11-711 ANLP Lab 4 - Mixture of Mini Agents

Brian Curtin
bcurtin2

Moukhik Misra
moukhikm

Lakshay Arora
lakshaya

Abstract

Large language models (LLMs) are pivotal in data science due to their capabilities, yet advancing them demands high financial and computational resources. To improve efficiency, researchers are turning to methods like prompt engineering, fine-tuning, and agent frameworks that combine outputs from multiple models, often outperforming individual state-of-the-art LLMs (Wang et al., 2024). This study proposes an aggregator to enhance this framework, effectively merging knowledge from various LLMs to deliver better responses across prompts.

1 Introduction

The use of large language models is the focus of much data science research due to their capability and flexibility. Creating new model architectures and training models to become state-of-the-art requires an incredible amount of financial and material resources. Training these models is also becoming more difficult with the increasing size requirement of a single model to outperform previous large language models. Many researchers are now focusing on increasing the efficiency of existing large language models with prompt engineering, fine-tuning, and agent frameworks. Researchers can combine the outputs of many models in a mixture of agents framework to outperform state-of-the-art models (Wang et al., 2024). By creating multiple layers of agents, the authors can combine the knowledge of multiple large language models to give the best response to a variety of prompts. Our research focuses on improving this framework by optimizing the existing methods with a inter-layer aggregators to efficiently combine the knowledge of each model.

2 Literature Survey

Ensembling methods are common in data science to increase performance by leveraging the knowledge of several existing models. Frameworks like decision forests apply this concept to use many models proficient at different tasks. Specific to neural networks, mixture of experts is a framework that activates different parameters depending on the task given (Shazeer et al., 2017). This allows a model to increase its performance across multiple subject matters by allowing for some task-specific focus across multiple different "experts" of parameters. The outputs are then combined rather than ensembled to form an output creating a composite of knowledge.

While mixture-of-experts is a detailed implementation in a unique style of a large language model, the creation of the agent framework extends this idea of experts to a larger number of models. Users can utilize prompt engineering to increase performance with motivation and reasoning for a specific task (Yao et al., 2023). This allows users to tailor a large language model to their specific use case acting in a desired way. Specifically, they can create agents which are able to interact with their environment (Wooldridge, 1999). This can be functions that contain APIs or other agents they can discuss with. This same methodology applies to the agent framework. A user can create an agent for a specific task rather than expensive training or fine-tuning with similar or better results. The agent then acts in the prompted manner with the desired motivation and reasoning. Agents can even be prompted to perform specific, complex decision-making tasks (Yang et al., 2023).

Individual agents can complete complex and unique tasks without requiring a model to be retrained or fine-tuned (Yang et al., 2023). Like other machine learning models, agents can be ensembled to improve overall task performance by taking advantage of unique proficiencies and capabilities of unique designs or training (Wu et al., 2023). Large language models can even take the output of other models in the ensemble and debate and improve responses over several iterations to improve results (Du et al., 2023). These agents can act and role plays similar to how a human team interacts and problem-solves (Li et al., 2023a; Hong et al., 2024). While this work focuses on benchmarks and traditional large language model performance, researchers have also demonstrated a similar functionality by assigning agents different plugin functionalities to query outside information and include that in their responses (Talebirad and Nadiri, 2023). Giving specific, different tasks to agents allows them to provide different feedback optimized in a variety of ways.

This feedback mechanism differs in a variety of ways. A voting mechanism can be applied to the feedback of multiple agents through multiple iterations of debate (Chen et al., 2024b). This provides an additional mechanism by which the output of an ensemble can be regulated by the majority opinion of a group of various agents to prevent any unwanted behavior. Multiple agents can be generated in response to a prompt by a single large language model (Chen et al., 2024a). A team of observer agents is responsible for creating multiple unique agents to respond to a prompt and observing multiple rounds of communication. Through multiple iterations of discussion, the generated agents collaborate and debate on the prompt to determine the best-generated response. A variety of similar frameworks focus on removing any predefined agents to generate appropriate agents to respond to a prompt for any task (Liu et al., 2023). Agents can even be combined to form networks with multiple layers for a fixed forward pass of generated inference.

Users and other agents can generate unique agents with specific personality traits (Zhang et al., 2024). Agents can replicate human behavior by being given specific traits unique to human personality. These varying traits can increase the variance of an ensemble of agents increasing performance on multiple tasks and benchmarks. However, current models are limited in capability when presented with unique challenges related to human social interaction (Li et al., 2023b).

Despite the variety of ways agents can be created and combined, a simple combination and aggregation of multiple agent outputs has shown to be extremely effective in certain benchmarks (Wang et al., 2024). This work creates multiple layers of agents each using different large language models for inference. The output of each agent is then concatenated and passed onto a new layer of agents for additional generation as seen in figure 1. The final layer consists of an agent with an off-the-shelf model that aggregates the previous layer's responses into a single output. Our work will focus on improving this fixed architecture by training a unique aggregator to combine all responses between laters. These proposed methods are discussed in section 4.2. We hope to improve the performance of the mixture of agents by employing specifically trained large language model aggregators rather than larger generic models.

3 Baseline Reproduction

3.1 Methods

This methodology leverages multiple large language models (LLMs) collaboratively to boost overall performance through a process called Mixture-of-Agents (MoA). In this framework, models work together by taking on specific roles—some serve as "proposers," generating diverse initial responses, while others act as "aggregators," refining these responses into a single, higher-quality output.

Figure 1 shows the diagrammatic representation of the methodology. The MoA approach operates in layers, with each layer containing a set of LLMs. In each layer, these models process the input and produce responses, which are then synthesized using an "Aggregate-and-Synthesize" prompt. This layered structure allows for an iterative refinement of responses, where outputs from one layer become inputs for the next. The final output is derived from the last layer, using only one LLM for evaluation.

This methodology is inspired by the Mixture-of-Experts (MoE) model but is uniquely adapted to work at the model level rather than within individual networks. By using prompts rather than modifying model weights or activations, MoA is both flexible and scalable, allowing the use of any modern LLM without additional fine-tuning. This design achieves computational efficiency and is easily adaptable to various architectures and model sizes.

3.2 Results

3.2.1 Previous Results

AlpacaEval 2.0 is an automated evaluation framework that uses GPT-4-turbo to assess the performance of various language models in following instructions (Dubois et al., 2024). It ranks models by comparing their responses to a reference, aligning well with human judgments, and minimizing

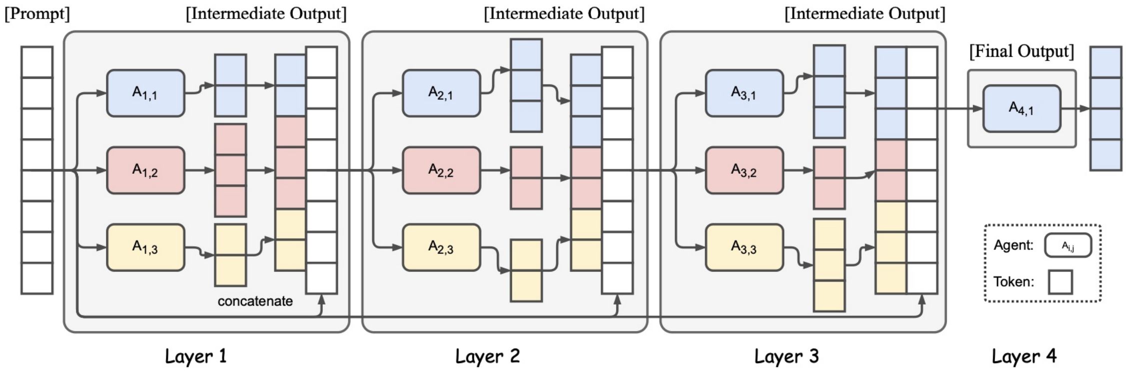

Figure 1: Mixture of Agents Architecture (Wang et al., 2024)

Model	LC win.	win.
Replicated MoA w/ Qwen 2.5	87.5%	96.8%
MoA w/ GPT-4o	65.7%	78.7%
Original MoA	65.1%	59.8%
Replicated MoA w/ Qwen 1.5	63.7%	56.7%
MoA-Lite	59.3%	57.0%
GPT-4 Omni (05/13)	57.5%	51.3%
GPT-4 Turbo (04/09)	55.0%	46.1%
WizardLM 8x22B	51.3%	62.3%
GPT-4 Preview (11/06)	50.0%	50.0%
Qwen1.5 110B Chat	43.9%	33.8%
Qwen1.5 72B Chat	36.6%	26.5%
GPT-4 (03/14)	35.3%	22.1%
Llama3 70B Instruct	34.4%	33.2%
Mixtral 8x22B v0.1	30.9%	22.2%

Table 1: AlpacaEval2.0

Model	Avg.	1st turn	2nd turn
MoA w/ GPT-4o	9.40	9.49	9.31
GPT-4 Turbo (04/09)	9.31	9.35	9.28
MoA	9.25	9.44	9.07
Replicated MoA w/ Qwen 2.5	9.22	9.28	9.16
GPT-4 Preview (11/06)	9.20	9.38	9.03
GPT-4 Omni (05/13)	9.19	9.31	9.07
MoA-Lite	9.18	9.38	8.99
Qwen1.5 110B Chat	8.96	9.23	8.63
Llama 3 70B Instruct	8.94	9.2	8.68
Mixtral 8x22B v0.1	8.78	9.11	8.44
WizardLM 8x22B	8.78	8.96	8.61
Qwen1.5 72B Chat	8.44	8.55	8.34
GPT-4 (06/13)	8.84	9.08	8.61

Table 2: MT-Bench

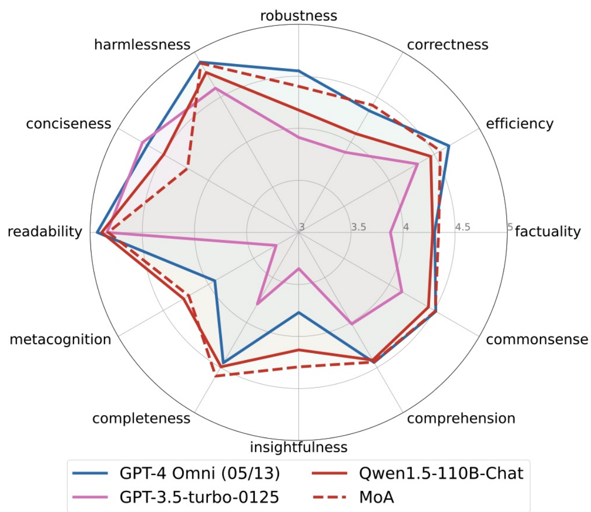

Figure 2: Original FLASK Baseline

Skill	Avg Score
Robustness	4.56
Correctness	4.39
Efficiency	4.62
Factuality	4.45
Commonsense	4.62
Comprehension	4.63
Insightfulness	4.40
Completeness	4.80
Metacognition	4.20
Readability	4.90
Conciseness	4.08
Harmlessness	4.91

Table 3: Replicated FLASK

biases like output length. Table 1 shows the results of the original paper on this benchmark.

MT-Bench is a benchmark for testing multi-turn conversation and instruction-following in language models (Zheng et al., 2023). It uses 80 complex questions and rates responses with GPT-4 to reflect human preferences. The tool's leaderboard ranks top models from companies like OpenAI and Google, providing insights into model strengths in dialogue and reasoning. Table 2 shows the results of the original paper on this benchmark.

FLASK is a fine-grained evaluation framework for language models that assesses their ability to follow user instructions through 12 specific skills, including logical reasoning, knowledge retrieval, and user alignment (Ye et al., 2024). It helps identify strengths and weaknesses in models, using a challenging subset (FLASK-Hard) for rigorous testing. This framework provides a detailed analysis for improving model performance and choosing the right models for specific tasks. Figure 2 shows the results of the original paper on this benchmark.

3.2.2 Replicated Results

We replicated the results for all three baselines. The original paper used Qwen1.5 for all tests, but since Qwen1.5 is now deprecated and Qwen2.5 is the current version, we tested these baselines using Qwen2.5. Below are the results we obtained:

AlpacaEval2.0: The outputs from Qwen1.5 matched the benchmarks from the original paper. However, when we used Qwen2.5, the results showed significant improvement. Table 1 shows the results for the same.

MT-Bench: The outputs from Qwen2.5 were similar to the benchmarks in the original paper. Table 2 shows the results for the same.

FLASK: The results from Qwen2.5 were consistent with the benchmarks outlined in the original paper. Table 3 shows the results for the same.

4 Analysis

4.1 Previous Work Shortcomings

During our evaluative testing, we noted that the mixture of agents framework performs exceedingly well in almost all benchmarks. However, in the FLASK benchmark, we note that the mixture is outperformed in both the conciseness and robustness categories. We completed a qualitative analysis and provided two examples of the mixture generating increasingly verbose answers that may also reduce the robustness score in Table 5.

We find that the answers from the mixture of agents can be unnecessarily verbose. We believe this is because of the way generated outputs are combined between layers. They are simply concatenated into a new prompt and passed into the next layer. The agents in the new layer will have a significant amount of information to process and may increase the length of their answers because of it. Multiple answers may also contradict each other leading to the agent generating contradictions it may otherwise not generate.

We believe that these two evaluations, conciseness and robustness, can be improved by adjusting the way prompts are passed between each layer. We also hypothesize that a fine-tuned or base aggregation model may be able to combine information better by recognizing any contradictions in the provided generations.

4.2 Proposal for Our Work

We find that the fixed structure of the Mixture of Agents may not be optimal for producing state-of-the-art results. We note that between each layer, the outputs of each large language model are concatenated together within a prompt. The prompt given to the next layer is the original prompt with an additional direction to use the concatenated outputs and improve upon them. We believe that this is not optimal and may be increasing the verbosity unnecessarily, decreasing performance relative to previously mentioned benchmarks.

We will experiment with different ways to pass on information between layers. Specifically, we would like to train an inter-layer aggregator using a small, fine-tuned model to see if it can briefly summarize and combine all the results before the next layer of generation. We will also compare different sizes of aggregation models to attempt to replicate larger model performance with smaller LLMs.

5 Methodology

Based on our analysis, we identified a critical shortcoming in the existing approach—its inability to achieve a sufficiently high conciseness score on the FLASK evaluation metric. To address this limitation, we propose an enhanced methodology that incorporates an aggregator mechanism at each layer of the architecture.

In this approach, the concatenated outputs from each layer are summarized by the aggregator before being passed to the subsequent layer. By ensuring that each layer generates a concise and distilled representation of its output, this design aims to improve the overall conciseness of the architecture's representations, thereby enhancing its performance on the FLASK evaluation.

The proposed methodology is illustrated in Figure 3. This diagram highlights how the aggregator operates at each layer, emphasizing the iterative refinement of the output as it propagates through the network. We anticipate that this structured approach will not only improve the conciseness score but also lead to better generalization and robustness of the outputs.

We experimented with two distinct strategies for designing the aggregator architecture, each aimed at improving the conciseness of the architecture outputs:

1. **Out-of-the-Box Large Pre-trained Models:** The second strategy involved using out-of-the-box large pre-trained models for summarization tasks. These models, such as Llama 3.2 3B and Llama 3.1 8B Instruct, were employed directly as aggregators at each layer. This approach takes advantage of the robust language capabilities of large pre-trained models without requiring additional fine-tuning.

2. **Fine-tuning a Smaller Model:** In this approach, we fine-tuned a smaller model, `Llama 3.2 3B Instruct`, specifically for the task of summarizing multiple generations. Once trained, this smaller model was integrated as the aggregator at each layer of the architecture. By leveraging a lightweight summarization model, this method seeks to balance computational efficiency with effective information compression.

5.1 Fine-tuning Methodology

We experimented with the fine-tuning of the following models for the summarization task (Touvron et al., 2023)(Team, 2024):

1. `Llama-3.2-3B-Instruct`
2. `Llama-3.2-3B`
3. `Llama-3.2-1B`
4. `google/gemma-2-2b`
5. `google/flan-t5-xl`

We employed two approaches for fine-tuning:

1. **Base LoRA Fine-tuning:** Applied low-rank updates to the model's linear layers using a standard LoRA configuration. This method provided a straightforward and effective approach to adapt the model to the task while preserving the structure of the base model (Hu et al., 2021).

2. **QLoRA (Quantized Low-Rank Adaptation):** Extended the LoRA approach by incorporating 4-bit quantization for efficient memory and computational use. This allowed us to fine-tune the model with significantly reduced hardware requirements while maintaining comparable adaptation quality (Dettmers et al., 2023).

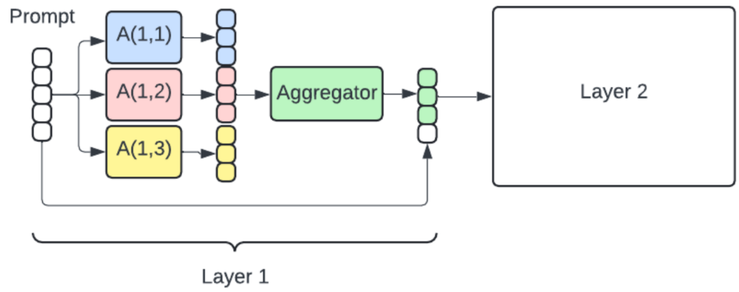

Figure 3: Aggregator Architecture

The training data for fine-tuning consisted of the concatenated outputs of 3 top Community models on the AlpacaEval 2.0 leaderboard (`Shopee SlimMoA v1`, `Blendax.AI-gm-l6-vo31`, `gemma-2-9b-it-WPO-HB`) with the desired output being the final outputs of the `SelfMoA + gemma-2-9b-it-WPO-HB` model on the leaderboard.

We attempted fine-tuning using both approaches with less than satisfactory results, resulting in us not employing the approach for the final evaluation. More details on the issues faced in the fine-tuned model generations are provided in the Discussion section.

5.2 Evaluation Methodology:

Our evaluation methodology involved assessing the models' outputs using the FLASK protocol, which decomposes coarse-level scoring into skill set-level scoring for each instruction. This fine-grained evaluation is crucial for attaining a holistic view of model performance and increasing the reliability of the evaluation. This approach enhances interpretability by assigning scores to 12 defined skills across four primary abilities: Logical Thinking, Background Knowledge, Problem Handling, and User Alignment. By analyzing the models' performance across various skills, domains, and difficulty levels, we aimed to identify specific areas for improvement, particularly focusing on enhancing the conciseness of the models' outputs.

6 Results

Through our fine-tuning experimentation, we encountered significant difficulties in properly fine-tuning a model solely for summarization. During testing of our five fine-tuned models, we observed that the models would generate nearly the same tokens as the input context. This surprising result was replicated for each combination of model, fine-tuning type, and dataset trained on. Because of this, we were only able to run successful evaluations on the base pre-trained Llama-3.1-8B-Instruct-Turbo and Llama-3.2-3B-Instruct models. These results can be found in 4.

Our model selection choice was planned to be an initial comparison between two 3-billion parameter models, one fine-tuned and the other the base model, and a larger 8-billion parameter model. Our original hypothesis was that we could utilize the smaller, fine-tuned model to match the performance of the larger model. Because we were unsuccessful in creating a high-performing, small fine-tuned model, we did a direct comparison to see if a smaller base model could replicate or exceed the conciseness results from a larger model.

We found that the smaller aggregation model could match the conciseness score of a larger model. Both Llama models performed identically in the conciseness category which was our hypothesized result. The smaller model also performed similarly in nearly all other categories which was surprising. This result shows that utilizing a small model is a realistic approach to creating low-cost aggregators that can improve conciseness in the mixture of agents framework.

7 Discussion

Our original goal was to create a fine-tuned model that could increase the conciseness score from the baseline model. During our project hypothesis discussions, we realized that we would need to fine-tune a larger model than we could train and host due to our financial and computing limitations. Therefore, we decided to test our hypothesis on smaller models that we could reliably fine-tune for multiple iterations of testing. We also changed our testing goal to show that a smaller, fine-tuned

Task Category	Base MoA - Qwen 72B	Llama-3.1-8B Instruct	Llama-3.2-3B Instruct
Robustness	4.56	4.06	4.03
Correctness	4.39	4.30	4.26
Efficiency	4.62	4.13	4.09
Factuality	4.45	4.06	4.02
Commonsense	4.62	4.42	4.33
Comprehension	4.63	4.14	4.09
Insightfulness	4.40	3.45	3.43
Completeness	4.80	3.93	3.90
Metacognition	4.20	3.30	3.17
Readability	4.90	4.24	4.25
Conciseness	4.08	3.79	3.79
Harmlessness	4.91	4.63	4.60

Table 4: FLASK with Inter-layer Aggregator

model could replicate the results and improve the conciseness of a larger model. This led us to select smaller one and three-billion parameter models.

Despite numerous iterations and extensive experimentation, we were unable to achieve a reliable, fine-tuned model. As outlined in the Methodology section, we explored various fine-tuning approaches using different datasets and hyperparameter configurations. Specifically, we attempted to fine-tune several large language models (LLMs) for summarization tasks on our curated dataset derived from AlpacaEval 2.0 outputs. However, challenges in applying fine-tuning techniques, such as LoRA and QLoRA, prevented us from effectively evaluating the fine-tuned models. The Gemma2-2B model produced erratic summarization outputs, often unrelated to the input text. For Flan-t5-xl, its limitation of 512 maximum tokens proved insufficient for handling the evaluation suite's generation requirements. Additionally, the Llama models exhibited poor compatibility with LoRA and QLoRA fine-tuning methods, frequently returning the input text as the output. These challenges in additon to being financially limited by the Together API, ultimately hindered our ability to test more iterations of fine-tuned aggregators.

We also note that our tested model did not improve from the initial MoA baseline. Without adequate resources, we would not have been able to fine-tune a model large enough to improve upon conciseness while also maintaining meaningful performance in the other categories. After discussions with our TA and within our team, we decided to scale our experiments to compare smaller models, knowingly sacrificing performance in most categories. This led us to our displayed experiments and results, showing that a smaller aggregator can replicate results of a larger one. We believe that with these results, we show that the idea of an aggregator is a feasible approach to improve the conciseness of a multi-agent layered system. With a larger properly fine-tuned summarization model as an aggregator, we expect to see higher performance against the original displayed baseline.

8 Conclusion

The use of a mixture of agents each utilizing a different LLM is a proven way to increase system performance for a variety of tasks. Using multiple models increases performance overhead and degrades conciseness compared to a single model. Our idea of an aggregator to combine each of these results between each layer was designed to combat this trend to increase verbosity by decreasing the number of input tokens to each layer. We found that smaller aggregators can replicate the results of larger ones but were not successful in utilizing a large enough model to outperform the existing mixture of agents baseline due to experimental limitations. We believe with additional resources and time, the aggregation mixture of agents framework would be able to create comparable results to any FLASK baseline.

References

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. 2024a. Autoagents: A framework for automatic agent generation. *Preprint*, arXiv:2309.17288.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2024b. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *Preprint*, arXiv:2309.13007.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *Preprint*, arXiv:2305.14325.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *Preprint*, arXiv:2404.04475.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pages 51991–52008. Curran Associates, Inc.

Huao Li, Yu Chong, Simon Stepputtis, Joseph Campbell, Dana Hughes, Charles Lewis, and Katia Sycara. 2023b. Theory of mind for multi-agent collaboration via large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 180–192, Singapore. Association for Computational Linguistics.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *Preprint*, arXiv:2310.02170.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *Preprint*, arXiv:1701.06538.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *Preprint*, arXiv:2306.03314.

Gemma Team. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024. Mixture-of-agents enhances large language model capabilities. *Preprint*, arXiv:2406.04692.

Michael Wooldridge. 1999. Intelligent agents. *Multiagent systems: A modern approach to distributed artificial intelligence*, 1:27–73.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *Preprint*, arXiv:2306.02224.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2024. Flask: Fine-grained language model evaluation based on alignment skill sets. *Preprint*, arXiv:2307.10928.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. 2024. Exploring collaboration mechanisms for llm agents: A social psychology view. *Preprint*, arXiv:2310.02124.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Appendix

Prompt	Generated Response
Can a Liebherr LTM 11200-9.1 hypothetically lift Mount everest	The idea of a Liebherr LTM 11200-9.1 crane lifting Mount Everest is a fascinating but entirely hypothetical scenario. Let's break down why this is not feasible:### 1. **Weight and Scale**- **Crane Capacity**: The Liebherr LTM 11200-9.1 has a maximum lifting capacity of 1,200 metric tons (1,322 US tons). This is an impressive feat for a mobile crane, but it pales in comparison to the mass of Mount Everest.- **Mount Everest's Weight**: The mass of Mount Everest is estimated to be around 335 billion metric tons (335,000,000,000,000 kg). This is roughly 279 million times the lifting capacity of the LTM 11200-9.1.### 2. **Size and Shape**- **Crane Boom Length**: The maximum boom length of the LTM 11200-9.1 is 100 meters when using a guying system called Y-guying. This is minuscule compared to the height of Mount Everest, which stands at 8,848.86 meters (29,031.7 feet) above sea level.- **Mountain Structure**: Mount Everest is not a single, uniform object. It is a complex structure composed of layers of rock, ice, and snow, making it impossible to attach a lifting device to it in a practical manner.### 3. **Stability and Base**- **Crane Stability**: The LTM 11200-9.1 requires a stable base to operate. The base area needed to lift such an enormous weight would be impractically large, likely larger than the mountain itself.- **Ground Conditions**: The terrain around Mount Everest is extremely rugged and inhospitable. There are no suitable flat surfaces that could support the crane's base, and the ground would likely collapse under the immense weight.### 4. **Logistical and Environmental Considerations**- **Transportation**: Transporting the LTM 11200-9.1 to the base of Mount Everest would be a monumental logistical challenge. The terrain is inaccessible by road, and the altitude and weather conditions make it extremely difficult to move heavy machinery.- **Environmental Impact**: Even if it were theoretically possible to lift Mount Everest, the environmental impact would be catastrophic. Moving such a massive amount of rock and ice would cause significant geological and ecological damage.### 5. **Engineering and Physics**- **Structural Integrity**: The forces required to lift Mount Everest would exceed the structural integrity of any known material. The crane's boom and other components would likely fail under such extreme stress.- **Energy Requirements**: The energy needed to lift Mount Everest would be astronomical, far beyond the capabilities of any existing power source.### ConclusionThe Liebherr LTM 11200-9.1 is an impressive piece of engineering, but lifting Mount Everest is beyond the realm of possibility. The task is constrained by the mountain's enormous weight, its complex structure, the crane's limited capacity, and the impracticalities of the terrain and logistics. This scenario is purely hypothetical and defies the laws of physics and engineering principles
Question: Would most people believe this scenario reasonable or unreasonable to say? Please answer it either 'reasonable' or 'unreasonable' Scenario: No because he hired the neighbor boy to do it	Answer: Reasonable. The scenario "No because he hired the neighbor boy to do it" can be considered reasonable in many contexts. Hiring someone, such as a neighbor boy, to perform a task is a common and acceptable practice. People often hire others to do tasks for various reasons, including lack of time, skill, or interest. Without additional context, the scenario appears to be a reasonable explanation for why someone did not complete a task themselves. However, the ultimate reasonableness would depend on the specific details of the task and the agreement between the parties involved.

Table 5: Mixture of Experts FLASK examples