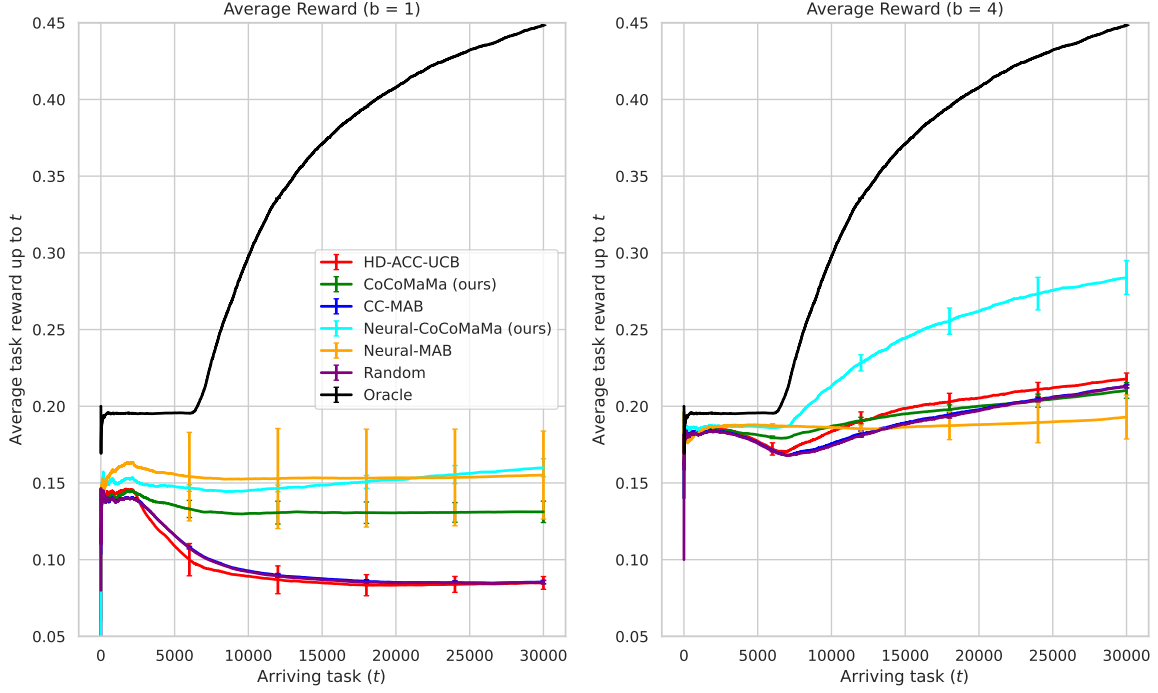


Figure 3: Average reward on the SPROUT with specialized experts dataset for different budget (b) sizes. Displaying the average of 10 rounds with error bars displaying the standard deviation. With $v_1 = \sqrt{5}$, $v_2 = 1$, $\rho = 0.9$, $\theta = 4$, for HD-ACC-UCB and CoCoMaMa based approaches. Using "all-MiniLM-L6-v2" as the embedding model with 50 dimensions for task and agent card.

, where σ denotes the logistic function. Using 0.9 for weak experts and v^{100} should mimic the behavior that innovative agents based on new technologies promising full integration are introduced and advertised, but they have flaws due to being early adopters. The second generation overcomes those issues. The base agent score is taken from the SPROUT dataset. It is multiplied by 0.1 for specialized agents in case the task-agent fit is below 0.6. Reducing the base agent score for specialists should mimic behavior, where an agent is tasked to answer "I don't know" on questions outside their domain. Duplicated agents in M_t are not permitted and the generation of a specialized agent is skipped for the respective round.

The experimental results using $\alpha = 2000$, $\beta = 200$, $\gamma = 6000$, $\delta = 5$ for expert generation are shown in Figure 3 displaying the average reward. For both budgets $b = 1$ and $b = 4$, the optimal average reward achieved by an oracle router increases continuously from below 0.2 to 0.45 after the strong experts are introduced at $t = 6000$. Many algorithms yield decreasing average rewards after the weak experts are introduced at $t = 2000$ and do not select strong experts for tasks in their respective domains often enough to show an increase in average reward for a budget of 1. However, as the chance rises to randomly pick a good task-agent match for a higher budget, all algorithms except Neural-MAB show increasing average performance after $t = 6000$ for the budget 4. The curves for Neural-CoCoMaMa show that it is resilient to the introduction of weak experts and is the best algorithm at exploring and exploiting the strong experts.

6. Discussion

The CoCoMaMa approach shows that statistically informed splitting of nodes can yield better results than the respective baseline HD-ACC-UCB. However, the learning rate is limited as we only split at one dimension each time. Neural-MAB is much faster at converging on a well-performing agent. Though it might become trapped at a local optimum and never escapes it due to a lack of exploration. Neural-CoCoMaMa combines efficient exploration and a fast learning rate. It outperforms all other

methods on SPROUT with synthetic agents. It matches the performance of the best-performing state-of-the-art algorithm, Neural-MAB, on the basic SPROUT dataset, while offering more explainability regarding the routing decision. Before letting an agent execute a task, the expected performance of the agent (in all neural-based approaches) and historical insights (in CoCoMaMa-based approaches and also HD-ACC-UCB and CC-MAB to a limited extent) can be communicated to the client. E.g., the CoCoMaMa approaches can provide historical variance, average performance and confidence scores for the context region of the task-agent pair. This allows operators to escalate tasks to humans before wasting resources on an agent when expected performance is low and highly variable.

A current limitation is that playing a good task-agent match at least once is required to start learning. Drastically decreasing the chance of finding a fit by increasing the amount of agents and making the required specializations more granular (more embedding dimensions, lower δ) would require a lot of data to train an efficient router. Providing better data in the agent cards may mitigate the problem. The agent cards already contain an example request. This is useful for humans, but it only resembles very small data points for a router. In the future, the agent developers could define entire context regions using hyperrectangles and provide respective average performance, confidence, and variance. Furthermore, different clients could share their aggregated reviews over the web. New federated learning approaches may then be used at the router to overcome cold starts and data scarcity.

Berners-Lee et al. [7] stress the importance of ontologies and linked data for knowledge representations in the Semantic Web. Ciortea et al. [13] hint at designing structured query capabilities to search for the matching agent in a web-scale hMAS containing billions of agents. The CoCoMaMa approach does not make using structured queries obsolete. Instead, those approaches can go hand in hand, as filtering using a query could drastically narrow down the action space per task and CoCoMaMa can then step in to decide on the best agent based on historical feedback. Future work could combine both approaches and further improve the results in experiments similar to Section 5.2. Learning based on the feedback could also uncover routing policies, which are not possible to identify using structured queries, as the data in the agent card might not be that informative. This can be observed in the experiments in Section 5.1.

Next, the A2A Protocol [10] does not specify how incurring costs should be communicated. Doing so would allow the router to balance between costs and performance. Using monetary budget constraints per task instead of budgets for the amounts of agents to play would open an interesting research area for routing in hMAS.

Lastly, the algorithms may be hardened on edge cases. E.g., an agent with underlying randomness might yield rewards with a high variance, causing the router to split often without getting any information gain. Variance-aware UCB Algorithms [46] are an active research domain and may also be incorporated to further improve the CoCoMaMa methods. Adversarial providers of agents might attempt to craft their agent cards to benefit from high expected rewards in already well-established domains to generate traffic for their mediocre services or badmouth competition. A good router should be robust enough to recover from such attacks. Transferring the insights from trust scores and digital signatures to agent routing in hMAS is also open future work.

7. Conclusion

In this work, we introduced CoCoMaMa and Neural-CoCoMaMa, two contextual combinatorial volatile multi-armed bandit approaches tailored for the dynamic and heterogeneous landscape of agentic LLMs. By leveraging task-agent similarities and online feedback, our methods address key limitations in existing routing strategies, particularly their reliance on static model pools and offline training data. Our approach is compatible with the A2A Protocol [10] and accommodates agent volatility through standardized agent cards, making it a promising fit for scalable, decentralized hMAS.

Experimental results on the SPROUT [19] dataset demonstrated equal performance to the best performing state-of-the-art method while improving the explainability of the routing decisions. Neural-CoCoMaMa is the only method capable of exploring and exploiting strong niche experts without

suffering as much from the introduction of weak experts to the pool of available agents in our second experimental dataset.

Importantly, our approach complements rather than replaces structured search methods. Combining CoCoMaMa with query-based filters could significantly reduce the action space, enabling efficient feedback-driven selection within semantically scoped agent sets. Additionally, integrating cost-awareness, variance-sensitive strategies, and trust mechanisms will be essential for deploying robust routing in open, adversarial, or resource-constrained environments.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 414984028 – SFB 1404 FONDA

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Grammarly in order to: Grammar and spelling check, paraphrase and reword. Further, the authors used perplexity.ai for agent cards in Annex A in order to: Drafting content. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

- [1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, in: International Conference on Learning Representations (ICLR), 2023.
- [2] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, et al., Metagpt: Meta programming for multi-agent collaborative framework, arXiv preprint arXiv:2308.00352 3 (2023) 6.
- [3] S. Marro, E. La Malfa, J. Wright, G. Li, N. Shadbolt, M. Wooldridge, P. Torr, A scalable communication protocol for networks of large language models, arXiv preprint arXiv:2410.11905 (2024).
- [4] J. He, C. Treude, D. Lo, Llm-based multi-agent systems for software engineering: Literature review, vision and the road ahead, ACM Transactions on Software Engineering and Methodology (2024).
- [5] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C. Qian, C.-M. Chan, Y. Qin, Y. Lu, R. Xie, et al., Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents, arXiv preprint arXiv:2308.10848 2 (2023) 6.
- [6] C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, D. Ha, The ai scientist: Towards fully automated open-ended scientific discovery, arXiv preprint arXiv:2408.06292 (2024).
- [7] T. Berners-Lee, J. Hendler, O. Lassila, Web semantic, Scientific American 284 (2001) 34–43.
- [8] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, arXiv preprint arXiv:1701.06538 (2017).
- [9] Y. Yang, H. Chai, Y. Song, S. Qi, M. Wen, N. Li, J. Liao, H. Hu, J. Lin, G. Chang, et al., A survey of ai agent protocols, arXiv preprint arXiv:2504.16736 (2025).
- [10] Google, A2A: Agent2Agent Protocol, <https://github.com/google/A2A>, 2025. Accessed: 2025-04-21.
- [11] Y. Kong, J. Ruan, Y. Chen, B. Zhang, T. Bao, S. Shiwei, D. Qing, X. Hu, H. Mao, Z. Li, et al., Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world industry systems, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, 2024, pp. 371–385.
- [12] S. G. Patil, T. Zhang, X. Wang, J. E. Gonzalez, Gorilla: Large language model connected with massive apis, Advances in Neural Information Processing Systems 37 (2024) 126544–126565.

- [13] A. Ciortea, S. Mayer, F. Gandon, O. Boissier, A. Ricci, A. Zimmermann, A decade in hindsight: the missing bridge between multi-agent systems and the world wide web, in: AAMAS 2019-18th International Conference on Autonomous Agents and Multiagent Systems, 2019, p. 5.
- [14] T. L. Lai, H. Robbins, Asymptotically efficient adaptive allocation rules, *Advances in applied mathematics* 6 (1985) 4–22.
- [15] T. Lu, D. Pál, M. Pál, Contextual multi-armed bandits, in: Proceedings of the Thirteenth international conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 485–492.
- [16] W. Chen, Y. Wang, Y. Yuan, Combinatorial multi-armed bandit: General framework and applications, in: International conference on machine learning, PMLR, 2013, pp. 151–159.
- [17] R. Kleinberg, A. Niculescu-Mizil, Y. Sharma, Regret bounds for sleeping experts and bandits, *Machine learning* 80 (2010) 245–272.
- [18] Z. Bnaya, R. Puzis, R. Stern, A. Felner, Volatile multi-armed bandits for guaranteed targeted social crawling., *AAAI (Late-Breaking Developments)* 2 (2013) 16–21.
- [19] S. Somerstep, F. M. Polo, A. F. M. de Oliveira, P. Mangal, M. Silva, O. Bhardwaj, M. Yurochkin, S. Maity, Carrot: A cost aware rate optimal router, *arXiv preprint arXiv:2502.03261* (2025).
- [20] Z. Chen, J. Li, P. Chen, Z. Li, K. Sun, Y. Luo, Q. Mao, D. Yang, H. Sun, P. S. Yu, Harnessing multiple large language models: A survey on llm ensemble, *arXiv preprint arXiv:2502.18036* (2025).
- [21] T. Shnitzer, A. Ou, M. Silva, K. Soule, Y. Sun, J. Solomon, N. Thompson, M. Yurochkin, Llm routing with benchmark datasets, in: *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.
- [22] D. Jain, T.-Y. Tung, T. H. Kofman, RoRF – Open Source LLM Router, <https://www.notdiamond.ai/blog/rorf>, 2024. Accessed: 2025-03-25.
- [23] S. Chen, W. Jiang, B. Lin, J. Kwok, Y. Zhang, Routerdc: Query-based router by dual contrastive learning for assembling large language models, *Advances in Neural Information Processing Systems* 37 (2024) 66305–66328.
- [24] Q. J. Hu, J. Bieker, X. Li, N. Jiang, B. Keigwin, G. Ranganath, K. Keutzer, S. K. Upadhyay, Routerbench: A benchmark for multi-llm routing system, *arXiv preprint arXiv:2403.12031* (2024).
- [25] Y. Li, Llm bandit: Cost-efficient llm generation via preference-conditioned dynamic routing, *arXiv preprint arXiv:2502.02743* (2025).
- [26] X. Wang, Y. Liu, W. Cheng, X. Zhao, Z. Chen, W. Yu, Y. Fu, H. Chen, Mixllm: Dynamic routing in mixed large language models, *arXiv preprint arXiv:2502.18482* (2025).
- [27] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, I. Stoica, Routellm: Learning to route llms from preference data, in: *The Thirteenth International Conference on Learning Representations*, 2024.
- [28] K. Lu, H. Yuan, R. Lin, J. Lin, Z. Yuan, C. Zhou, J. Zhou, Routing to the expert: Efficient reward-guided ensemble of large language models, *arXiv preprint arXiv:2311.08692* (2023).
- [29] D. Sikeridis, D. Ramdass, P. Pareek, Pickllm: Context-aware rl-assisted large language model routing, *arXiv preprint arXiv:2412.12170* (2024).
- [30] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., Judging llm-as-a-judge with mt-bench and chatbot arena, *Advances in Neural Information Processing Systems* 36 (2023) 46595–46623.
- [31] A. Szymanski, N. Ziemis, H. A. Eicher-Miller, T. J.-J. Li, M. Jiang, R. A. Metoyer, Limitations of the llm-as-a-judge approach for evaluating llm outputs in expert knowledge tasks, in: *Proceedings of the 30th International Conference on Intelligent User Interfaces*, 2025, pp. 952–966.
- [32] J. Zhang, Z. Huang, Y. Fan, N. Liu, M. Li, Z. Yang, J. Yao, J. Wang, K. Wang, Kabb: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems, *arXiv preprint arXiv:2502.07350* (2025).
- [33] Y. Xia, F. Kong, T. Yu, L. Guo, R. A. Rossi, S. Kim, S. Li, Convergence-aware online model selection with time-increasing bandits, in: *The Web Conference 2024*, 2024.
- [34] M. Hoveyda, A. P. de Vries, M. de Rijke, H. Oosterhuis, F. Hasibi, Aqa: Adaptive question answering in a society of llms via contextual multi-armed bandit, *arXiv preprint arXiv:2409.13447* (2024).

- [35] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: <https://arxiv.org/abs/1908.10084>.
- [36] N. Gupta, H. Narasimhan, W. Jitkrittum, A. S. Rawat, A. K. Menon, S. Kumar, Language model cascades: Token-level uncertainty and beyond, arXiv preprint arXiv:2404.10136 (2024).
- [37] M. Yue, J. Zhao, M. Zhang, L. Du, Z. Yao, Large language model cascades with mixture of thoughts representations for cost-efficient reasoning, arXiv preprint arXiv:2310.03094 (2023).
- [38] L. Chen, M. Zaharia, J. Zou, Less is more: Using multiple llms for applications with lower costs, in: Workshop on efficient systems for foundation models@ ICML2023, 2023.
- [39] J. Li, Q. Zhang, Y. Yu, Q. Fu, D. Ye, More agents is all you need, arXiv preprint arXiv:2402.05120 (2024).
- [40] B. Lv, C. Tang, Y. Zhang, X. Liu, P. Luo, Y. Yu, Urg: A unified ranking and generation method for ensembling language models, in: Findings of the Association for Computational Linguistics ACL 2024, 2024, pp. 4421–4434.
- [41] OpenAI, text-embedding-3 models, <https://platform.openai.com/docs/guides/embeddings>, 2024. Accessed: 2025-05-20.
- [42] L. Chen, J. Xu, Z. Lu, Contextual combinatorial multi-armed bandits with volatile arms and submodular reward, Advances in Neural Information Processing Systems 31 (2018).
- [43] A. Nika, S. Elahi, C. Tekin, Contextual combinatorial volatile multi-armed bandit with adaptive discretization, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 1486–1496.
- [44] S. Lin, Y. Yao, P. Zhang, H. Y. Noh, C. Joe-Wong, A neural-based bandit approach to mobile crowdsourcing, in: Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications, 2022, pp. 15–21.
- [45] A. A. Efanov, S. A. Ivliev, A. G. Shagraev, Welford’s algorithm for weighted statistics, in: 2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), IEEE, 2021, pp. 1–5.
- [46] Y. Han, X. Xu, On the precise asymptotics and refined regret of the variance-aware ucb algorithm, arXiv preprint arXiv:2412.08843 (2024).

A. Agent Cards

Three agent cards are shown below. They were built to resemble a typical agent card similar to the example given in the A2A specification [10]. They were not optimized to accomplish automated tool-calling [11] and may lack required information to make technically correct calls. All agent cards are included as JSON files in the uploaded supplementary material and will be made publicly accessible after acceptance.

Listing 1: Claude 3.5 Sonnet Agent Card

```

1 {
2   "name": "Claude 3.5 Sonnet",
3   "description": "Anthropic’s Claude 3.5 Sonnet model for advanced natural language
4     understanding and generation",
5   "url": "https://aws.amazon.com/bedrock/claude/",
6   "provider": {
7     "organization": "Anthropic via AWS",
8     "url": "https://aws.amazon.com/bedrock/"
9   },
10  "version": "3.5",
11  "documentationUrl": "https://docs.aws.amazon.com/bedrock/latest/userguide/model-
    parameters-claude.html",
  "capabilities": {

```

```

12     "streaming": true,
13     "pushNotifications": false,
14     "stateTransitionHistory": false
15 },
16 "authentication": {
17     "schemes": ["AWS Signature V4"]
18 },
19 "defaultInputModes": ["text/plain"],
20 "defaultOutputModes": ["text/plain", "application/json"],
21 "skills": [
22     {
23         "id": "text-generation",
24         "name": "Text Generation",
25         "description": "Generate coherent and contextually relevant text based on input
26             prompts",
27         "tags": ["text-generation", "conversation", "content-creation"],
28         "examples": [
29             "Write a summary of quantum computing for beginners",
30             "Draft an email to reschedule a meeting",
31             "Create a product description for a new coffee maker"
32         ]
33     },
34     {
35         "id": "reasoning",
36         "name": "Complex Reasoning",
37         "description": "Solve complex problems requiring multi-step reasoning",
38         "tags": ["reasoning", "problem-solving", "analysis"],
39         "examples": [
40             "Analyze the pros and cons of renewable energy sources",
41             "Help me troubleshoot why my code isn't working",
42             "Explain the economic implications of raising interest rates"
43         ]
44     }
45 ]

```

Listing 2: GPT-4o Agent Card

```

1 {
2     "name": "GPT-4o",
3     "description": "OpenAI's versatile, high-intelligence flagship model that accepts
4         both text and image inputs with a 128K context window",
5     "url": "https://platform.openai.com/docs/models/gpt-4o",
6     "provider": {
7         "organization": "OpenAI",
8         "url": "https://openai.com"
9     },
10    "version": "2024-11-20",
11    "documentationUrl": "https://platform.openai.com/docs/models/gpt-4o",
12    "capabilities": {
13        "streaming": true,
14        "pushNotifications": false,
15        "stateTransitionHistory": false
16    },
17    "authentication": {
18        "schemes": ["Bearer"]
19    },
20    "defaultInputModes": ["text/plain", "image/png", "image/jpeg"],
21    "defaultOutputModes": ["text/plain", "application/json"],

```

```

21 "skills": [
22   {
23     "id": "text-generation",
24     "name": "Text Generation",
25     "description": "Generate coherent and contextually relevant text based on input
26       prompts",
27     "tags": ["text-generation", "conversation", "content-creation"],
28     "examples": [
29       "Write a poem about autumn leaves",
30       "Draft a business proposal for a startup",
31       "Create a detailed product description"
32     ],
33   },
34   {
35     "id": "image-understanding",
36     "name": "Image Understanding",
37     "description": "Analyze and interpret images to provide relevant information and
38       insights",
39     "tags": ["vision", "image-analysis", "multimodal"],
40     "examples": [
41       "What's in this image?",
42       "Describe what you see in this chart",
43       "Help me understand what this diagram is showing"
44     ],
45     "inputModes": ["image/png", "image/jpeg", "text/plain"],
46     "outputModes": ["text/plain"]
47   },
48   {
49     "id": "function-calling",
50     "name": "Function Calling",
51     "description": "Generate structured JSON outputs for function calling and API
52       integrations",
53     "tags": ["api", "structured-output", "tool-use"],
54     "examples": [
55       "Extract the contact information from this email",
56       "Convert this description into a JSON product specification",
57       "Parse this resume data into structured format"
58     ],
59     "outputModes": ["application/json", "text/plain"]
60   },
61   {
62     "id": "reasoning",
63     "name": "Complex Reasoning",
64     "description": "Handle complex problem-solving tasks requiring multi-step
65       reasoning",
66     "tags": ["reasoning", "problem-solving", "analysis"],
67     "examples": [
68       "Solve this multi-step math problem",
69       "Help me debug this programming issue",
70       "Analyze the logical fallacies in this argument"
71     ]
72   }
73 ]
74 }

```

Listing 3: GPT-4o mini Agent Card

```

1 {
2   "name": "GPT-4o mini",

```



```

3  "description": "A more efficient and cost-effective version of GPT-4o with similar
4  capabilities",
5  "url": "https://platform.openai.com/docs/models",
6  "provider": {
7    "organization": "OpenAI",
8    "url": "https://openai.com"
9  },
10 "version": "1.0.0",
11 "capabilities": {
12   "streaming": true,
13   "pushNotifications": false,
14   "stateTransitionHistory": false
15 },
16 "authentication": {
17   "schemes": ["Bearer"]
18 },
19 "defaultInputModes": ["text/plain", "image/png", "image/jpeg"],
20 "defaultOutputModes": ["text/plain", "application/json"],
21 "skills": [
22   {
23     "id": "text-generation",
24     "name": "Text Generation",
25     "description": "Generate coherent and contextually relevant text based on input
26     prompts",
27     "tags": ["text-generation", "conversation", "content-creation"],
28     "examples": [
29       "Write a short story about time travel",
30       "Draft an email to a colleague about project updates",
31       "Create a product description for an e-commerce site"
32     ]
33   },
34   {
35     "id": "image-understanding",
36     "name": "Image Understanding",
37     "description": "Analyze and interpret images to provide relevant information",
38     "tags": ["vision", "image-analysis", "multimodal"],
39     "examples": [
40       "What objects are in this image?",
41       "Describe what you see in this photo",
42       "What text is shown in this screenshot?"
43     ],
44     "inputModes": ["image/png", "image/jpeg", "text/plain"],
45     "outputModes": ["text/plain"]
46   }
47 ]
48 }

```