BOSS LLM: ADAPTATION VIA NO-REGRET LEARNING

Yu Feng, Avishree Khare, Nghia Nguyen , & Sikata Sengupta
Department of Computer Science
University of Pennsylvania
Philadelphia, PA 19104, USA
{akhare,fengyu1,nghianhh, sikata}@seas.upenn.edu

ABSTRACT

Several Large Language Models (LLMs) are released for public use on a regular basis and tend to have different strengths and weaknesses. For any given prompt to one of these models, multiple categories or topics may apply. In this paper, we focus on learning an adaptive mixture of these base models (our Boss LLM) such that it has provably low regret with respect to every category and expert model. We extend the framework of multi-objective optimization with exponential weights algorithm (MOEW) (provided by (11) for action-independent subsequences) to LLMs. We created the flexible package boss-llm for others to test this framework out on whatever class of models they have available. We ultimately verify that this Boss LLM is able to effectively adapt its mixture based upon the categories of a given prompt and improve upon the base models. We empirically study our Boss LLM's generalization properties and also provide directions for further theoretical and empirical exploration.

1 INTRODUCTION

In the era of rapid progress and development of LLMs, there are many different models from many different providers each of which have strengths and weaknesses across different domains and task categories. In particular, multiple categories can be assigned to any given prompt. In this work, as shown in Figure 1, we aim to explore mechanisms to combine the strengths of all of these different models to create a boss or master LLM that is able to select which model to respond with or use in a context-dependent manner.

1.1 RELATED WORK

In an online adversarial setting, one might seek regret guarantees that are not just with respect to each fixed expert model, but rather conditional on group (or in this context category) membership. In the setting where events are action-independent (as is the case for categories of prompts), (11) provides a nice closed-form algorithm (Multi-Objective Optimization with Exponential Weights). There are also several other lines of work around conditional regret guarantees and multi-objective optimization (6). In the space of Uncertainty Quantification and LLMs for factual correctness (with theoretical multi-group and multi-valid guarantees applied to LLM applications), (8) via calibration and conformal prediction.

Building on the theme of interactive updating processes, recent works on self-improvement for LLMs (19; 10; 13) explore iterative frameworks where models generate multiple outputs, verify their own results, and distill knowledge by selecting and refining the reranked or filtered results. One of the most closely related papers is (21), which formulates the problem of self-improvement as self-play with an LLM, allowing the policy to iteratively play against itself using no-regret learning. Our method adheres to the same philosophy but incorporates more controlled and precise updates for distilling knowledge from multiple LLMs.

In the recent past, there has also been quite a bit of work around the idea of Multi-LLM collaborations. Recent advancements in multi-LLM agent frameworks have resulted in the development of various platforms designed to enable collaboration among multiple agents. AutoGen (15) introduces a framework for building LLM-based applications where agents communicate with each other

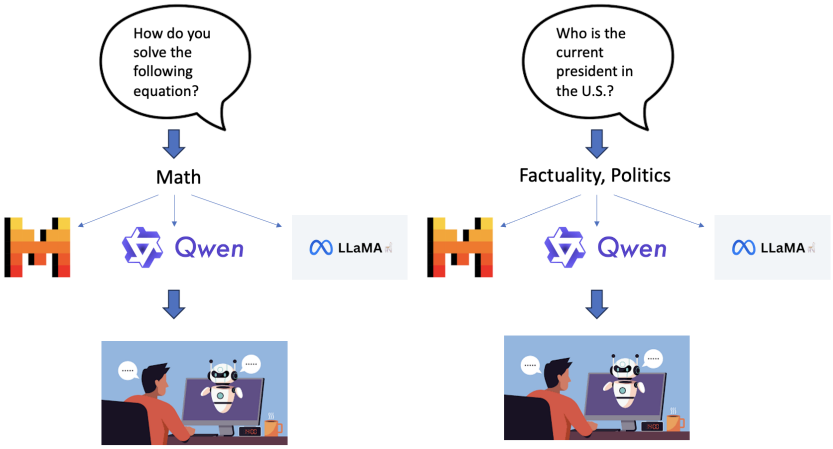


Figure 1: Model Motivation

to accomplish tasks. CAME (7) is designed to enable autonomous collaboration among chat-based language models using role-playing and inception prompting. These frameworks complement recent efforts (16; 3; 5) to improve factuality in LMs through multi-agent cooperation or debate, primarily using cross-model agents. (12) further build "weak-learner" LLMs to do smaller-scale tasks like planning, calling, and summarizing to build up a multi-LLM agent. Unlike the majority of related studies that rely on interactions between agents, our proposed method eliminates the need for inter-LLM interactions. By leveraging the closed-form algorithm, a meta-agent is capable of effectively guiding the behavior of subagents without requiring direct communication among them.

Meanwhile, the idea of LLM-merging has also recently emerged as it is cost-efficient and empirically has shown some signs of success regarding its ability to generalize to new tasks well from specialized fine-tuned existing models (Zhang et al.; 17). In a similar vein, mixture-of-experts models that use sparsity to selectively activate individual expert models have rapidly grown in interest (9). Our work offers a potential next step toward effective mixture-of-agents (14).

2 Model Setup

Suppose we have access to K LLMs, which we denote as $\{LM_i\}_{i\in[K]}$. We define the set of possible prompts to be χ and we will define a corresponding set of categories $\mathcal{C}$. We use the shorthand $c(x)$ to denote the set of categories assigned to prompt x, so we can think of the following function $c : \chi \to 2^{\mathcal{C}}$. Let $N = |\mathcal{C}|$ be the number of categories. We define a collection of events $\mathcal{E}$ as the collection of functions $E_i(t, x_t) = I(c_i \in c(x_t))$ for all $i \in [N]$. We denote the space of responses $\mathcal{Y}$. We may think of a designated y_t or set of responses for each x_t corresponding to a desired or correct response. Next, we formally define the sequential interaction setting between our multi-LLM method and the users of a chat system as a sequential (repeated) game between a learner and an adversary in Def 2.1.

Definition 2.1 (Sequential Interaction Setting)**.** For $t = 1, ..., T$

- Adversary selects categories $C \in 2^{\mathcal{C}}$ and corresponding prompt $x_t \in \chi$, $y_t \in \mathcal{Y}$
- Learner selects $p_t \in \Delta[K]$ as its distribution over experts.
- Adversary selects cost $l_t \in [0, 1]^K$
- Learner experiences the expected loss $\langle p_t, l_t \rangle$ over the choice of the experts.

Here, the adversary embodies how the set of prompts is selected and how the cost is calculated after the learner answers the prompts. The reason we chose this setting is that the categories of the prompts asked by the user and how satisfied the user is with the response of the system may

not follow any distributional assumptions (e.g., identically and independently distributed prompts). Therefore, this setting allows us to have guarantees over potentially arbitrary category choices and values of the loss.

Next, we define the concept of regret in a repeated game, where $i \in [K]$ is a fixed action.

Definition 2.2 (Regret over a sequence compared to taking a fix action (11))**.**

$$Reg(\pi^T, i) = \sum_{t=1}^{T} E(t, x_t)(\langle p_t, l_t \rangle - l_t^i)$$

Intuitively, regret quantifies how much worse playing a repeated game according to the set of distributions $\{p_t\}_{t=1}^{T}$ compared with choosing a fixed action i. For LLMs setting, this helps us quantify how using our method compares with just choosing one LLM.

We can extend regret to also condition on questions belonging to a category (or even set of categories) with the following definition.

Definition 2.3 (Action-Independent Subsequence Regret (11))**.**

$$Reg(\pi^T, E, i) = \sum_{t=1}^{T} E(t, x_t)(\langle p_t, l_t \rangle - l_t^i)$$

For our method, we are choosing the probability of choosing the experts using the Multi-Objective Optimization with Exponential Weights (MOEW) algorithm. The complete algorithm is presented as follows.

Algorithm 1 Multi-Objective Optimization with Exponential Weights (MOEW). This algorithm is from (11)

for $t = 1, ..., T$ **do**
  Adversary selects x_t, c_t
  Learner updates $p^t \in [K]$ as
  $p_i^t = \frac{\sum_{k=1}^{N} E_k(t,x^t) \exp\left(\frac{\eta}{2} \sum_{t'=1}^{t-1} E_k(t',x^{t'})(\langle p^{t'}, l^{t'} \rangle - l_i^{t'})\right)}{\sum_{j=1}^{K} \sum_{k=1}^{N} E_k(t,x^t) \exp\left(\frac{\eta}{2} \sum_{t'=1}^{t-1} E_k(t',x^{t'})(\langle p^{t'}, l^{t'} \rangle - l_j^{t'})\right)}$
  Adversary selects l_t
end for

Using this algorithm, we get the following guarantee on the performance of our method compared to a fixed best LLM.

Theorem 2.4. *Algorithm 1 obtains action-independent subsequence regret of* $4\sqrt{T(\ln(N) + \ln(K))}$ *where* N *is the number of categories,* K *is the number of* $LLMs$*, and* T *is the time horizon. This theorem (and proof) is from (11).*

$$\max_{k \in [N], i \in [K]} Reg(\pi^T, E_k, i) \leq 4\sqrt{T(\ln(N) + \ln(K))}.$$

This result is useful because in a new setting when we do not know what the best LLM is, using our method implies that we are not doing much worse than the best LLM in hindsight.

3 Experimental Setup

3.1 Adversary Simulation

The adversary, given access to transcript up until the current time step will select a prompt (alongside its corresponding category and true answer). To select a loss vector, our adversary is an LLM trained via zero-shot learning to select prompts, categories to follow this sequential interaction protocol. For a given prompt x_t and select correct answer y_t, the adversary will score the responses provided by

the other expert models based upon some examples and overlap with the select correct answer. The adversary will also be provided with access to p_t before selecting its loss vector, though in practice it will likely not affect the adversary's loss assignment in this setting. We consider utilizing a variety of different models for the adversary LLM, and, for our experiments, use Llama-3.1-8B-Instruct (4). For category selection, our main adversary is selecting the worst category performance of the learner in hindsight (weighted by proximity to current time step). We also include ϵ-greedy to encourage some exploration of category selection.

For loss scoring, an example of a prompt to the Adversary LLM is given in Listing 1.

3.2 Categories and Datasets

The main categories we consider in this project are factuality, math, and code. While we don't explicitly include prompts / questions that correspond to multiple categories at once, our framework can be extended to such cases (for example, questions that require retrieving a fact and using that to answer a math problem). In the future, we also hope to broaden this to include the following list from Llama3's human evaluation as well (note that three of these are already included in our current setup):

1. Coding
2. Mathematical reasoning
3. Asking for advice
4. Brainstorming
5. Classification
6. Closed Question Answering
7. Creative Writing
8. Extraction
9. Inhabiting a character/persona
10. Open question answering
11. Rewriting
12. Summarization

We next list the datasets we use for the factuality, math and code categories respectively.

HotpotQA. (18) provide "HotpotQA, which is a new dataset with 113k Wikipedia-based question-answer pairs with four key features: (1) the questions require finding and reasoning over multiple supporting documents to answer; (2) the questions are diverse and not constrained to any pre-existing knowledge bases or knowledge schemas; (3) it provides sentence-level supporting facts required for reasoning, allowingQA systems to reason with strong supervision and explain the predictions; (4) it offers a new type of factoid comparison questions to test QA systems' ability to extract relevant facts and perform necessary comparison." An example of a question from HotpotQA, with the answer and expert responses, is given in Listing 2.

GSM8K. (2) provide "GSM8K (Grade School Math 8K) which is a dataset of 8.5K high quality linguistically diverse grade school math word problems. The dataset was created to support the task of question answering on basic mathematical problems that require multi-step reasoning. These problems take between 2 and 8 steps to solve. Solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations to reach the final answer. A bright middle school student should be able to solve every problem: from the paper, "Problems require no concepts beyond the level of early Algebra, and the vast majority of problems can be solved without explicitly defining a variable. Solutions are provided in natural language, as opposed to pure math expressions." An example of a question from GSM8K, with the answer and expert responses, is given in Listing 3.

Code dataset. The final dataset "contains problem descriptions and code in python language. This dataset is taken from sahil2801/code_instructions_120k, which adds a prompt column in alpaca style." (1). An example of a question from this dataset, with the answer and expert responses, is given in Listing 4.

3.3 EXPERTS

We utilize the following expert models via Hugging Face for our experiments.

- mistralai/Mistral-7B-Instruct-v0.3[1]
- Qwen/Qwen2.5-7B-Instruct[2]
- Qwen/Qwen2.5-0.5B-Instruct[3]
- meta-llama/Llama-3.2-1B-Instruct[4]

We present an example question from HotpotQA along with the corresponding response from each of the five possible experts.

Question from hotpotQA:
prompt: Which actress does this Netflix's project by Nick Bernardone and Tina Fey star in the title role?

category: factual

Answer from hotpotQA:
answer: Ellie Kemper

Adversary response:
To calculate the loss vector, I will evaluate the responses of the experts and the learner's distribution over the experts. Since the category is factual, I will compare the responses to the true answer, Ellie Kemper.learner's distribution over the experts is not provided, but I will assume it is uniform, i.e., the learner is equally uncertain about all experts.0 provides a correct response, mentioning Ellie Kemper as the star of the Netflix project G̈irlboss.̈ Expert 1 provides an incorrect response, stating Emily Blunt as the star. Expert 2 provides an unclear response, stating that they couldn't find any information about a Netflix project starring Tina Fey and Nick Jonas.on this information, I will calculate the loss vector as follows:* For Expert 0, the loss is 0, as the response is correct.* For Expert 1, the loss is 1, as the response is incorrect.* For Expert 2, the loss is 1, as the response is unclear and does not provide the correct answer.vector for the learner: $[0, 1, 1]$

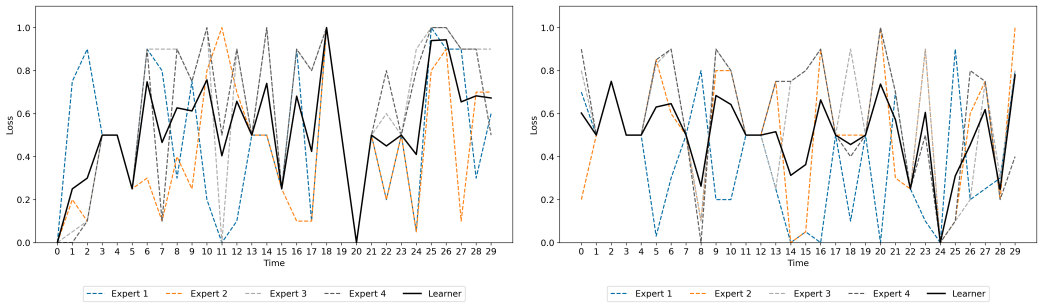
Model	Response	Adversary's Loss
Mistral-7B-Instruct-v0.3	The actress who stars in the title role of the Netflix project by Nick Bernardone and Tina Fey is Ellie Kemper. The project is called G̈irlboss,̈based on the autobiographical book of the same name by Sophia Amoruso. Ellie Kemper plays the character Sophia Marlowe, a fictionalized version of Amoruso.",	0
Qwen2.5-0.5B-Instruct	The actress who stars in the title role of the Netflix project by Nick Bernardone and Tina Fey is Emily Blunt.	1
Llama-3.2-1B-Instruct	I couldn't find any information about a Netflix project starring Tina Fey and Nick Jonas. It's possible that you may be thinking of a different project or incorrect information.	1

Table 1: Expert Responses to Prompt from HotpotQA and losses from the adversary

3.4 GENERALIZATION

We explore a simple notion of generalization for the time-being. That is, we explore how the final round mixture of weights presented by our algorithm fare against the base expert on new examples

[1] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[2] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[3] https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct
[4] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct



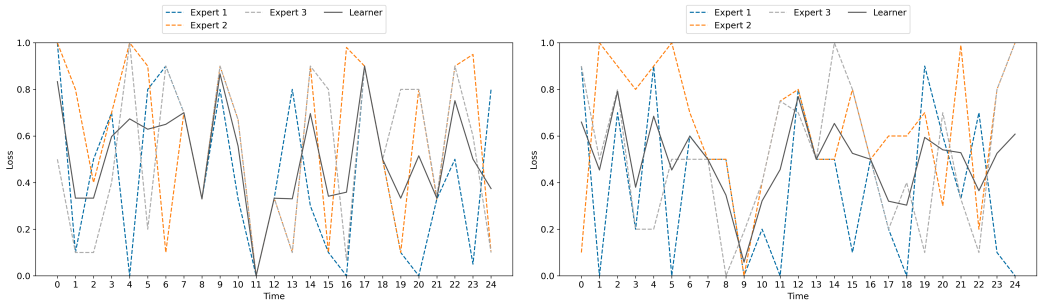
(a) Train loss (b) Test loss

Figure 3: Loss over $T = 30$ rounds incurred by 4 base experts and the learner's selection. The 4 base experts are `Mistral-7B-Instruct-v0.3`, `Qwen/Qwen2.5-7B-Instruct`,`Qwen2.5-0.5B-Instruct` and `Llama-3.2-1B-Instruct` respectively. The adversary is simulated by `Meta-Llama-3-8B-Instruct`.

when it is no longer updating its weights and immediately receiving the loss. We do this to simply check whether this mixture has any properties that make it more successful on new unseen examples across topics (empirically).

4 RESULTS

For our experiments, first we analyze the regret of the learner relative to all the base experts across all categories as shown in Figure 2 and Figure 3. We also provide the subsequence regret for every topic and expert in table 2. We also then consider the question of generalization (Figure 3). That is, we consider how the mixture model of the learner fares on unseen prompts compared to the base model and across topics, when it is no longer receiving immediate information about the loss of all responses.



(a) Train loss (b) Test loss

Figure 2: Loss over T=25 rounds incurred by 3 base experts and the learner's selection. The 3 base experts are `Mistral-7B-Instruct-v0.3`, `Qwen2.5-0.5B-Instruct` and `Llama-3.2-1B-Instruct` respectively. The adversary is simulated by `Meta-Llama-3-8B-Instruct`.

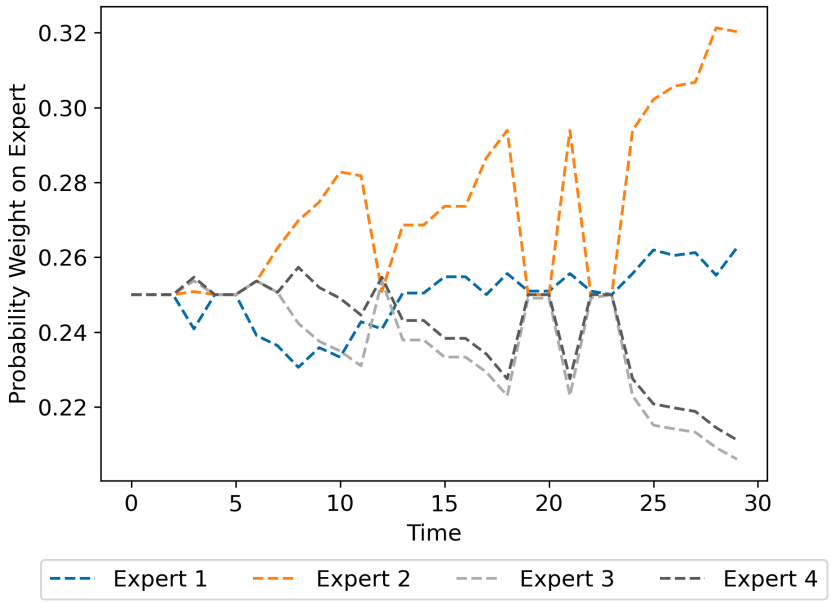


Figure 4: Learner's Distribution over Experts from Figure 3, across $T = 30$ rounds. The 4 base experts are `Mistral-7B-Instruct-v0.3`, `Qwen2.5-7B-Instruct`,`Qwen2.5-0.5B-Instruct` and `Llama-3.2-1B-Instruct` respectively. The adversary is simulated by `Meta-Llama-3-8B-Instruct`.

Category/Model	Mistral-7B-Instruct-v0.3	Qwen2.5-0.5B-Instruct	Llama-3.2-1B-Instruct
Factual	1.701	-1.039	-0.699
Math	-0.036	-0.216	0.297
Code	0.233	-0.467	0.233

Table 2: Subsequence Regret by Expert (from Figure 2) and Category for $T = 25$ and $l_t \in [0, 1]$. A negative regret value means that our learner performs better than the model in the corresponding column at the category in the corresponding row and vice versa.

Category/Model	Mistral-7B-Instruct-v0.3	Qwen2.5-7B-Instruct	Qwen2.5-0.5B-Instruct	Llama-3.2-1B-Instruct
Factual	0.963	3.163	-2.787	-1.987
Math	0.000	0.000	0.000	0.000
Code	0.307	0.257	-0.193	-0.343

Table 3: Subsequence Regret by Expert (from Figure 3) and Category for $T = 30$ and $l_t \in [0, 1]$.

5 Discussion

In Figure 2 we find that when we include one larger model and two smaller models as experts, over time, more weight is put on the expert that is larger (first) model $[0.39, 0.30, 0.31]$ which we may intuitively expect to be the expert performing better. We also broadly see in this figure that the learner is performing competitively with the best fixed expert over the training set. In the test set (regarding generalization) we also see that the learner still performs somewhat competitively. In Figure 3, however, when we include several larger models, we find that on categories where no expert performs particularly better than the others (loss wise over time) that our Boss LLM broadly remains with a uniform distribution over the base experts. In Table 2, we can see for different categories, there are different experts with respect to which we have maximum regret. All of these are quite small (given our time horizon and regret bound). However, in Figure 3 when there are

two larger expert models and two smaller expert models, one can see that more weight goes towards the first two models (as seen in Figure 4, $[0.26, 0.32, 0.21, 0.21]$) and the learner performs relatively competitively with the best fixed expert in hindsight.

We include plots of the evolution of probability mass put on each expert across time as well, as can be seen in Figure 4, to study how the learner is updating sequentially across categories. Recall that in this example, since the first two experts are larger in terms of number of parameters than the subsequent expert models, one can see that over time the learner is putting more weight on the first two models than the latter two. Moreover, we can see from Table 3 that for the categories sampled (here mainly Factual and Code), the subsequence regret values are for the most part quite low. In the future, we intend to evaluate several other ablations on our experiments. First, we want to consider the role of enabling the adversary to evaluate the loss of all responses combined vs. individual evaluation of each expert's response separately. We also want to evaluate how the adversary generates losses when looking at the transcript vs just the current round expert responses.

Of course, our adversary is not necessarily always playing its best response against the learner's selected distribution and our losses are not necessarily always reflective of underlying properties of the quality of the responses generated by the expert. In the next section, we propose some directions for future exploration to address this point.

6 Conclusion and Future Work

In this work, we apply the MOEW algorithm of (11) to multiple expert LLM models to enable adaptation based upon various possibly intersecting categories of prompts. We empirically verify the theoretical results by showcasing how our Boss LLM models is competitive with the best of the base models across topics (especially in the limit as the horizon becomes larger). Some of the strengths of this approach is that it enables the Boss LLM to select weights according to the various categories that the prompt is a member of in a clean fashion without needing to handle disjoint categories and weights. Moreover, it has provable guarantees on how much worse than the base class of models it can do across all possible topics which enables one to select parameters according to their specific objectives and goals.

Some of the limitations of the current approach is our reliance on a large language model to simulate the adversary. That is, first, our adversary may not select the worse case action against the learner in certain rounds. Moreover, we rely on the general success of few-shot learning to teach the adversary how to score the responses generated by the expert model LLMs. Lastly, we are also in the full information setting which can become costly to generate for large K.

We also assume that each prompt belongs to a single category. Future work could explore more complex prompts that may fall under multiple categories. Another limitation of our current pipeline is that we do not currently generate a final answer for each prompt question. We leave it for future work to aggregate expert responses to a single, high quality output for evaluating final task performance.

We propose a few possible extensions to address some of limitations above. First, we could consider trying to calibrate the loss function of the adversary if we are provided with some labeled human feedback of possible responses to given prompts. We could also consider the bandit information setting rather than the full-feedback setting as this may be more realistic and practical. Lastly, we could consider extending this into other settings by considering no-swap regret versions of such algorithms. We could also consider generalizations of the following approach via Reinforcement Learning with Human Feedback (RLHF).

AUTHOR CONTRIBUTIONS

Equal contribution.

ACKNOWLEDGMENTS

REFERENCES

[1] (2024). Dataset card for python_code_instructions_18k_alpaca. *Hugging Face*.

[2] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

[3] Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. (2024). Improving factuality and reasoning in language models through multiagent debate.

[4] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

[5] Feng, S., Shi, W., Wang, Y., Ding, W., Balachandran, V., and Tsvetkov, Y. (2024). Don't hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. *arXiv preprint arXiv:2402.00367*.

[6] Haghtalab, N., Jordan, M., and Zhao, E. (2024). A unifying perspective on multi-calibration: Game dynamics for multi-objective learning. *Advances in Neural Information Processing Systems*, 36.

[7] Li, G., Hammoud, H. A. A. K., Itani, H., Khizbullin, D., and Ghanem, B. (2023). CAMEL: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[8] Liu, T. and Wu, Z. S. (2024). Multi-group uncertainty quantification for long-form text generation. *arXiv preprint arXiv:2407.21057*.

[9] Lo, K. M., Huang, Z., Qiu, Z., Wang, Z., and Fu, J. (2024). A closer look into mixture-of-experts in large language models. *arXiv preprint arXiv:2406.18219*.

[10] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. (2023). Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[11] Roth, A., editor (2024). *Learning in Games*. Philadelphia, PA.

[12] Shen, W., Li, C., Chen, H., Yan, M., Quan, X., Chen, H., Zhang, J., and Huang, F. (2024). Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324*.

[13] Song, Y., Zhang, H., Eisenach, C., Kakade, S., Foster, D., and Ghai, U. (2024). Mind the gap: Examining the self-improvement capabilities of large language models. *arXiv preprint arXiv:2412.02674*.

[14] Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., and Zou, J. (2024). Mixture-of-agents enhances large language model capabilities.

[15] Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., and Wang, C. (2023). Autogen: Enabling next-gen llm applications via multi-agent conversation.

[16] Xiong, K., Ding, X., Cao, Y., Liu, T., and Qin, B. (2023). Examining inter-consistency of large language models collaboration: An in-depth analysis via debate. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7572–7590, Singapore. Association for Computational Linguistics.

[17] Yang, E., Shen, L., Guo, G., Wang, X., Cao, X., Zhang, J., and Tao, D. (2024). Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

[18] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[19] Zelikman, E., Wu, Y., Mu, J., and Goodman, N. (2022). STar: Bootstrapping reasoning with reasoning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Zhang et al.] Zhang, Y., Ding, Y., Wu, J., and Yang, Y. Llm merging competition technical report for neurips 2024: Efficiently building large language models through merging. In *LLM Merging Competition at NeurIPS 2024*.

[21] Zhang, Y., Yu, D., Peng, B., Song, L., Tian, Y., Huo, M., Jiang, N., Mi, H., and Yu, D. (2024). Iterative nash policy optimization: Aligning llms with general preferences via no-regret learning. *arXiv preprint arXiv:2407.00617*.

A APPENDIX

A.1 ADVERSARY SIMULATION

```
System: You are a judge evaluating a learner on K tasks.
The learner has access to N experts who provide responses to the tasks.
Your task is to provide a loss vector over experts corresponding to a
given question/prompt, the responses of the experts, and the category of
the question/prompt. The loss vector should be in the range [0,1] but
the values need not sum to 1.
For example, if the true answer is closest to the response of the first
expert and farthest from the responses of the other three experts, the
loss vector could be [0.1, 0.9, 0.9, 0.9] if there are four experts and
learner is equally sure about all four experts.
On the other hand, if all the four experts are correct, the loss vector
would be [0, 0, 0, 0]. The learner's response is provided as it's
distribution over the experts.
Your response should start with some reasoning about how the loss vector
is calculated and should strictly end with the following: 'Loss vector
for the learner: [<YOUR LOSS VECTOR OF LENGTH N>]'.

User: Here are your interactions with the learner:
Start of round t=3:

Category chosen: factual
Question from the category to the learner: Command and Control was a
book that discusses the 1980 incident in what state?

True Answer to the question: Arkansas

Response from expert 0 to the question:  The book \"Command and
Control\" by Eric Schlosser primarily focuses on the 1989 Titan II
missile accident that occurred in Damascus, Arkansas.

Response from expert 1 to the question: New Mexico

Learner's distribution over experts: [0.49999062 0.50000938]

Loss vector for the learner: ?.

Let's calculate the loss vector of length 2 for round t=3:

Response:
```

Listing 1: An example of the adversary's prompt to score expert responses

```
Question: Command and Control was a book that discusses the 1980
incident in what state?

Answer: Arkansas

Response (Expert 0): The book "Command and Control" by Eric Schlosser
primarily focuses on the 1989 Titan II missile accident that occurred in
Damascus, Arkansas.

Response (Expert 1): New Mexico
```

Listing 2: An example from HotpotQA

```
Question: On Monday, Matt worked for 450 minutes in his office.  On
Tuesday, he worked half the number of minutes he worked on Monday.  On
Wednesday, he worked for 300 minutes.  How many more minutes did he work
on Wednesday than on Tuesday?

Answer: On Tuesday Matt worked 450 minutes / 2 = <<450/2=225>>225
minutes.
On Wednesday Matt worked 300 minutes - 225 minutes = <<300-225=75>>75
minutes more.
#### 75

Response (Expert 0): Matt worked for 450 minutes on Monday. On Tuesday,
he worked half of that, which is 450/2 = 225 minutes. On Wednesday, he
worked for 300 minutes. So, he worked 300 - 225 = 75 more minutes on
Wednesday than on Tuesday.

Response (Expert 1): To find the difference in minutes worked on
Wednesday and Tuesday, we need to subtract the minutes worked on Tuesday
from the minutes worked on Wednesday.\n\nWednesday: 300
minutes\nTuesday: 450 minutes\n\n300 - 450 = -150 minutes\n\nSo, Matt
worked 150 minutes less on Wednesday than on Tuesday.
_
```

Listing 3: An example from GSM8k

```
Question: Implement a Python program to check if a number is divisible
by another number. Input:7, 2

Answer:
def is_divisible(n, m):
    if (n % m == 0):
        return True
    else:
        return False
print(is_divisible(7, 2))  # Outputs False

Response (Expert 0):
```python

def is_divisible(num, divisor):
    """
    Check if the first number is divisible by the second number.
    Parameters:
        num (int): The number to check.
        divisor (int): The number to divide by.
    Returns:
        bool: True if num is divisible by divisor, False otherwise.
    """
    return num % divisor == 0

# Example usage:
num = 7
divisor = 2
print(is_divisible(num, divisor))  # Output: True
```

Response (Expert 1):
```python
def check_divisibility(num1, num2):
    if num2 == 0:
        print("Number 2 is not divisible by number 1")
    elif num1 % num2 == 0:
        print(f"{num1} is divisible by {num2}")
    else:
        print(f"{num1} is not divisible by {num2}")

# Test the function
check_divisibility(7, 2)
```
_
```

Listing 4: An example from the Code dataset