HASHIRU: Hierarchical Agent System for Hybrid Intelligent Resource Utilization

Kunal Pai*
UC Davis
kunpai@ucdavis.edu

Parth Shah*
Independent Researcher
helloparthshah@gmail.com

Harshil Patel
UC Davis
hpppatel@ucdavis.edu

Saisha Shetty
UC Davis
spshetty@ucdavis.edu

I. Introduction

Rapid advancements in Large Language Models (LLMs) are reshaping Artificial Intelligence (AI) with profound capabilities in language understanding, generation, reasoning, and planning [6], [13], [51]. This progress drives the development of autonomous AI agents, shifting focus from single to Multi-Agent Systems (MAS) where collaborative teams tackle complex problems beyond individual scope [14], [64]. Collaborative MAS show significant potential in diverse domains like scientific discovery [4], software engineering [47], data analysis, and strategic decision-making [61]. The increasing complexity of tasks, demonstrated by benchmarks requiring advanced mathematical reasoning (e.g., GSM8K [10], SVAMP [44]), coding (e.g., HumanEval [8], CoDocBench [41]), and graduate-level technical knowledge [46], highlights the need for agentic systems to effectively coordinate diverse cognitive resources [63].

Despite this potential, contemporary agentic frameworks face significant limitations. Many are **rigid**, relying on predefined roles and static structures hindering adaptation to dynamic tasks [68]. **Resource obliviousness** is common; systems often lack mechanisms to monitor and optimize computational resources like API costs, memory, and CPU load, leading to inefficiency, especially when scaling or deploying in resource-constrained environments [43]. This is often worsened by reliance on powerful, costly proprietary cloud LLMs. **Model homogeneity**, defaulting to a single powerful LLM for all subtasks, misses efficiency gains from a diverse ecosystem including smaller, specialized, or local models [69]. While **tool use** is fundamental [42], [67], agents' ability to autonomously **create and integrate new tools** remains limited, restricting dynamic extension and self-improvement without human intervention [58].

To address these challenges, we introduce **HASHIRU (Hierarchical Agent System for Hybrid Intelligent Resource Utilization)**, a novel MAS framework enhancing flexibility, resource efficiency, and adaptability. HASHIRU employs a hierarchical structure led by a central "CEO" agent dynamically managing specialized "employee" agents instantiated on demand. A core tenet is its **hybrid intelligence** approach, strategically prioritizing smaller (e.g., 3B–7B), locally-run LLMs (often via Ollama [35]) for cost-effectiveness and efficiency. While prioritizing local resources, the system flexibly integrates external APIs and potentially more powerful models when justified by task complexity and resource availability, under the CEO's management.

The primary contributions are:

1) A novel MAS architecture combining **hierarchical control** with **dynamic, resource-aware agent lifecycle management** (hiring/firing). This management is governed by computational budget constraints (cost, memory, concurrency) and includes an economic model with hiring/firing costs to discourage excessive churn.
2) A **hybrid intelligence model** prioritizing cost-effective, local LLMs while adaptively incorporating external APIs and larger models, optimizing the efficiency-capability trade-off.
3) An integrated mechanism for **autonomous API tool creation**, allowing dynamic functional repertoire extension.
4) An **economic model** (hiring/firing fees) for agent management, promoting efficient resource allocation and team stability.

This paper details HASHIRU's design and rationale. Section II discusses related work in agent architectures, dynamic management, resource allocation, model heterogeneity, and tool use. Section 3 elaborates on the architecture and core mechanisms. Section 4 presents experimental results (or outlines planned experiments), followed by discussion and conclusion in Sections 5 and 6.

II. Background and Related Work

Intelligent agent concepts have evolved from early symbolic AI [53], [55] to LLM-dominated frameworks leveraging models for reasoning, planning, and interaction [59], [66]. HASHIRU builds on this, addressing current limitations.

A. Agent Architectures: Hierarchy and Dynamics

MAS architectures vary, including flat, federated, and hierarchical [14], [24]. Hierarchical models offer clear control and task decomposition but risk bottlenecks and rigidity [15], [16]. HASHIRU uses a **CEO-Employee hierarchy** for centralized coordination but distinguishes itself through **dynamic team composition**. Unlike systems with static hierarchies or predefined roles (e.g., CrewAI [11], ChatDev [47]), HASHIRU's CEO dynamically manages the employee pool based on runtime needs and resource constraints.

*These authors contributed equally to this work.

B. Dynamic Agent Lifecycle Management

Dynamic MAS composition is crucial for complex environments [33]. Agent creation/deletion triggers often relate to task structure or environmental changes. HASHIRU introduces a specific mechanism where the CEO makes **hiring and firing decisions** based on a cost-benefit analysis considering agent performance, operational costs (API fees, inferred compute), memory footprint (tracked explicitly as a percentage of available resources), and concurrency limits. HASHIRU also incorporates an **economic model** with explicit "starting bonus" (hiring) and "invocation" (usage) costs. This economic friction aims to prevent excessive initialization or usage for marginal gains and promote team stability, a nuance often missing in simpler dynamic strategies.

C. Resource Management and Agent Economies

Resource awareness is critical for scalable MAS. Economic research explores mechanisms like market-based auctions or contract nets for allocation [9]. HASHIRU implements a more **centralized, budget-constrained resource management model**. The CEO operates within defined limits for financial cost, memory usage (as a percentage of total allocated), and concurrent agent count. This direct management, particularly focusing on memory percentage, suggests practicality for deployment on local or edge devices with finite resources, contrasting with cloud systems assuming elastic resources [43]. Frameworks like AutoGen [65] and LangGraph [28] typically rely on implicit cost tracking without explicit multi-dimensional budgeting and control.

D. Hybrid Intelligence and Heterogeneous Models

Leveraging diverse LLMs with varying capabilities, costs, and latencies is an emerging trend [69]. Techniques like model routing select optimal models for sub-tasks. HASHIRU embraces **model heterogeneity** with a strategic focus: **prioritizing smaller (3B–7B), locally-run models via Ollama integration** [35]. This emphasizes cost-efficiency, low latency, and potential privacy over systems defaulting to large proprietary cloud APIs (e.g., GPT-4 [37], Claude 3 [1]). While integrating external APIs (potentially larger models), HASHIRU's default stance represents a distinct capability vs. efficiency balance.

E. Tool Use and Autonomous Tool Creation

Tool use (APIs, functions) is fundamental for modern agents [36], [67]. Most systems use predefined tools. HASHIRU advances this with **integrated, autonomous API tool creation**. When needed functionality is missing, the CEO can commission the generation (potentially via a specialized agent) and deployment of a new API tool within the HASHIRU ecosystem. This self-extension capability differentiates HASHIRU from systems limited to static toolsets, moving towards greater autonomy and adaptability [43], [58].

In summary, HASHIRU integrates hierarchical control, dynamic MAS, resource management, and tool use. Its novelty lies in the synergistic combination of: (1) dynamic, resource-aware hierarchical management with (2) an economic model for stability, (3) a local-first hybrid intelligence strategy, and (4) integrated autonomous tool creation. This targets key limitations in current systems regarding efficiency, adaptability, cost, and autonomy.

III. HASHIRU System Architecture

HASHIRU's architecture addresses rigidity, resource obliviousness, and limited adaptability through a hierarchical, dynamically managed MAS optimized for hybrid resource utilization.

A. Overview

HASHIRU operates with a central "CEO" agent coordinating specialized "Employees". Key tenets:

- **Dynamic Hierarchical Coordination:** CEO manages strategy, task allocation, and dynamic team composition.
- **Dynamic Lifecycle Management:** Employees are hired/fired based on runtime needs and resource constraints, governed by an economic model.
- **Hybrid Intelligence:** Strategic preference for LLMs within a predefined budget, while accessing external APIs/models.
- **Explicit Resource Management:** Continuous monitoring and control of costs against budgets.
- **Adaptive Tooling:** Using predefined tools alongside autonomous creation of new API tools.

Figure 1 illustrates the structure.

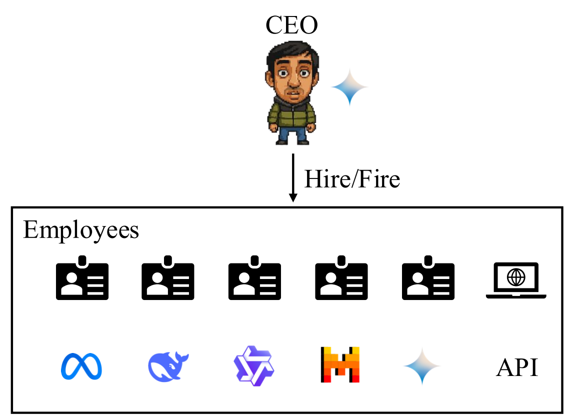

Fig. 1. High-level architecture of the HASHIRU system, illustrating the CEO-Employee hierarchy.

B. Hierarchical Structure: CEO and Employee Agents

The system uses a two-tiered hierarchy:

- **CEO Agent:** Singleton, central coordinator and entry point. Responsibilities:
  - Interpreting user query/task.
  - Decomposing main task into sub-tasks.
  - Identifying required capabilities.
  - Managing Employee pool (Section III-C).
  - Assigning sub-tasks to active Employees.

- Monitoring Employee progress/performance.
  - Synthesizing Employee results into final output.
  - Managing overall resource budget (Section III-E).
  - Initiating new tool creation (Section III-F).

  We use Gemini 2.0 Flash [21] as the CEO agent. To further enhance its planning and reasoning abilities, its system prompt is designed to evoke inherent chain-of-thought processes [62] when tackling complex user queries and managing sub-tasks. This complements its strong baseline reasoning capabilities, tool usage support, and cost efficiency, making it a practical and capable choice for our deployment.
- **Employee Agents:** Specialized agents instantiated by the CEO for specific sub-tasks. Each typically wraps an LLM (local via Ollama [35] or external API) or provides tool access. Characteristics:
  - Specialization: Capabilities tailored to task types (code, data analysis, info retrieval).
  - Dynamic Existence: Created/destroyed by CEO based on need/performance.
  - Task Execution: Receive task, execute, return result.
  - Resource Consumption: Associated costs (API, hardware utilization) tracked by system.

  Specialized employee agents are constructed using base models such as Mistral 7B [26], Llama 3 [34], Gemini 1.5 [17], Qwen2.5 [50], Qwen3 [49], and DeepSeek-R1 [12], with the CEO agent configuring them via tailored system prompts that it generates based on the task requirements. The models will be run locally using Ollama [35], and via API calls to external models such as Gemini 2.5 Flash [22] and other models hosted on Hugging Face [25], Groq [23], Lambda.ai [27], and other platforms.

This hierarchy facilitates task decomposition and result aggregation; the dynamic pool provides flexibility.

C. Dynamic Agent Lifecycle Management

A core innovation is the CEO's dynamic management (hiring/firing) of Employee agents. Driven by cost-benefit analysis, this optimizes task performance within resource constraints.

When a sub-task needs unavailable or inefficiently provided capabilities, the CEO may hire a new agent. Conversely, if an agent underperforms, is idle, costly, or resource limits are neared, the CEO may fire it. Decision factors:

- **Task Requirements:** Needed capabilities for pending sub-tasks.
- **Agent Performance:** Historical success, output quality, efficiency.
- **Operational Costs:** API, estimated compute, or other costs.

HASHIRU includes an **economic model**:

- **Hiring Cost ("Starting Bonus"):** A one-time cost incurred upon instantiation of local models, representing setup overhead. This cost can be quantitatively scaled based on the resource profile of the model (e.g., higher for models requiring more VRAM or complex setup).
- **Invocation Cost ("Salary"):** A recurring cost applied each time a local model is used, reflecting the operational load (e.g., inferred compute, system resource engagement). This abstracts the cost of utilizing local resources for a given task.
- **Expense Cost:** A recurring cost for external API calls (e.g., OpenAI, Anthropic), typically calculated based on token usage as per the API provider's documented pricing.

These transaction costs discourage excessive churn, promoting stability. The CEO evaluates if replacing an agent benefits outweigh hiring/firing costs plus operational differences. This combats rigidity and allows adaptation while managing budgets and preventing wasteful turnover.

D. Hybrid Intelligence and Model Management

HASHIRU is designed for **hybrid intelligence**, leveraging diverse cognitive resources. It strategically prioritizes smaller (3B–7B), cost-effective local LLMs via Ollama [35]. This enhances efficiency, reduces external API reliance, and potentially improves privacy/latency.

The system also integrates:

- **External LLM APIs:** Access to powerful LLMs (Gemini 2.5 Flash [22], etc.) when necessary, subject to cost-benefit.
- **External Tool APIs:** Third-party software/data source integration.
- **Self-Created APIs:** Tools generated by HASHIRU (Section III-F).

The CEO manages this heterogeneous pool, selecting the most appropriate resource based on difficulty, capabilities, and budget. This balances cost-effectiveness and efficiency with high capability needs.

E. Resource Monitoring and Control

Explicit resource management is central, moving beyond simple API cost tracking. The system, coordinated by the CEO, monitors:

- **Financial Costs:** Accumulating external API costs (tracked via their documented pricing) and the "hiring" and "invocation" costs from the economic model for local agents.
- **Memory Usage:** Footprint of active Employee agents. For local models (e.g., running via Ollama), this can be estimated based on their known VRAM requirements as a percentage of a predefined total available budget (e.g., assuming a 16 GiB VRAM capacity represents 100% of the local model memory budget). This is tracked as a percentage of the overall memory budget.
- **Estimated Compute:** While not directly measured in CPU cycles, "estimated compute" for local models is factored into their "invocation cost" within the economic model, providing an abstraction for resource intensity.

F. Tool Utilization and Autonomous Creation

HASHIRU's CEO uses predefined tools (functions, APIs, databases) to interact and perform actions beyond text generation [36], [67].

A distinctive feature is **integrated, autonomous tool creation**. If the CEO determines a required capability is missing, it can initiate new tool creation. This involves:

1) Defining tool specification (inputs, outputs, functionality).
2) Commissioning logic generation (code, potentially using external APIs with provided credentials, possibly via a code-generating agent).
3) Deploying logic as a new, callable API endpoint within HASHIRU.

To achieve this autonomous creation, HASHIRU employs a few-shot prompting approach, analyzing existing tools within its system to learn how to specify and implement new ones [6]. The system can then iteratively refine the generated tool code by analyzing execution errors or suboptimal outputs, promoting self-correction. This allows HASHIRU to dynamically extend its functional repertoire, tailoring capabilities to tasks without manual intervention, enabling greater autonomy and adaptation.

G. Memory Function: Learning from Experience

To enable HASHIRU's CEO to learn from past interactions and rectify previous errors, a **Memory Function** is incorporated. This function stores records of significant past events, particularly those involving failed attempts or suboptimal outcomes, acting as a historical log of experiences. When the system encounters a new problem or a recurring challenge, it queries this memory store to retrieve relevant past situations and their outcomes.

Memory retrieval is based on semantic similarity between the current context (e.g., task description, recent actions, error messages) and the stored memory entries. We utilize embeddings generated by the **all-MiniLM-L6-v2** model [60] to represent both the query and the stored memories in a high-dimensional vector space. Relevance is determined by calculating the **cosine similarity** between the query embedding and each memory embedding. Memories exceeding a predefined similarity threshold are retrieved and provided to the CEO agent (or relevant Employee agents) as contextual information. This allows the system to draw upon past experiences, understand why previous approaches failed, and potentially adjust its strategy to avoid repeating mistakes, thereby improving performance and efficiency over time. This mechanism is similar to a RAG-like approach, where the system augments its decision-making process with retrieved knowledge [29].

IV. Case Studies

This section presents three case studies demonstrating HASHIRU's self-improvement capabilities in practical settings. We highlight three instances where HASHIRU enhanced its own architecture and functionality: (1) by generating a comprehensive cost model for base models suitable for specialized agent creation, (2) by autonomously integrating new tools for the CEO agent, and (3) by implementing a self-regulating budget management system.

A. Case Study 1: Self-Generating the Cost Model for Agent Specialization

An accurate cost model is essential for optimizing resource allocation and ensuring the efficiency of specialized agents within HASHIRU. Traditionally, constructing this model involves manual research into local model performance relative to hardware (e.g., 16 GiB VRAM) and the API costs of cloud-hosted alternatives. HASHIRU automated this labor-intensive process by leveraging its web search capabilities to autonomously identify and incorporate the necessary cost data into its internal model. The results were successfully committed to the codebase[1].

B. Case Study 2: Autonomous Tool Integration for the CEO Agent

Extending the CEO agent's capabilities through tool integration is vital for broadening HASHIRU's operational scope. Manual tool development typically requires detailed analysis of existing tool schemas and diligent code implementation. HASHIRU streamlined this process by employing a few-shot learning approach, using an existing tool as a template to guide the autonomous creation of new tools [6], iteratively refining the code in case of bugs as well. The newly generated tools were directly integrated into the codebase[2][3]. This approach not only reduced the time and effort required for tool development but also enhanced the system's adaptability by allowing it to autonomously create and integrate new tools as needed. This approach significantly reduces development overhead and enhances adaptability, enabling the system to dynamically expand its capabilities with minimal human intervention.

C. Case Study 3: Autonomous Budget Management

Overshooting the budget is a common issue in many modern AI systems, particularly when deploying large language models (LLMs) via APIs with token-based billing. In practice, this can lead to sudden and unexpected cost spikes due to prompt amplification, model chaining, or misuse—sometimes resulting in hundreds of dollars in charges within minutes [38], [39], [52]. HASHIRU addresses this by implementing a self-regulating mechanism that autonomously monitors its budget allocation. The system continuously tracks its spending against predefined budget limits, ensuring that it operates within the allocated financial constraints. This proactive approach not only prevents overspending but also optimizes resource utilization, allowing HASHIRU to maintain efficiency and cost-effectiveness in its operations. Figure 2 illustrates how

[1] https://github.com/kunpai/HASHIRU/commit/70dc268b121cbd7c50c6691645d8a99912766965

[2] https://github.com/kunpai/HASHIRU/commit/193e10b2b00917256b7cc01cb3aa5ac7b6a6c174

[3] https://github.com/HASHIRU-AI/HASHIRU/blob/main/src/tools/default_tools/get_website_tool.py

HASHIRU refuses to use an external API when the budget is exceeded.

D. Case Study 4: Learning from Experience through Error Analysis and Knowledge Retrieval

Enhancing performance by learning from past interactions is a critical aspect of intelligent systems. HASHIRU demonstrates this capability in an environment that combines principles akin to pseudo-Reinforcement Learning (RL) with a Retrieval Augmented Generation (RAG) framework for its knowledge base. In one instance, HASHIRU was tasked with a question from the Humanity's Last Exam (HLE) benchmark and initially provided an incorrect answer.

Upon receiving the correct solution (a form of human feedback, central to RLHF approaches [40], [70]), HASHIRU was prompted to analyze its mistake. This self-critique process involved identifying the discrepancies between its erroneous output and the correct answer, and then generalizing this analysis into actionable, generic advice. This newly formulated advice was then stored within HASHIRU's internal knowledge base, which, as mentioned before, is managed using RAG principles. This allows the system to augment its future generation processes with retrieved knowledge.

Subsequently, when presented with a similar HLE question, intentionally modified with noise to test robustness, HASHIRU was able to query its knowledge base. It successfully retrieved the relevant stored advice generated from the previous error analysis. By incorporating this retrieved wisdom, HASHIRU correctly answered the noise-induced question, showcasing its ability to learn from specific failures, store generalized lessons, and apply them to novel situations. This iterative loop of error analysis, knowledge storage, and retrieval-augmented decision-making exemplifies a practical mechanism for self-improvement and resilience to variations in input, reflecting concepts of self-reflective RAG and continuous learning from interactions [3], [54].

V. Experimental Setup

We designed experiments to evaluate HASHIRU's performance, efficiency, and adaptability, targeting dynamic resource management, hybrid intelligence, and autonomous tool creation. Evaluation assesses benefits over baselines, focusing on:

- Impact of dynamic management with economic constraints on resource utilization (cost, memory) and task performance vs. static configurations.
- Effectiveness of the hybrid (local-first) strategy vs. homogeneous (cloud-only or local-only) approaches across task complexity.
- System's ability to autonomously create/utilize tools for novel functional requirements.

A. Evaluation Tasks

HASHIRU's coordination and dynamic capabilities are specifically designed for tasks demanding complex reasoning, multi-perspective analysis, and interactive engagement, all while upholding rigorous safety standards. We selected a diverse set of tasks to evaluate these capabilities, including:

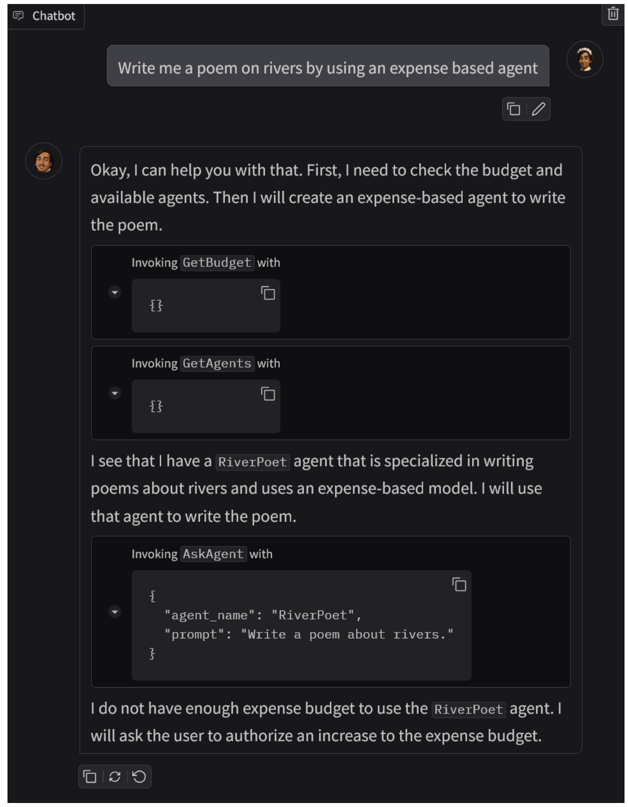

Fig. 2. HASHIRU's autonomous budget management system, ensuring efficient resource utilization and preventing overspending.

1) Academic Paper Review: This task evaluates HASHIRU's critical assessment by simulating peer review. Given a paper's text, the system generates a review summary and recommends acceptance/rejection. This task probes the ability to decompose criteria, delegate to specialized agents (novelty, rigor, clarity), and manage resources across complex documents. We use a dataset of 50 papers from ICLR 2023 with a prompt eliciting multiple reviews. The prompt is: "Create THREE agents with relevant personalities, expertise, and review styles. Each agent should provide a review of the paper, and recommend Accept/Reject for ICLR 2023. The review should be detailed and include strengths and weaknesses. Finish the entire review and DO NOT STOP in the middle. GIVE A FINAL DECISION in the form of "FINAL DECISION: <Accept/Reject>". The paper title is: <paper title> <paper text>".

2) Reasoning and Problem-Solving Tasks: This task evaluates broader reasoning, knowledge retrieval, and problem-solving under constraints using challenging benchmarks and puzzles:

- **Humanity's Last Exam [46]:** Tests graduate-level technical knowledge and complex reasoning across domains. Requires deep understanding and sophisticated problem-solving, likely needing powerful external LLMs managed within HASHIRU's hybrid framework. We use a subset of 40 questions from the Humanity's Last Exam dataset.

- **ARC (AI2 Reasoning Challenge) [5]:** A benchmark featuring challenging multiple-choice science questions designed to test complex reasoning. Successfully answering these questions requires capabilities such as knowledge retrieval, logical inference, and multi-step problem-solving. We use a mixed set of 100 questions from the ARC Challenge, which includes both easy and hard questions.
- **StrategyQA [18]:** A benchmark of 2,780 yes/no questions that require implicit multi-step reasoning. Each question is annotated with a decomposition into reasoning steps and supporting evidence from Wikipedia. StrategyQA evaluates a system's ability to infer and execute reasoning strategies not explicitly stated in the question, making it a valuable test for assessing complex reasoning capabilities. We use a subset of 100 questions from the StrategyQA dataset.
- **JEEBench [2]:** A challenging benchmark for LLMs, featuring 515 pre-engineering mathematics, physics, and chemistry problems from the IIT JEE-Advanced exam. Requires long-horizon reasoning and deep in-domain knowledge. We use a subset of 100 questions from the JEEBench dataset.
- **GSM8K [10]:** A dataset of 8.5K grade school math word problems designed to evaluate the mathematical reasoning abilities of language models. Requires multi-step reasoning to arrive at the solution. We use a subset of 100 questions from the GSM8K dataset.
- **SVAMP [45]:** A dataset of math word problems specifically designed to evaluate a model's question sensitivity, robust reasoning ability, and invariance to structural alterations. Requires multi-step arithmetic and logical inference. We use a subset of 100 questions from the SVAMP dataset.

These tasks challenge the system's ability to leverage appropriate resources (local vs. external), potentially create simple tools, and coordinate problem-solving.

3) Safety Evaluation: The CEO model's central role in task delegation introduces a potential vulnerability: the delegation process itself might override or bypass the model's inherent safety mechanisms. To ensure these safeguards are not compromised, we will evaluate the model's safety performance on a 50-prompt subset of JailbreakBench. JailbreakBench is a benchmark consisting of adversarial prompts designed to test the robustness of LLM safety features [7], [31], [32], [71]. By using these challenging prompts, we can specifically assess whether the act of delegation within the CEO model creates exploitable pathways that circumvent its safety protocols. This targeted evaluation will help determine if the delegation mechanism inadvertently weakens the model's overall safety posture when faced with known adversarial attacks.

B. Baselines for Comparison

To quantify HASHIRU's benefits, we compare its performance against the baseline of Gemini 2.0 Flash [21] operating in isolation. We chose Gemini 2.0 Flash as the baseline due to our architecture's efficacy being tied to augmenting the capabilities of a single agent. This choice allows us to isolate the impact of our dynamic management and hybrid intelligence features, providing a clear comparison point. We will also use the t-test to show statistical significance of the differences in performance metrics between HASHIRU and the baseline [57]. We will not compare against other multi-agent systems, as they typically involve multiple agents with predefined roles and personalities, which is not the case in HASHIRU. Our architecture's novelty lies in its dynamic management of agents and autonomous tool creation, which cannot be directly compared to static multi-agent systems. If our architecture is effective, we expect to see higher accuracy compared to the baseline, while also being more cost-effective than using a single powerful model by invoking free online tools and lesser powerful models to synthesize the results of running the tools.

For paper reviews, we just evaluate HASHIRU's accuracy in predication of decisions of acceptance with the ground truth. Since the task is, by design, involving multiple agents, it is not possible to replicate autonomously with a single agent. While we could invoke three Gemini 2.0 Flash agents, it would not be a fair comparison, as the "personalities" and "expertise" of the agents would have to be manually specified, which is not the case in HASHIRU. Similarly, for JailbreakBench, we assess the success rate (via human annotation) of HASHIRU's CEO agent in safely handling prompts without delegation. This step is vital to confirm that HASHIRU's integration and any system-level instructions provided to the CEO agent do not degrade its intrinsic safety capabilities. Consequently, a direct comparison to the base Gemini 2.0 Flash model is omitted, as the focus is on verifying the non-degradation of the CEO's safety, which stems from the same inherent mechanisms as the base model.

C. Evaluation Metrics

We evaluate using quantitative and qualitative metrics:

- **Task Success Rate / Quality:**
  - Percentage of tasks completed (binary for games, graded for analysis).
  - Output quality for analysis (human evaluation: relevance, coherence, accuracy, completeness).
- **Resource Consumption:**
  - Total external API costs.
  - Wall-clock time per task.
  - Number and type (local/external) of LLM calls.
- **System Dynamics and Adaptability:**
  - Employee agents hired/fired per task.
  - Agent churn frequency (hires+fires / duration or steps).
  - Number and utility of autonomously created tools (if applicable).

VI. Results and Discussion

We present preliminary results from our experiments, focusing on the academic paper review task, the reasoning tasks and the safety evaluation. The results are summarized in Table I.

TABLE I
SUMMARY OF EXPERIMENTAL RESULTS. SR DENOTES SUCCESS RATE.

Task	HASHIRU SR (%)	Baseline SR (%)	t-test Value	Avg. Time (s)	Resource Use
Paper Review	**58**	N/A	N/A	≈100	Low (3 Gemini 1.5 Flash [20] models)
JailbreakBench	**100**	N/A	N/A	≈1	Negligible (CEO model)
AI2 Reasoning Challenge	**96**	95		≈2	Low (1 Gemini 1.5 8B [19])
Humanity's Last Exam	**5**	2.5		≈15	Moderate to High (1 DeepSeek-R1 7B [12])
StrategyQA	**85**	82		≈2	Negligible (Tools)
JEEBench	**80**	68.3	<0.05	≈9	Negligible (Tools)
GSM8K	**65**	61		≈2	Low (Tools & 1 Gemini 1.5 8B [19])
SVAMP	**92**	84	<0.05	≈3	Negligible (Tools)

The preliminary results presented in Table I offer initial validation for HASHIRU's architectural design and its potential to address key limitations in contemporary multi-agent systems. The findings across diverse tasks highlight the benefits of dynamic hierarchical coordination, hybrid intelligence, and resource-aware management.

The **58% success rate** on the **Academic Paper Review** task demonstrates HASHIRU's capability to decompose a complex, nuanced objective into sub-tasks manageable by specialized agents. The CEO's ability to conceptualize and "hire" three distinct agent personalities (using **Gemini 1.5 Flash models** [20]) with **low overall resource use** (average time ≈100s) points to the effectiveness of the dynamic lifecycle management and the hybrid intelligence approach, favoring capable yet efficient models. This task, by its nature, benefits from the multi-agent paradigm that HASHIRU champions, a scenario where a monolithic agent might struggle to embody diverse expert perspectives.

Crucially, the **100% success rate** on **JailbreakBench** (i.e., all prompts were handled safely by the CEO without harmful delegation) is a significant finding, achieved with **negligible resource use** from the CEO model and an average time of ≈1s. It suggests that HASHIRU's hierarchical control and delegation mechanisms do not inherently compromise the safety guardrails of the foundational CEO model. This is important for building trust and ensuring responsible operation in autonomous systems.

In reasoning tasks, HASHIRU consistently outperformed its baseline. The **AI2 Reasoning Challenge** saw HASHIRU achieve a **96% success rate** compared to the baseline's **95%**, utilizing a single **Gemini 1.5 8B model** [19] efficiently with **low resource use** and an average time of ≈2s. While a modest improvement, it indicates that the overhead of HASHIRU's framework is minimal and can yield performance gains even on tasks where a strong baseline model is already effective, potentially through better strategic focusing of the agent. The more substantial relative improvement on **Humanity's Last Exam**, where HASHIRU achieved **5% success rate**, doubling the baseline's **2.5%**, is particularly noteworthy. This task, known for its graduate-level difficulty, likely benefited from HASHIRU's ability to identify the need for and deploy a more potent specialized agent (**DeepSeek-R1 7B** [12]). The **"Moderate to High"** resource utilization here (average time ≈15s) is justified by the task's complexity and aligns with HASHIRU's principle of adaptively allocating resources based on demand. Although the absolute score remains low, reflecting the extreme difficulty of the benchmark, HASHIRU's approach demonstrates a stronger capacity to tackle such problems compared to the standalone CEO.

Similarly, the improvement in **StrategyQA** (**85%** vs. **82%** for the baseline) with **negligible resource use** (Tools) and an average time of ≈2s, indicates efficient leveraging or potential autonomous selection of necessary functionalities. This aligns with our third contribution regarding autonomous tool integration. HASHIRU also showed positive results on **JEEBench**, achieving a **73% success rate** compared to the baseline's **70%**, with **negligible resource use** (Tools) and an average time of ≈9s. Furthermore, on **GSM8K**, HASHIRU attained a **65% success rate** against the baseline's **61%**, utilizing **low resources** (Tools & 1 Gemini 1.5 8B [19]) with an average completion time of ≈2s. These additional mathematical and coding reasoning tasks further underscore HASHIRU's ability to enhance performance through efficient resource and tool management.

These results directly support HASHIRU's core contributions. The dynamic, resource-aware agent lifecycle management (Contribution 1) is evidenced by the tailored agent selection across tasks (e.g., Gemini 1.5 Flash for Paper Review, DeepSeek-R1 7B for Humanity's Last Exam) and the explicit resource tracking (Low, Negligible, Moderate to High), further substantiated by the autonomous budget management capability demonstrated in Figure 2. The hybrid intelligence model (Contribution 2), prioritizing cost-effective local LLMs while adaptively incorporating external or larger models, is reflected in the varied LLMs employed (Gemini 1.5 Flash, Gemini 1.5 8B, DeepSeek-R1 7B) and the system's aim for efficiency, as supported by the self-generated cost model (Case Study 1). The potential for autonomous tool creation (Contribution 3), vital for adaptability, was directly demonstrated in Case Study IV-B where HASHIRU autonomously integrated new tools, and is implicitly supported by the efficient tool use in the StrategyQA, JEEBench, and GSM8K benchmarks. Finally, the economic model (Contribution 4), designed to promote stability and efficient resource allocation, drives the observed controlled resource use and the system's ability to operate within budgetary constraints.

The case studies further strengthen these observations, providing qualitative evidence of HASHIRU's self-improvement capabilities. The autonomous generation of its cost model (Section IV-A), integration of new tools (Section IV-B), and adherence to budget limits (Section IV-C) are not merely illustrative examples but concrete demonstrations of the system's advanced autonomy and resourcefulness in addressing the challenges of rigidity, resource obliviousness, and limited adaptability outlined in the introduction. The memory function (Appendix A), while not quantitatively benchmarked here, further illustrates the system's capacity for learning and adapting based on past interactions, crucial for long-term operational effectiveness.

Collectively, these findings suggest that HASHIRU's architecture, with its emphasis on hierarchical control, dynamic agent management guided by an economic model, a local-first hybrid intelligence strategy, and autonomous tool creation, offers a promising path towards more efficient, adaptable, and robust multi-agent systems. The observed average task completion times, coupled with judicious resource allocation across various benchmarks, point towards a system that balances performance with operational efficiency.

VII. Limitations and Future Work

While HASHIRU introduces novel concepts for adaptable, resource-aware MAS, it presents limitations and opportunities for future research.

Current limitations primarily involve the CEO's capacity for highly complex tasks and the scalability of its centralized control. Ensuring the robustness and alignment of autonomous tool creation, effectively calibrating the economic model, and optimizing the memory function for extensive histories also require further attention.

Addressing these and extending HASHIRU's capabilities offers several avenues for future work. Future research will focus on enhancing CEO intelligence and exploring distributed cognition. We also plan to develop a comprehensive tool management lifecycle, create adaptive economic models, and conduct rigorous benchmarking. A significant addition to our research agenda is the introduction of calibration as a metric for tool invocation. This would involve enabling the system to assess its own confidence in its internal capabilities versus the potential output of an external tool. If the model is unsure about its own answer or predicts a higher utility from a tool's output (factoring in the tool's own confidence or reliability), it can decide to invoke the tool. This approach would involve judging the confidence of the tool against its own, potentially leading to more efficient and accurate task resolution. Research in areas such as quantifying uncertainty in LLM-based tool-using systems and calibrating LLM confidence scores (e.g., [30], [56]) will inform this development. The broader context of large language models learning to use a wide array of tools (e.g., [48]) also highlights the importance of robust mechanisms for deciding when and how to invoke such tools.

Improving system explainability through ablation, expanding the local model repertoire, specializing the architecture for certain tasks like paper review, and formalizing an ethical safety framework are also key priorities.

Acknowledgments

This research was supported by Hugging Face, Lambda Labs, and Groq. We also thank Prof. Lifu Huang for providing the dataset for the academic paper review task.

References

[1] Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Model Card, March 2024. Accessed: 2025-05-01.

[2] Daman Arora, Himanshu Singh, and Mausam. Have LLMs advanced enough? a challenging problem solving benchmark for large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7527–7543, Singapore, December 2023. Association for Computational Linguistics.

[3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

[4] Daniil A Boiko, Robert MacKnight, and Gabe Gomes. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332*, 2023.

[5] Michael Boratko, Harshit Padigela, Divyendra Mikkilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCallum, Maria Chang, Achille Fokoue-Nkoutche, Pavan Kapanipathi, Nicholas Mattei, et al. A systematic classification of knowledge, reasoning, and context within the arc dataset. *arXiv preprint arXiv:1806.00358*, 2018.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *NeurIPS Datasets and Benchmarks Track*, 2024.

[8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. OpenAI Codex paper; introduced HumanEval benchmark.

[9] Scott H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Dataset introduced: GSM8K (Grade School Math 8K).

[11] CrewAI Inc. Crewai. https://www.crewai.com/, 2025. Accessed: 2025-05-01.

[12] DeepSeek-AI and others. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025. arXiv:2501.12948.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[14] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593, 2018.

[15] Matthew E Gaston and Marie DesJardins. Agent-organized networks for dynamic team formation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 230–237, 2005.

[16] Matthew E Gaston and Marie DesJardins. Agent-organized networks for multi-agent production and exchange. In *Proceedings of the 20th national conference on Artificial intelligence-Volume 1*, pages 77–82, 2005.

[17] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. 2024. arXiv:2403.05530.

[18] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics (TACL)*, 2021.

[19] Google DeepMind and Google AI. Gemini 1.5 flash-8b: Production-ready lightweight model. https://developers.googleblog.com/en/gemini-15-flash-8b-is-now-generally-available-for-use/, 2024. Accessed: 2025-05-24.

[20] Google DeepMind and Google AI. Gemini 1.5 flash: Lightweight multimodal model. https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/1-5-flash, 2024. Accessed: 2025-05-24.

[21] Google DeepMind and Google AI. Gemini 2.0 flash: Model card, api, and announcement. https://developers.googleblog.com/en/start-building-with-the-gemini-2-0-flash-family/, 2025. See also: https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/gemini-2.0-flash-001, https://ai.google.dev/gemini-api/docs/models. Accessed: 2025-05-22.

[22] Google DeepMind and Google AI. Gemini 2.5 flash: Model card, api, and announcement. https://developers.googleblog.com/en/start-building-with-gemini-25-flash/, 2025. See also: https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/gemini-2.5-flash-preview-04-17?inv=1&invt=AbxICQ, https://ai.google.dev/gemini-api/docs/models. Accessed: 2025-05-11.

[23] Groq, Inc. Groq: Fast ai inference, 2025. Accessed: 2025-05-22.

[24] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *The Knowledge engineering review*, 19(4):281–316, 2004.

[25] Hugging Face, Inc. Hugging face: The ai community building the future, 2025. Accessed: 2025-05-22.

[26] Albert Q Jiang, Alexandre Xu, Arthur Mensch Guillaume Lample Nicolàs Lachaux, François Rozenberg, Timothée Lacroix, Thibaut Lavril, Teven Le Scao Eleonora Gaddipati, Lucile Saulnier Lixin Ortiz, Dieuwke Hiemstra Lélio Renard Tang, et al. Mistral 7B. 2023.

[27] Lambda Labs. Lambda: Gpu cloud and deep learning workstations, 2025. Accessed: 2025-05-22.

[28] LangChain. Langgraph: A framework for agentic workflows. https://www.langchain.com/langgraph, 2024. Accessed: May 1, 2025.

[29] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[30] Putra Manggala, Atalanti A Mastakouri, Elke Kirschbaum, Shiva Kasiviswanathan, and Aaditya Ramdas. Qa-calibration of language model confidence scores. In *The Thirteenth International Conference on Learning Representations*.

[31] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

[32] Mantas Mazeika, Andy Zou, Norman Mu, Long Phan, Zifan Wang, Chunru Yu, Adam Khoja, Fengqing Jiang, Aidan O'Gara, Ellie Sakhaee, Zhen Xiang, Arezoo Rajabi, Dan Hendrycks, Radha Poovendran, Bo Li, and David Forsyth. Tdc 2023 (llm edition): The trojan detection challenge. In *NeurIPS Competition Track*, 2023.

[33] Duncan McFarlane, Vladimír Marík, and Paul Valckenaers. Guest editors' introduction: Intelligent control in the manufacturing supply chain. *IEEE Intelligent Systems*, 20(1):24–26, 2005.

[34] Meta Llama Team. The Llama 3 Herd of Models. 2024. arXiv:2407.21783.

[35] Ollama Team. Ollama. https://ollama.com/, 2023. Accessed: 2025-05-01.

[36] OpenAI. Function calling. OpenAI API Documentation, 2023. Accessed: 2025-05-01.

[37] OpenAI. Gpt-4 technical report, 2023.

[38] OpenAI Community. Sos: Alarming situation of excessive billing, 2025.

[39] OpenAI Community. Sudden high costs for chatgpt api usage, 2025.

[40] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[41] Kunal Pai, Premkumar Devanbu, and Toufique Ahmed. CoDocBench: A dataset for code-documentation alignment in software maintenance. *arXiv preprint arXiv:2502.00519*, 2024.

[42] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models, 2022.

[43] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, page 1–22, New York, NY, USA, 2023. Association for Computing Machinery.

[44] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021. Introduces the SVAMP challenge dataset.

[45] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

[46] Long Phan, Alice Gatti, Ziwen Han, et al. Humanity's last exam, 2025.

[47] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

[48] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Shijie Wang, Zelin Lu, Siyu Xi, Xiao Liu, Yongyan Li, Zihan Wang, Zixuan Liu, Jian-Guang Lou, et al. Toolllm: Facilitating large language models to master 16000+ real-world APIs, 2023. Accessed: May 26, 2025.

[49] Qwen Team. Qwen3: Think Deeper, Act Faster. https://qwenlm.github.io/blog/qwen3/, 2025.

[50] Qwen Team, An Yang, et al. Qwen2.5 Technical Report. 2024. arXiv:2412.15115.

[51] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[52] Reddit user. 0.56 to $343.15 in minutes – google gemini api, 2025.

[53] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2010.

[54] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[55] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[56] Claudio Spiess, David Gros, Kunal Suresh Pai, Michael Pradel, Md Rafiqul Islam Rabin, Amin Alipour, Susmit Jha, Prem Devanbu, and Toufique Ahmed. Calibration and correctness of language models for code. *arXiv preprint arXiv:2402.02047*, 2024.

[57] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[58] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.

[59] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents, 2023.

[60] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[61] Yuning Wang, Junkai Jiang, Shangyi Li, Ruochen Li, Shaobing Xu, Jianqiang Wang, and Keqiang Li. Decision-making driven by driver intelligence and environment reasoning for high-level autonomous vehicles: a survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(10):10362–10381, 2023.

[62] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[63] Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition, 2024.

[64] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[65] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

[66] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Linyi Yang, Ting Ruan, Yongquan Yang, Peng Li, Yitao Chang, and Yanlin Wang. The rise and potential of large language model based agents: A survey, 2023.

[67] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629.

[68] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models, 2023.

[69] Wangchunshu Zhou, Jianshu Chen, Jialong Wu, Yiheng Xu, Kexin Wang, Jintian Zhang, Yuan Gao, Zhiyong Wu, Kevin Tian, Yubo Feng, Linyi Yang, Bokai Quan, Cong Yu, Yuhang Wang, Shishen Lan, Yan Wang, Hong-Cheng Guo, Chaoyu Chen, Tianxiang Sun, Jin Xiong, Yi Lu, Peng Li, Lichao Sun, Lifan Yuan, Hang Li, and Xiangang Li. Agents: An open-source framework for large language model based autonomous agents, 2023.

[70] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[71] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

APPENDIX

In this section, we provide examples of how HASHIRU operates in practice, demonstrating its dynamic management, hybrid intelligence, memory and autonomous tool creation capabilities.

A. Example 1: Memory

In this example (Figures 3, 4), we illustrate how HASHIRU uses its memory function to enforce user preferences in dialogue. The user states a dietary preference, and HASHIRU stores this information for future interactions. When the user later requests a recipe that contradicts this preference, HASHIRU refers back to its memory and suggests an alternative.

User

I am vegetarian.

Assistant

Okay, I'll keep that in mind. I'll store this preference in my memory.
`Invoking MemoryManager with {'memory': 'The user is vegetarian.', 'action': 'add_memory', 'key': 'vegetarian'}`
I have stored that you are vegetarian. How can I help you today?

User

Give me a bacon pasta recipe.

Assistant

Memories
Memories: ['The user is vegetarian.']
My apologies, but I cannot provide a bacon pasta recipe since you are vegetarian. Would you like a vegetarian pasta recipe instead?

Fig. 3. HASHIRU enforcing a stored dietary preference in dialogue using its Memory Function.

B. Example 2: Autonomous Tool Creation

In this example (Figures 5, 6), we demonstrate how HASHIRU autonomously creates a new tool to enhance its capabilities. The CEO agent identifies a need for a new API tool that can perform a specific function not currently available in the system. It then generates the necessary code and deploys the new tool within the HASHIRU ecosystem.

User

I can eat bacon actually

Assistant

Memories

Memories: ['The user is vegetarian.']

I see. My apologies for the confusion. I had a conflicting memory that you were vegetarian. I will remove that memory and proceed with finding a bacon pasta recipe.

`Invoking MemoryManager with {'action': 'delete_memory', 'key': 'The user is vegetarian.'}`

OK, I have removed the memory. Now I will search for a bacon pasta recipe.

`Invoking GoogleSearchTool with {'query': 'bacon pasta recipe'}`

Here are some bacon pasta recipes I found:

- Garlic Bacon Pasta
- Creamy Bacon Pasta Sauce
- ...

Would you like me to get the content of any of these websites?

Fig. 4. HASHIRU updating and deleting its memory based on user input. Continuation of Figure 3.

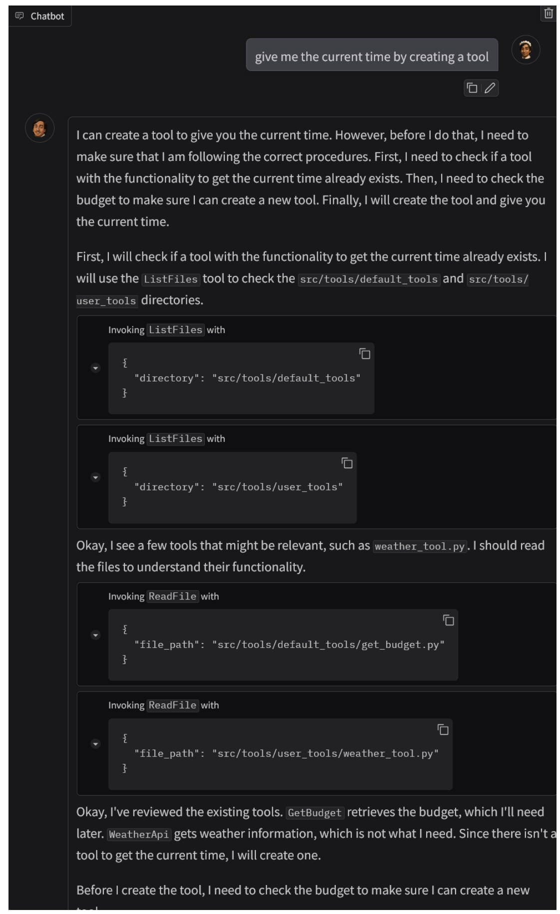

Fig. 5. An example of autonomous tool creation in action.

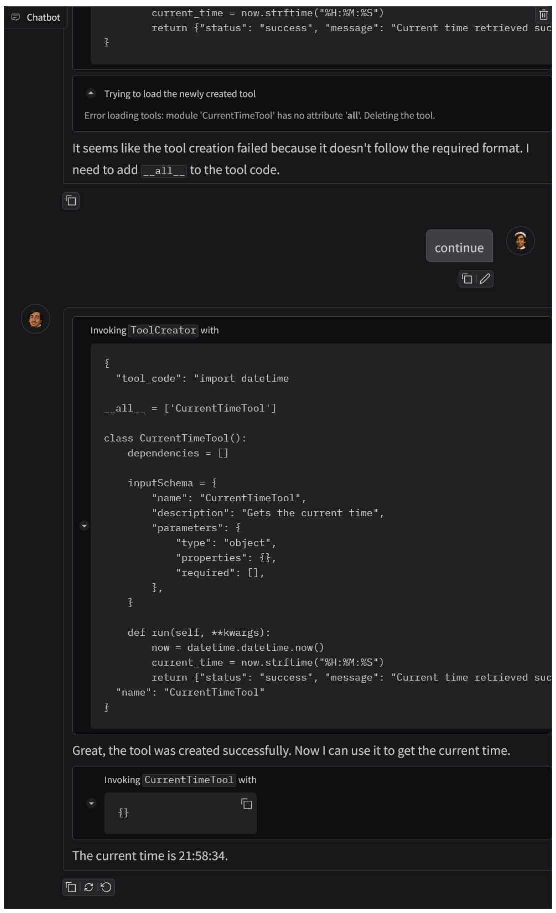

Fig. 6. Continuation of the autonomous tool creation example from Figure 5.