PPAI: Enabling Personalized LLM Agent Interoperability for Collaborative Edge Intelligence

Zile Wang[1,*], Qianli Liu[1,*], Kaibin Guo[2], Haodong Wang[1], Jian Lin[1], Zicong Hong[1] and Song Guo[1]
[1]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong
[2]School of Software Engineering, Sun Yat-Sen University, China
zwangox@connect.ust.hk, qianli.liu@connect.ust.hk, guokb@mail2.sysu.edu.cn, hwanghb@connect.ust.hk,
jlindc@connect.ust.hk, congcong@ust.hk, songguo@cse.ust.hk

***Abstract*—Deploying large language model (LLM) on edge device enables personalized LLM agents for various users. The growing availability of diverse personalized agents presents a unique opportunity for peer-to-peer (P2P) collaboration, wherein each user can delegate tasks beyond the local agent's expertise to remote agents more suited for the specific query. This paper introduces PPAI, the first personalized LLM agent interoperability system, which enables users to collaborate with each other based on agent specialization. However, the ever-changing pool of agents and their interchangeable capacity introduce new challenges when it comes to matching queries to agents and balancing loads, compared with existing P2P systems. Therefore, we propose a scalable query-agent pair scoring mechanism based on prototypes to identify suitable agents within a P2P network with churn. Moreover, we propose a multi-agent interoperability Bayesian game to balance local demand and global efficiency, when changes in remote agent load occur too quickly to be observed. Finally, we implement a prototype of PPAI and demonstrate that it substantially broadens the range of tasks that could be carried out while maintaining load balance. On average, it achieves an accuracy improvement of up to 7.96% across multiple tasks, while reducing latency by 16.34% compared to the baseline.**

***Index Terms*—Large language model, agent interoperability, peer-to-peer computing, collaborative intelligence.**

I. Introduction

The rapid proliferation of AI-capable PCs, fueled by advances in on-device GPUs and dedicated AI accelerators, is reshaping the computing landscape. Modern consumer-grade devices now possess the capacity to run sophisticated large language models (LLM) locally, enabling the deployment of LLM-based agents at the edge [1]–[4]. To enable these LLM agents to handle specialized tasks or domains, users can personalize their local agent via fine-tuning [5]–[7] and retrieval-augmented generation (RAG) [8], [9]. While both approaches create opportunities for the flourishing of diverse personalized LLM-based agents at the edge.

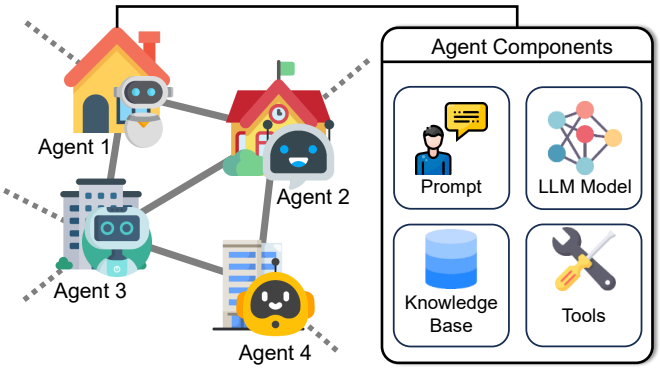


Fig. 1: Our vision for PPAI: Two decades ago, P2P networks (e.g. BitTorrent) allowed users to obtain files they did not have locally by downloading pieces from other users on the network. As agents advance, we envisage a future in which individuals can use others' personalized agents to complete tasks at which their own local agents are less proficient.

Given this growing ecosystem of diverse personalized LLM agents, exploring peer-to-peer (P2P) collaboration emerges as an increasingly compelling avenue to unlock their collective potential. P2P paradigms have been extensively studied across a range of applications, most notably in decentralized file sharing and data distribution. Classical systems [10], [11] enable efficient dissemination of large files by leveraging swarm-based chunk exchange among peers and distributed hash table (DHT) mechanism for scalable content lookup in vast unstructured networks. Later, Gnutella-inspired protocols and gossip-based overlays [12] have demonstrated robust information propagation and fault tolerance in dynamic environments. The principal advantage of these P2P approaches lies in their scalability and adaptability to heterogeneous nodes, enabling efficient information exchange across diverse, vast users.

P2P architecture presents an opportunity to harness complementary agent competencies across devices. This enables a paradigm that routes specific tasks suitable agents in the system, maximizing overall system utility by leveraging agent strengths. There are existing works performing agent routing for given inputs. RouterDC [13] establish a router based on

*Equal contribution.

This research was supported by fundings from the Hong Kong RGC General Research Fund (152169/22E, 152228/23E, 162161/24E, 162116/25E), Research Impact Fund (No. R5060-19, No. R5011-23), Collaborative Research Fund (No. C1042-23GF), NSFC/RGC Collaborative Research Scheme (Grant No. 62461160332 & CRS_HKUST602/24), Areas of Excellence Scheme (AoE/E-601/22-R), and the InnoHK (HKGAI). Corresponding authors: Zicong Hong, Song Guo.

dual-loss design, while KABB [14] achieve agent routing by building a knowledge graph. These methods, however, are not designed for P2P environments. Instead, they assume on-device co-location and focus on local model selection.

To achieve dynamic agent interoperability in a large-scale network, we propose a *P2P Agent Interoperability* (PPAI) system tailored for agent-to-agent collaborative task execution as shown in Figure 1. For example, when a user's local agent, specialized in legal knowledge, encounters a financial query, the user can delegate the task to another peer whose agent excels in finance. Our system dynamically matches queries to peers with both high task-specific performance and balanced load in fully decentralized environments. By establishing a P2P collaborative system where each user only deploys one specialized agent locally, these agents coordinate to solve a wider variety of tasks, effectively leveraging the collective expertise of the network with minimized resource footprints.

However, realizing this new system faces two key challenges: 1) **Personalized Agent Diversity & Churning**. Unlike traditional P2P networks that distribute uniform data, each agent in our system has distinct specialized competencies. Moreover, the agent pool churns as users may join, leave, or update over time. As the network scales to encompass a large number of participants with evolving agent set, determining which agent is best suited for a given query becomes increasingly complex and computationally demanding. 2) **Limited Agent Load Visibility & Congestion**. In highly active environments, users compete for limited agent resources. Because the system is large and query arrivals are dynamic, the real-time load of peer agents fluctuates rapidly and becomes unobservable to others. Since agents are functionally interchangeable, routing decisions made solely for local optimality could overload a subset of agents while others are underutilized. As the system scales, this lack of load visibility exacerbates congestion and degrades overall efficiency.

To address these challenges, we design query-agent pair scoring module for dynamic matching queries to churning agents and a Bayesian game to estimate the load at other edges. First, to tackle query-agent matching under churning, we establish a fixed latent space. By projecting both queries and agents to such space, the pair scoring can be done through anchor-based coordinates. Second, to mitigate the limited visibility of other agents' real-time load, we incorporate a probabilistic belief distribution derived from historical interactions, enabling agents to estimate peer load and route to substitutes when necessary. Our contributions are:

- We develop a scalable query-agent pair scoring mechanism based on prototype anchors, enabling users to efficiently determine suitable agents for their specific query locally, supporting churning agent sets.
- We formulate the routing and resource allocation problem as a Bayesian game. We introduce Cost of Delegation to analyze local utility maximization with latency estimation based on belief distribution. We demonstrate such system evolves towards a Bayesian Nash equilibrium.
- Our proposed system demonstrate evident improvement based on collaboration between small agents and improve average accuracy up to 7.96% on multiple tasks while reducing processing time by 16.34% than baseline method under high user demand simulation.

II. Related Work

A. LLM Agent

LLM agents are LLMs with equipped prompts, knowledge base and tools, becoming autonomous software entities that leverage the reasoning, planning, and interaction capabilities of LLMs to perceive their environment, make decisions, and execute actions toward achieving specific goals. Unlike traditional rule-based agents, LLM agents dynamically interpret user instructions, generate plans, and interact with external tools or other agents via natural language. Recent advances have demonstrated the potential of LLM agents across diverse application scenarios, including Mind2Web [15] for real-world websites tasks, SWE-agent [16] for automated software engineering, and MetaGPT [17] for multi-agent collaborative programming. These works illustrate the versatility of LLM agents and their ability to perform complex tasks through interaction with environments or cooperation among agents.

B. On-device LLM Agent

On-device LLM agents deployed on personal devices have become increasingly popular for delivering personalized AI services directly at the edge. These agents operating independently without constant server connectivity are ideal for latency-sensitive or privacy-critical scenarios. They preserve data privacy while benefiting from personalized AI capabilities. Recent work like Mobile-Agent-v2 [18] proposes mobile device operation assistance. MobileSteward [19] extends this paradigm by integrating multiple app-oriented agents coordinated by a steward agent. Meanwhile, MiniRAG [20] presents a lightweight RAG system designed for resource-constrained devices. These works highlight the potential of local agents to deliver efficient, context-aware, and personalized capabilities without relying on powerful cloud infrastructures.

C. Optimization on Agent Efficiency

1) Local Efficiency Optimization: On the system side, some efforts [21]–[23] improve throughput via pipeline parallelism, multi-GPU coordination and compiler-level tensor optimizations across heterogeneous hardware. Meanwhile, model compression methods based on quantization techniques such as GPTQ [24] achieves low-bit inference with minimal accuracy loss and knowledge distillation methods [25] compress models by teacher-student learning. These techniques are orthogonal to ours for shrinking the memory of specialized agents or accelerating inference within our P2P system.

2) Multi-agent Collaboration: To enable performance beyond single agent, multi-agent frameworks have emerged as a promising paradigm for combining specialized capabilities across multiple models. Agent routing approaches routes each query to the most suitable agent via contrastive

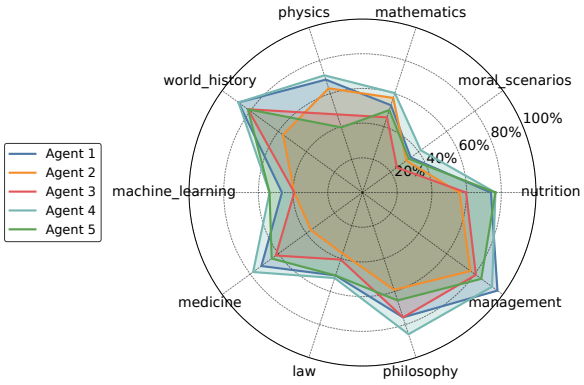


Fig. 2: Comparison of model accuracies across representative MMLU tasks.

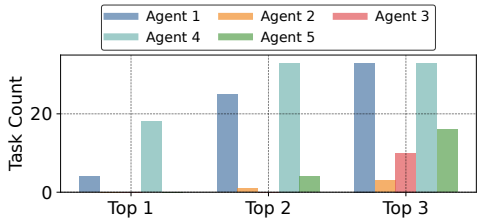


Fig. 3: Task counts where each agent ranks as the top-1, top-2, or top-3 performer across MMLU tasks.

training objectives [13], reward-based feedback loops [26], and semantic alignment through knowledge graphs [14]. Cost-aware variants [27] further adapt routing decisions based on predicted cost constraints. Other efforts achieve agent ensembling by post-ranking mechanisms [28], weighted probability fusion [29], or parameter merging [30]. However, they are not intended for P2P settings. They operate under the assumption that multiple models reside on a single device and aim to choose the most suitable local model for each input.

III. Motivation

In this section, we analyze the characteristics of personalized agents and present some findings that motivate our design.

Finding 1. *Agents deployed across devices exhibit strong specialization, suggesting the potential of decentralized cooperation to jointly fulfill diverse tasks.*

We first examine the distribution of task-specific strengths across several representative edge-deployable LLM agents. The performance of agents are shown in Figure 2, Agent 1–5 are respectively built on Qwen2.5-7B-Instruct [31], DeepSeek-R1-Distill-Qwen-7B [32], OpenCodeReasoning-Nemotron-7B [33], HuatuoGPT-o1-7B [34], and Meta-Llama-3-8B-Instruct [35]. Each agent is equipped with prior knowledge from conversation dataset ShareGPT [36]. These agents exhibit systematic differences in their competencies across diverse MMLU tasks. For example, Agent 4 shows pronounced advantages in medical tasks, reflecting its underlying finetuning

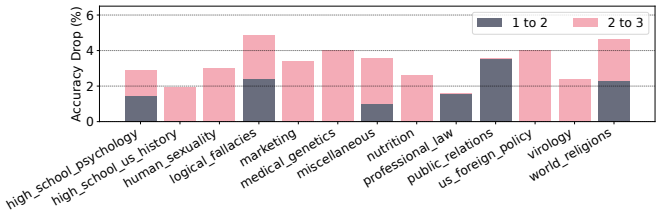


Fig. 4: Accuracy degradation across some tasks when selecting the second-best (1 to 2) and third-best (2 to 3) models instead of the top performer.

on healthcare datasets. Meanwhile, Agent 1 and 2 stand out on mathematical tasks. This complementary pattern indicates that while no single agent dominates universally, each agent provides expertise in a subset of tasks.Consequently, if different users independently deploy specialized agents on their own devices to meet personalized needs, the resulting distributed network could collectively achieve task coverage comparable to having all expertise locally, but without each device bearing the memory cost of hosting agents specialized in all domains.

Finding 2. *Naively selecting the best agent risks overloading a few nodes, especially in active user environments.*

Further examination shows that agent competencies are both specialized and highly imbalanced. Figure 3 quantifies how often each agent ranks as top-1, top-2, or top-3 across MMLU tasks, revealing that only a few agents dominate the top ranks while most rarely appear among the best. If queries are always routed to the top-performing agent, these few nodes would quickly become overloaded, increasing latency and creating congestion bottlenecks, especially under heavy traffic. It also suggests that expanding selection to include top-2 or top-3 agents substantially enlarges the candidate pool. By slightly relaxing strict optimality, the system can distribute queries more evenly, alleviating load concentration and improving robustness in large decentralized networks with heterogeneous users. This also increases overall system utilization.

Finding 3. *Routing queries to the second- or third-best agents can effectively alleviate system load while incurring only minimal performance degradation.*

Figure 4 illustrates the trade-offs when expanding agent selection beyond the top performer. It shows the accuracy drop on MMLU tasks when falling back from the best to the second-best agent ("1 to 2") and from the second- to the third-best ("2 to 3"). For many tasks, the drop from top-1 to top-2 remains under 5%, and in several tasks even the top-3 agent provides comparable performance. Although degradation may become steeper beyond top-2, routing some queries to the second- or third-best agents can effectively reduce load while incurring only minor performance loss. This finding highlights that modest relaxation of strict optimality can improve system

efficiency without severely compromising quality. Allocating queries among near-optimal agents helps prevent overload on highly specialized models, whereas excessive relaxation may still harm task accuracy. So, an effective routing strategy should leverage this trade-off to achieve both high performance and balanced system utilization.

These findings motivate our design of PPAI of scalable query-agent pair scoring to handle heterogeneous agents at scale, and congestion-aware multi-agent scheduling to optimize both local and global satisfaction.

IV. System Overview

Our PPAI system consists of a large network of users connected by P2P communication. With the rapid proliferation of open-source LLMs like Qwen [31] and LLaMA [35], users can locally deploy these models and adapt them to specific domains like law, medicine, or programming. As illustrated in Figure 5, each agent consists of system prompt, tool interfaces and specialized database to achieve personalized capability.

When a user issues a query, it can be served either by the user's local agent or by another agent in the network that is better suited for the task. To support such flexible and effective collaboration, our system routes each query to the most suitable agent across the network. Building on the limitations identified in prior work (§ II) and our findings in § III, we design the system to meet three key objectives.

• **Lightweight local deployment:** Each user only needs to maintain a single personalized agent on local device, while still being able to solve diverse tasks through agent interoperability.

• **Scalable and adaptive coordination:** Our system need to flexibly adapt to dynamic changes in the agent pool, such as agent joining, leaving, or updating, while maintaining scalable task coverage across diverse user requests.

• **Local and global efficiency balance:** The system aims to balance individual query performance with overall resource utilization by incorporating mechanisms that effectively estimate and account for real-time agent load conditions.

The core design of our system consists of two main components as illustrated in Figure 5. First, the query–agent pair scoring module (§ V) represents queries and agents in a shared latent space and computes their semantic compatibility. This abstraction enables scalable matching and naturally adapts to the agent pool with churning. Second, a multi-agent interoperability scheduler module (§ VI) formulates the scheduling as a Bayesian game complements relevance-based matching with real-time system state by Bayesian game to autonomously select the most suitable peer by balancing semantic relevance against current system congestion.

The system operates in a continuous loop driven by user queries and agent updates. When a new query arrives at a user, the query–agent pair scoring module first computes the semantic compatibility between the query and all agents, producing a shortlist of candidate agents. The multi-agent interoperability scheduler then refines this decision by incorporating load information and selecting the optimal agent that balances semantic relevance with current system congestion. Through this process, the system achieves effective query–agent matching while maintaining overall load balance.

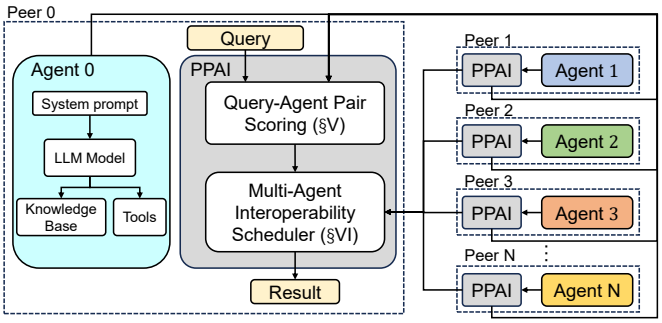


Fig. 5: Overview of PPAI system.

V. Prototype-Anchored Scalable Query-agent Pair Scoring

In this section, we introduce our prototype-anchored scalable query–agent pair scoring method. We first discuss the design rationale (§ V-A) and describe the architecture of the method (§ V-B). Then we introduce the pair scoring method (§ V-C) and agent capability updating under churning (§ V-D).

A. Design Rationale

Motivated by **Finding 1** in § III, we posit that collaboration among complementary agents can achieve high specialization and broad task coverage. As discussed in § II, conventional query–agent matching approaches typically train a centralized router model [13], [26]. By treating matching as classification, these methods implicitly assume a static agent pool. This rigidity renders them ill-suited for P2P environments, where frequent updates would necessitate prohibitively expensive retraining. Inspired by few-shot classification methods [37]–[39], which recognize new classes by computing prototype representations from few-shot labeled examples without retraining, we can solve query–agent routing by prototype representations. However, directly applying prototype learning is infeasible as the agent pool is large, heterogeneous, and dynamic. Assigning a dedicated prototype to each agent would create a fragmented and sparse representation space with poor generalization. To overcome this limitation, we construct a prototype-anchored semantic space into which both queries and agents are projected via QAGate in Figure 6. Such design allows us to compute compatibility scores between queries and agents based on prototype-anchored coordinates within a compact latent space, ensuring adaptability to churning.

B. Architecture and initialization

Projecting user queries and agents into a shared latent space requires reliable semantic alignment. To achieve this, we introduce QAGate, illustrated in Figure 6(b). It constructs a prototype-anchored abstraction to capture the inherent semantic diversity of user queries. Each query is first embedded into a dense semantic vector $q \in \mathbb{R}^d$ by a lightweight projector composed of a pretrained encoder followed by MLP. The

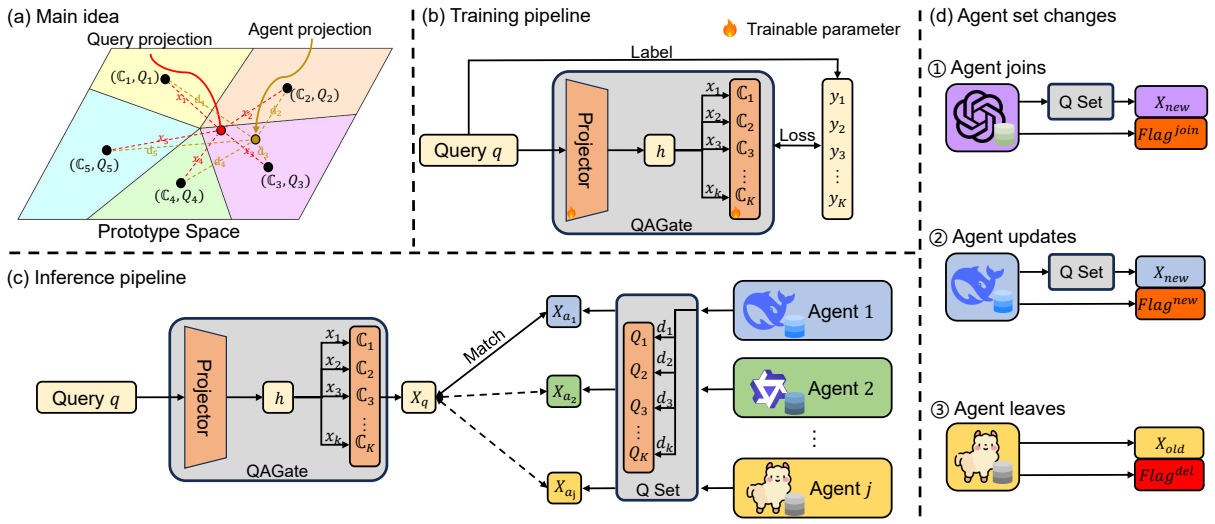


Fig. 6: Overview of our prototype-anchored framework for scalable query–agent scoring and matching. (a) Queries and agents are projected into a shared prototype space, where their coordinates relative to pretrained prototypes are computed to obtain relevance scores. (b) During training, the projector learns to map queries to prototype-anchored relevance vectors by labeled supervision. (c) At inference, QAGate produces a relevance vector for each query, which is then matched with the stored capability vectors of candidate agents. (d) The framework supports dynamic agent set changes including join, update, or leave the system by propagating updated capability vectors and associated flags, enabling scalable matching without retraining.

relevance vector is then the distance between the projection and a set of learned prototypes $\{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_K\}$.

The projector and prototypes are jointly trained by labeled queries collected from a broad range of tasks. These labels can be derived from domain-specific datasets, predefined task categories, or feedback by agents. For instance, a label may correspond to the predefined task category that a query belongs to. Rather than assigning each query to a single prototype, we explicitly models the nuanced associations between a query and multiple prototypes. Accordingly, we train the projector to produce a relevance vector that approximates a soft target distribution $\hat{p}$ over prototypes. The training objective minimizes the Kullback–Leibler (KL) divergence between y and the softmax-normalized prediction $x = f(q;\theta)$ as:

$$\mathcal{L}_{\mathrm{KL}}(q) = \mathrm{KL}\big(y \,\|\, \mathrm{softmax}(f(q;\theta))\big). \tag{1}$$

C. Query-agent pair scoring

Inference stage is shown in Figure 6(c), we perform ℓ_2 normalization on $h(q)$ and each prototype c_k, obtaining unit-norm vectors. The relevance score for prototype c_k is computed as:

$$f(q;\theta)[k] = \frac{h(q)^\top \mathbb{C}_k}{\|h(q)\|_2 \, \|\mathbb{C}_k\|_2}, \quad k = 1, \ldots, K. \tag{2}$$

To enforce sparsity and selectivity, we apply a normalization operation to Equation 2. We raise each component to a power α to amplify dominant affinities, retain the top-p fraction of prototypes $\mathrm{Top}_p \subset K$ as:

$$\tilde{f}(q)[k] = \begin{cases} \dfrac{(f(q;\theta)[k])^\alpha}{\sum_{j \in \mathrm{Top}_p} (f(q;\theta)[j])^\alpha}, & \text{if } k \in \mathrm{Top}_p, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

To represent agent capabilities in the same space, each agent M_j maintains a profile that quantifies its performance (i.e. accuracy) across the K prototypes as a distance d_k to them. Specifically, we evaluate M_j on held-out validation data $\{Q_1, Q_2, \ldots, Q_K\}$, yielding a capability vector:

$$p_j = [p(j,1), p(j,2), \ldots, p(j,K)] \in \mathbb{R}^K, \tag{4}$$

where each component $p(j,k)$ reflects the proficiency of agent M_j on tasks associated with prototype c_k.

When an user i receives a new query q, it obtains the masked relevance vector $\tilde{f} = \tilde{f}(q;\theta) \in \mathbb{R}^K$ as described in Equation 3. The suitability of agent M_j for handling this query is then quantified by the cosine similarity between $\tilde{f}$ at user i and p_j:

$$s(i,j) = \frac{(\tilde{f})^\top p_j}{\|\tilde{f}\|_2 \, \|p_j\|_2}. \tag{5}$$

D. P2P capability updating

To enable QAGate in dynamic environment, it is crucial to maintain a consistent and up-to-date global view of agent capabilities. As illustrated in Figure 6(d), three types of agent set changes are handled through gossip-style propagation. When an agent is updated, it recomputes its capability vector using the same query set and broadcasts the update with the timestamp and an update flag. When a new agent joins the system, it computes its initial capability vector and broadcasts it with a join flag and the timestamp. Conversely, when an agent leaves, it broadcasts its identifier with a delete flag. To ensure rapid network-wide consistency, these messages are propagated using a gossip protocol, where each recipient further disseminates to a random subset of peers [40].

VI. Multi-Agent Interoperability Bayesian Game

To handle unobservable and fluctuating agent loads, we introduce interoperability Bayesian game in this section. It leverages belief distributions to guide routing and flexibly delegate tasks to interchangeable agents.

A. System Model

1) User Set: We consider a P2P network comprising a set of N users $\mathcal{U} = \{1, 2, \ldots, N\}$. Each user j locally hosts a personalized agent M_j, resulting in heterogeneous specializations across the network. Each user independently decide how to process each incoming query based on locally maintained information. Each agent M_j has a private type $\theta_j = (\mu_j, \lambda_j)$, where μ_j denotes its service rate and λ_j its query arrival rate. The real-time θ_j is not observable to other agents due to frequent and dynamic changes.

2) Belief Distribution: Since real-time θ_j is unobservable, each user i maintains a *belief distribution* $b_i(\theta_j)$ over the type space Θ_j. This belief reflects user i's estimation of agent M_j's type. Whenever user i receives state feedback $\omega_j = (\lambda_j, \mu_j)$ from M_j, the belief is updated using Bayes' rule:

$$b_i^{\text{new}}(\theta_j) = \frac{P(\omega_j|\theta_j) b_i(\theta_j)}{\sum_{\theta'_j \in \Theta_j} P(\omega_j|\theta'_j) b_i(\theta'_j)}, \tag{6}$$

where $P(\omega_j|\theta_j)$ is the likelihood of observing ω_j under θ_j.

3) Serving Strategy: When user i receives a query q_i, it must decide whether to process it locally or delegate it to another agent. The semantic relevance between q_i and agent M_j is quantified by the prototype-anchored score $s(i, j)$ computed by Equation 5 in § V. Based on its belief distribution $b_i(\theta_j)$, user i selects an assignment variable $z_i \in \mathcal{J}$, where $\mathcal{J} = \{1, 2, \ldots, N\}$ is the set of all available agents and $z_i = j$ indicates that q_i is delegated to agent M_j.

4) System Cost: To capture congestion effects, we estimate the expected queue load of agent M_j as

$$\mathbb{E}_{\theta_j \sim b_i}[\rho_j] = \min\big(1,\, \rho_j + \delta(\lambda_j - \mu_j)\big), \tag{7}$$

where $\rho_j = \lambda_j/\mu_j$ is the current utilization, λ_j and μ_j are the estimated arrival and service rates under $b_i(\theta_j)$, and $\delta > 0$ controls the prediction horizon.

Using Equation 7, the *Cost of Delegation (CoD)* for assigning a query from user i to agent M_j is defined as

$$c(i, j) = \mathbb{E}_{\theta_j \sim b_i}[\rho_j] + t_j^{\text{infer}} + t_{(i,j)}^{\text{trans}}, \tag{8}$$

where t_j^{infer} is the expected inference time of M_j, and $t_{(i,j)}^{\text{trans}}$ is the expected communication latency between user i and M_j.

B. Problem Formulation

Given the relevance score $s(i, j)$ and the cost of delegation $c(i, j)$, the utility of assigning q_i to agent M_j is

$$U(i, j) = s(i, j) - \beta\, c(i, j), \tag{9}$$

where $\beta > 0$ controls the trade-off between semantic alignment and congestion. The decision-making problem is thus formalized as:

$$z_i = \arg\max_{j \in \mathcal{J}} U(i, j). \tag{10}$$

Algorithm 1: Multi-Agent Interoperability at User i

```
Input: Query q_i, belief set {b_i(θ_j)}
Compute prototype relevance s(i, j) for all j
Form candidate set E_i = {j | s(i, j) ≥ θ_s}
Request (λ_j, μ_j, t_j^infer, t_(i,j)^trans) from j ∈ E_i
Update beliefs b_i(θ_j) via Bayes' rule
for j ∈ E_i do
    Compute CoD c(i, j)
    Compute expected utility U(i, j)
end for
Select z_i = arg max_{j∈E_i} U(i, j)
Delegate q_i to M_{z_i} or execute locally if z_i = i
```

This naturally defines a *Bayesian Game* $\mathcal{G} = \langle \mathcal{U}, \mathcal{J}, \{U(i, j)\}, \{\Theta_j\}, \{b_i\} \rangle$ where each user i is a player, θ_j is the private type of agent M_j, and $b_i(\theta_j)$ is the belief distribution. A *Bayesian Nash Equilibrium* (BNE) is reached when all users adopt strategies that are mutual best responses given their beliefs $b_i(\theta_j)$.

C. Algorithm Design

We propose algorithm for multi-agent interoperability Bayesian game, as summarized in Algorithm 1. It combines prototype-anchored semantic relevance with Bayesian belief updates to make routing decisions under incomplete information. Upon receiving a new query q_i, user i obtain query-agent pair scores $s(i, j)$ from § V, forming a candidate set as

$$\mathcal{E}_i = \big\{ j \in \mathcal{J} \,\big|\, s(i, j) \geq \theta_s \big\}, \tag{11}$$

where $\theta_s > 0$ is a relevance threshold.

For each candidate agent $M_j \in \mathcal{E}_i$, user i requests state information $(\omega_j, t_j^{\text{infer}}, t_{(i,j)}^{\text{trans}})$ to update its belief distribution $b_i(\theta_j)$ via Equation 6. Using the updated beliefs, it computes the expected queue load (Equation 7), derives the CoD $c(i, j)$ as defined in Equation 8, and calculates the expected utility $U(i, j)$ (Equation 9). Finally, user i selects the agent that maximizes $U(i, j)$ as in Equation 10.

D. Analysis

We analyze the proposed multi-agent interoperability Bayesian game by establishing fundamental properties of the best response, potential function, equilibrium existence and uniqueness, efficiency bounds and belief convergence.

Lemma 1. *Given a fixed candidate set $\mathcal{E}_i$ and belief distribution $b_i(\theta_j)$, user i's best response $z_i^\star = \arg\max_{j \in \mathcal{E}_i} U(i, j)$ always exists and is unique.*

Proof. For each $j \in \mathcal{E}_i$, $U(i, j)$ is finite since both $s(i, j)$ and $c(i, j)$ are bounded under $b_i(\theta_j)$. Because $\mathcal{E}_i$ is a finite set, there exists at least one $j^\star$ that maximizes $U(i, j)$. A deterministic tie-breaking rule (e.g., selecting the smallest index) guarantees uniqueness of $z_i^\star$. □

Lemma 2. *For any user i and agent M_j, the utility $U(i, j)$ decreases as the expected queue load $\mathbb{E}_{\theta_j \sim b_i}[\rho_j]$ increases.*

Proof. From Equation 9 and Equation 8, we have

$$\frac{\partial U(i,j)}{\partial \mathbb{E}_{\theta_j \sim b_i}[\rho_j]} = -\beta < 0, \tag{12}$$

where $\beta > 0$. Hence $U(i,j)$ is strictly decreasing in the expected queue load. □

Theorem 1 (Existence and Uniqueness of Bayesian Nash Equilibrium)**.** *Define the global potential function*

$$\Phi(z) = \sum_{i=1}^{N} U(i, z_i), \tag{13}$$

where $z = (z_1, \ldots, z_N)$ is the joint strategy profile. The multi-agent interoperability Bayesian game is an exact potential game with potential function $\Phi(z)$ and thus admits at least one Bayesian Nash Equilibrium (BNE). Furthermore, if $U(i,j)$ is strictly concave in the mixed-strategy space and users' beliefs $b_i(\theta_j)$ converge to consistent distributions, the BNE is unique.

Proof. Consider any unilateral deviation of user i from z_i to z_i'. The change in the potential function is

$$\Phi(z_i', z_{-i}) - \Phi(z) = U(i, z_i') - U(i, z_i), \tag{14}$$

where z_{-i} denotes the strategy profile of all users except i. Because the change in $\Phi(z)$ equals the change in user i's utility, the game is an exact potential game. By the fundamental property of finite exact potential games, the game always admits at least one pure-strategy Nash equilibrium, which corresponds to a BNE in the original Bayesian setting via Harsanyi's transformation.

If $U(i,j)$ is strictly concave in the mixed-strategy space and users' beliefs $b_i(\theta_j)$ converge to consistent distributions, then each user has a unique best response (Lemma 1), and $\Phi(z)$ has a unique maximizer. Hence, the BNE is unique. □

Lemma 3. *Under sequential best-response updates, users' strategies converge to a pure-strategy BNE in finite steps.*

Proof. By Theorem 1, the game is a finite exact potential game. Each best-response update strictly increases $\Phi(z)$. Because the strategy space is finite, the process must terminate after a finite number of steps at a pure-strategy BNE. □

Theorem 2 (Bounded Bayesian Price of Anarchy)**.** *For affine CoD in Equation 8, the Bayesian Price of Anarchy (BPoA),*

$$\mathrm{BPoA} = \frac{\max\limits_{z \in BNE} \mathbb{E}_\theta[\Phi(z)]}{\min\limits_{z} \mathbb{E}_\theta[\Phi(z)]}, \tag{15}$$

is upper bounded by $\frac{5}{3}$.

Proof. Conditioned on each type realization θ, the game reduces to a singleton congestion game with affine cost functions. By the result of Roughgarden and Tardos [41], the PoA for such games is at most $\frac{5}{3}$. Taking the expectation over θ preserves this bound, yielding $\mathrm{BPoA} \leq 5/3$. □

Theorem 3 (Belief Convergence)**.** *Assume that feedback ω_j is repeatedly observed and satisfies the conditional independence assumption $P(\omega_j^t|\theta_j) = P(\omega_j|\theta_j)$ for all t. Then, for any user i and agent M_j, the belief distribution $b_i^t(\theta_j)$ updated by Bayes' rule converges almost surely to the true posterior distribution $P(\theta_j|\omega_j^{1:t})$ as $t \to \infty$.*

Proof. At each time step t, user i updates its belief about agent M_j's type θ_j according to Bayes' rule in Equation 6. Because the likelihood $P(\omega_j|\theta_j)$ is correctly specified and the observations $\{\omega_j^t\}_{t=1}^{\infty}$ are i.i.d. conditional on θ_j, the updating sequence satisfies the martingale property with respect to the filtration generated by past observations. For any t,

$$\mathbb{E}[\, b_i^t(\theta_j) \mid \omega_j^{1:t-1} \,] = b_i^{t-1}(\theta_j). \tag{16}$$

By Doob's martingale convergence theorem [42], this bounded martingale converges almost surely to a limit $b_i^{\infty}(\theta_j)$ as $t \to \infty$. Since the likelihood is correctly specified and the parameter space for θ_j is assumed to be well-behaved (e.g., finite or satisfying standard regularity conditions), classical Bayesian consistency results imply that the posterior distribution concentrates on the true type. Formally,

$$b_i^t(\theta_j) \to P(\theta_j|\omega_j^{1:t}) \quad \text{a.s. as } t \to \infty, \tag{17}$$

and $P(\theta_j|\omega_j^{1:t})$ itself converges to a degenerate distribution at the true θ_j. Hence, $b_i^t(\theta_j)$ converges almost surely to the correct posterior distribution as claimed.

□

VII. Discussion

While our work demonstrates the feasibility and effectiveness in P2P environments, several open challenges remain for future research. One critical direction is to enhance the security and robustness of the system in open P2P settings. In realistic deployments, malicious agents may perform denial-of-service attacks by issuing a large number of meaningless queries, deliberately return incorrect outputs, or attempt to reconstruct private data via model inversion or membership inference attacks through adaptive querying. These vulnerabilities could fundamentally compromise the integrity of collaborative inference. Addressing these threats will require incorporating mechanisms for trust and reputation management, anomaly detection, and rate-limiting policies to mitigate the influence of unreliable or adversarial agents.

Another important direction is to design effective incentive mechanisms to encourage sustainable and fair collaboration. In practice, the system faces the classic 'free-rider' problem, where rational agents may consume network resources without reciprocal contribution, leading to a tragedy of the commons scenario. Therefore, ensuring that active cooperation becomes the dominant strategy for rational agents is paramount. Future work could explore game-theoretic or blockchain-based incentive schemes that reward agents based on their contributions and reliability, as well as reputation-driven strategies that prioritize cooperative participants.

TABLE I: Accuracy (%) and process time (ms) across tasks.

	MMLU		GSM8K		MedQA		ARC-C		AGIEval	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
Single agent										
Agent 1	71.24	22.70	64.54	226.80	46.94	56.00	85.32	15.47	61.05	52.44
Agent 2	54.43	22.70	20.91	290.63	19.62	50.15	17.75	15.47	12.75	53.85
Agent 3	63.02	25.16	33.33	290.63	37.21	50.83	68.94	17.65	48.38	68.07
Agent 4	72.91	22.70	33.64	275.94	50.86	50.38	86.69	15.88	62.13	53.85
Agent 5	64.01	22.70	59.39	306.50	58.08	49.70	74.74	15.88	39.18	53.61
Multi-agents										
Voting	70.87	25.16	59.39	306.50	52.12	50.83	87.03	17.65	62.06	68.07
RouteDC [13]	64.98	19.56	47.58	103.24	46.94	49.48	85.67	15.13	61.36	43.81
KABB [14]	72.29	22.40	37.27	287.25	53.53	42.67	86.35	15.96	60.20	49.64
Ours	72.94	21.05	64.55	247.31	57.46	50.37	86.01	14.69	62.62	42.12

VIII. Experiment

A. Implementation

We build our system on top of vLLM[1], a fast and easy-to-use library for LLM inference, and libp2p[2], an open-source networking library for P2P communication. For QAGate, we adopt the paraphrase-multilingual-mpnet-base-v2 model [43] followed by a two-layer MLP with ReLU activation for sentence encoding. To train QAGate for knowledge alignment, we apply AgglomerativeClustering [44] to partition the training samples into 20 clusters, which serve as category labels. After conducting multiple rounds of hyperparameter search, we retain the best-performing configuration.

B. Experimental Setup

Agent setup. We evaluate our system using multiple heterogeneous LLM agents with distinct task specializations. Selected 5 agents each includes a LLM model: 1) Agent 1: Qwen2.5-7B-Instruct [31] is finetuned for coding and mathematics abilities, 2) Agent 2: DeepSeek-R1-Distill-Qwen-7B [32] is distilled from DeepSeek-R1 based on Qwen, 3) Agent 3: OpenCodeReasoning-Nemotron-7B [33] is post-trained for reasoning for code generation, 4) Agent 4: HuatuoGPT-o1-7B [34] is designed for advanced medical reasoning and 5) Agent 5: Meta-Llama-3-8B-Instruct [35] is optimized for dialogue use cases. Also, each agent is provided prior knowledge from dataset of conversations collected from ShareGPT [36]. These agents are based on various structures and fine-tuned for different usage. To evaluate the scalability of our PPAI system, we establish 50, 100, 500, 1000 agents with these 5 models with different set of conversation knowledge.

Workloads. Requests data are drawn from these datasets: 1) `MMLU` [45] is a benchmark that covers 57 general tasks, 2) `GSM8K` [46] consists of grade school math word problems, 3) `MedQA` [47] contains questions from medical exams, `ARC-C` [48] contains grade-school level science questions and `AGIEval` [49] contains 18 tasks from general exams like Gaokao, GRE and SAT. We combine data from `MMLU`, `GSM8K`, `MedQA` and `ARC-C` and allocate 75% of examples to train the prototype-gating network and 25% as user queries, while `AGIEval` serves as an out-of-distribution (OOD) benchmark with rich categories of tasks. The user queries arrival follow a Poisson distribution with different arrival rates to simulate system performance under heavy demand.

Node setup. The agents run on Xeon® Gold 6226 CPU and A6000 GPU with 45GB of memory. We consider each user settled cross different regions in a country. Within each region, the delay is 5 ms and bandwidth is 2Gbps.

Baselines. We compare our system against two types of works, local agent serving and multi-agents serving. There are three baselines of multi-agents serving: 1) Majority voting [50] is gathering outputs of all agents based on voting, 2) RouterDC [13] trains a central router to predict suitable agent, 3) KABB [14] construct a knowledge graph and assign queries to pre-defined expert agents. Although these approaches are inherently centralized, we implement their routing strategies within our system for a fair comparison.

Metrics. We primarily measure *average accuracy* across all queries as the key indicator of task coverage and routing quality. Additionally, we report *average process time*, quantified by the end-to-end processing time per query, incorporating queuing delays and model inference times. We report each result as the mean of 8 independent runs to mitigate randomness.

C. Results

Table I summarizes the comparative performance of our system against both single-agent serving and multi-agent baselines under arrival rate of $\lambda = 10$. Majority voting is robust and stable by aggregating outputs from all agents, but suffers from high latency due to issuing redundant queries. RouterDC achieves strong accuracy on certain tasks but exhibits inconsistent results on datasets like GSM8K and MedQA, reflecting the limitations of its centralized router design. KABB delivers performance comparable to ours on many benchmarks but suffers a notable drop on GSM8K, likely because it relies on manually defined task-expert associations that do not generalize well. In contrast, our approach attains the highest average accuracy across most tasks while substantially reducing inference latency compared to baseline. This demonstrates the effectiveness of PPAI, which enable more precise and consistently reliable query-to-agent assignments.

[1]https://github.com/vllm-project/vllm

[2]https://libp2p.io/

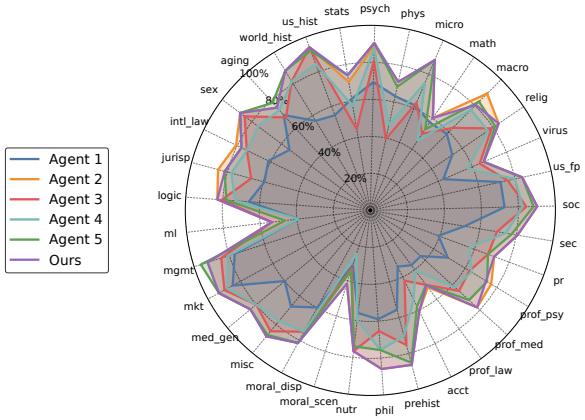


Fig. 7: Comparison of candidate models and our method's accuracies across all MMLU tasks.

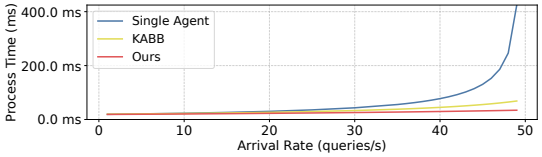
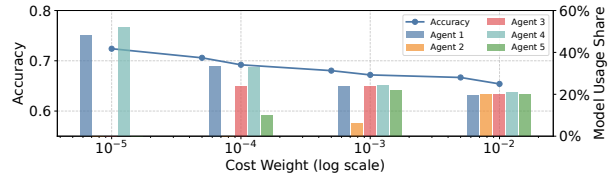


Fig. 8: Comparison of processing time under varying arrival rate λ on MMLU.

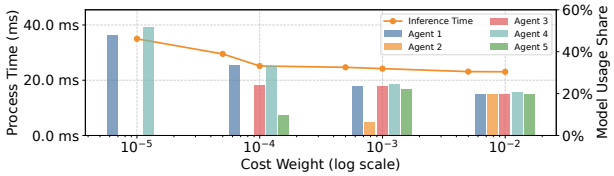
Figure 7 visualizes the accuracy distribution across all MMLU task categories, providing a fine-grained perspective on how each approach generalizes over diverse domains. Compared to individual agents, our method maintains a uniformly high level of accuracy across nearly all categories. This balanced performance pattern highlights that our PPAI method achieves consistently optimal or near-optimal accuracies across a broad spectrum of tasks.

While KABB shows comparable performance to our PPAI, we further evaluate their ability on heavier workload of user request. Figure 8 examines system behavior with different arrival rate λ on MMLU. We observe that the single-agent configuration rapidly becomes a bottleneck: as λ increases beyond 40, the processing time escalates sharply due to queuing delays at the overloaded agent. Our method consistently achieves the lowest overall latency, reflecting its more adaptive routing mechanism under interchangeable agent sets.

Figure 9 investigates how varying the cost hyperparameter β, which controls sensitivity to queuing delays, affects overall system behavior on MMLU. The arrival rate λ is set to 50 to reflect the load balancing ability, The overlaid bar plots reveal how the query distribution across the five agents evolves. As shown in Figure 9a, increasing β from 10^{-5} to 10^{-2} leads to a steady decline in average accuracy and more even distribution of queries. While Figure 9b shows that higher β significantly lowers average processing time. Varying β introduces a trade-off between accuracy and congestion, as routing shifts from highly specialized but busy agents to less loaded substitutes. This demonstrates that PPAI can adaptively delegate tasks to alternative agents under high and partially unobservable loads.



(a) Accuracy.



(b) Process time.

Fig. 9: Accuracy and process time on MMLU with varying β.

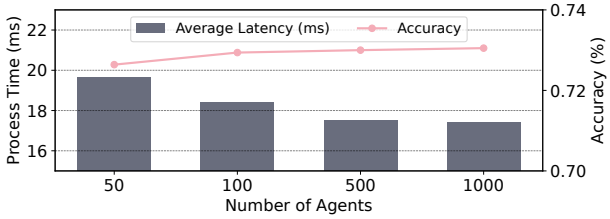


Fig. 10: Time and accuracy on MMLU under various scales.

Figure 10 shows the impact of scaling the number of agents on processing time and accuracy when λ is set to 100. As the number of agents increases from 50 to 1000, the average processing time decreases notably due to improved load distribution across a larger pool of agents. Meanwhile, accuracy remains largely stable, indicating that the routing mechanism can effectively utilize interchangeable agents without sacrificing task-specific performance. These results confirm that PPAI is effective with a large scale of agents, achieving lower processing time and stable accuracy.

IX. Conclusion

In this paper, we present PPAI for collaborative edge intelligence in P2P environments. We address the challenges of diverse, churning and interchangeable agents and limited load visibility. By introducing a prototype-anchored query–agent pair scoring mechanism, our system enables each user to dynamically match queries to suitable agents within a heterogeneous and churning agent pool. Also, our multi-agent interoperability Bayesian game incorporates belief-based load estimation to guide routing decisions and flexibly delegate tasks to substitute agents. Our experiments demonstrate that this approach significantly expands task coverage, improves routing accuracy, and maintains low latency under diverse workloads. Also, our system provides an interface that supports both broad and domain-specific task allocation, while allowing flexible integration of different matching mechanisms according to specific requirements.

REFERENCES

[1] Z. Lu, X. Li, D. Cai, R. Yi, F. Liu, W. Liu, J. Luan, X. Zhang, N. D. Lane, and M. Xu, "Demystifying small language models for edge deployment," in *ACL*, 2025.

[2] P. Belcak, G. Heinrich, S. Diao, Y. Fu, X. Dong, S. Muralidharan, Y. C. Lin, and P. Molchanov, "Small language models are the future of agentic ai," *arXiv preprint arXiv:2506.02153*, 2025.

[3] H. Wang, Q. Zhou, Z. Hong, and S. Guo, "D^2moe: Dual routing and dynamic scheduling for efficient on-device moe-based llm serving," *MobiCom*, 2025.

[4] Z. Fang, Y. Huang, Z. Hong, Y. Lyu, W. Chen, Y. Yu, F. Yu, and Z. Zheng, "Klotski: Efficient mixture-of-expert inference via expert-aware multi-batch pipeline," in *ASPLOS*, 2025.

[5] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *EMNLP*, 2021.

[6] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." in *ICLR*, 2022.

[7] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," in *NeurIPS*, 2022.

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *NeurIPS*, 2020.

[9] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," in *ICLR*, 2023.

[10] B. Cohen, "Incentives build robustness in bittorrent," in *P2P Econ*, vol. 6, 2003, pp. 68–72.

[11] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS*, 2002, pp. 53–65.

[12] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," in *TOCS*, vol. 23, no. 3, 2005, pp. 219–252.

[13] S. Chen, W. Jiang, B. Lin, J. Kwok, and Y. Zhang, "Routerdc: Query-based router by dual contrastive learning for assembling large language models," in *NeurIPS*, 2024.

[14] J. Zhang, Z. Huang, Y. Fan, N. Liu, M. Li, Z. Yang, J. Yao, J. Wang, and K. Wang, "Kabb: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems," in *ICML*, 2025.

[15] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su, "Mind2web: Towards a generalist agent for the web," in *NeurIPS*, 2023.

[16] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press, "Swe-agent: Agent-computer interfaces enable automated software engineering," in *NeurIPS*, 2024.

[17] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin *et al.*, "Metagpt: Meta programming for a multi-agent collaborative framework," in *ICLR*, 2023.

[18] J. Wang, H. Xu, H. Jia, X. Zhang, M. Yan, W. Shen, J. Zhang, F. Huang, and J. Sang, "Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration," in *NeurIPS*, 2024.

[19] Y. Liu, H. Sun, W. Liu, J. Luan, B. Du, and R. Yan, "Mobilesteward: Integrating multiple app-oriented agents with self-evolution to automate cross-app instructions," in *KDD*, 2025.

[20] T. Fan, J. Wang, X. Ren, and C. Huang, "Minirag: Towards extremely simple retrieval-augmented generation," *arXiv preprint arXiv:2501.06713*, 2025.

[21] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in *NeurIPS*, 2019.

[22] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "{TVM}: An automated {End-to-End} optimizing compiler for deep learning," in *OSDI 18*, 2018, pp. 578–594.

[23] Q. Liu, Z. Hong, P. Li, F. Chen, and S. Guo, "Mell: Memory-efficient large language model serving via multi-gpu kv cache management," in *INFOCOM*, 2025.

[24] E. Frantar, S. Ashkboos, T. Hoefler, and D.-A. Alistarh, "Optq: Accurate post-training quantization for generative pre-trained transformers," in *ICLR*, 2023.

[25] Y. Gu, L. Dong, F. Wei, and M. Huang, "Minillm: Knowledge distillation of large language models," in *ICLR*, 2024.

[26] K. Lu, H. Yuan, R. Lin, J. Lin, Z. Yuan, C. Zhou, and J. Zhou, "Routing to the expert: Efficient reward-guided ensemble of large language models," in *NAACL*, 2024.

[27] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica, "Routellm: Learning to route llms from preference data," in *ICLR*, 2025.

[28] D. Jiang, X. Ren, and B. Y. Lin, "Llm-blender: Ensembling large language models with pairwise ranking and generative fusion," in *ACL*, 2023.

[29] H. Wang, F. M. Polo, Y. Sun, S. Kundu, E. Xing, and M. Yurochkin, "Fusing models with complementary expertise," in *ICLR*, 2024.

[30] P. Yadav, D. Tam, L. Choshen, C. A. Raffel, and M. Bansal, "Ties-merging: Resolving interference when merging models," in *NeurIPS*, 2023.

[31] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen2.5: A party of foundation models!" 2025. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/

[32] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[33] W. U. Ahmad, S. Narenthiran, S. Majumdar, A. Ficek, S. Jain, J. Huang, V. Noroozi, and B. Ginsburg, "Opencodereasoning: Advancing data distillation for competitive coding," *arXiv preprint arXiv:2504.01943*, 2025.

[34] J. Chen, Z. Cai, K. Ji, X. Wang, W. Liu, R. Wang, J. Hou, and B. Wang, "Huatuogpt-o1, towards medical complex reasoning with llms," *arXiv preprint arXiv:2412.18925*, 2024.

[35] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," 2024. [Online]. Available: https://ai.meta.com/research/publications/the-llama-3-herd-of-models/

[36] OpenAI, "Sharegpt," 2024. [Online]. Available: https://huggingface.co/datasets/RyokoAI/ShareGPT52K

[37] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017.

[38] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *CVPR*, 2018.

[39] S. W. Yoon, J. Seo, and J. Moon, "Tapnet: Neural network augmented with task-adaptive projection for few-shot learning," in *ICML*, 2019.

[40] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *FOCS*, 2003.

[41] T. Roughgarden and É. Tardos, "How bad is selfish routing?" *Journal of the ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[42] R. Durrett, *Probability: Theory and Examples*, 5th ed., 2019.

[43] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP*, 2019.

[44] Scikit-learn, "Agglomerativeclustering," 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

[45] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *ICLR*, 2021.

[46] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[47] D. Jin, E. Pan, N. Oufattole, W.-H. Weng, H. Fang, and P. Szolovits, "What disease does this patient have? a large-scale open domain question answering dataset from medical exams," *Applied Sciences*, vol. 11, no. 14, p. 6421, 2021.

[48] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.

[49] W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen, and N. Duan, "Agieval: A human-centric benchmark for evaluating foundation models," in *NAACL*, 2024.

[50] J. Li, Q. Zhang, Y. Yu, Q. Fu, and D. Ye, "More agents is all you need," in *TMLR*, 2024.