MALLM: Multi-Agent Large Language Models Framework

Jonas Becker[1,2,*], Lars Benedikt Kaesberg[1,*], Niklas Bauer[1], Jan Philip Wahle[1], Terry Ruas[1], Bela Gipp[1]
[1]University of Göttingen, Germany; [2]LKA NRW, Germany
[*]{jonas.becker, l.kaesberg}@uni-goettingen.de

Code github.com/Multi-Agent-LLMs/mallm
Demo mallm.gipplab.org
Package pypi.org/project/mallm

Abstract

Multi-agent debate (MAD) has demonstrated the ability to augment collective intelligence by scaling test-time compute and leveraging expertise. Current frameworks for MAD are often designed towards tool use, lack integrated evaluation, or provide limited configurability of agent personas, response generators, discussion paradigms, and decision protocols. We introduce MALLM (Multi-Agent Large Language Models), an open-source framework that enables systematic analysis of MAD components. MALLM offers more than 144 unique configurations of MAD, including (1) agent personas (e.g., `Expert`, `Personality`), (2) response generators (e.g., `Critical`, `Reasoning`), (3) discussion paradigms (e.g., `Memory`, `Relay`), and (4) decision protocols (e.g., `Voting`, `Consensus`). MALLM uses simple configuration files to define a debate. Furthermore, MALLM can load any textual Hugging Face dataset (e.g., MMLU-Pro, WinoGrande) and provides an evaluation pipeline for easy comparison of MAD configurations. MALLM enables researchers to systematically configure, run, and evaluate debates for their problems, facilitating the understanding of the components and their interplay.

1 Introduction

Multi-agent debate (MAD) has emerged as a new paradigm to solve complex tasks with multiple large language models (LLMs) (Chan et al., 2024; Du et al., 2023; Liang et al., 2024; Wang et al., 2024b). Yet, we have not understood the exact mechanisms of when and why MAD is successful. Different hypotheses exist around whether MAD is another way to scale test-time compute (Yang et al., 2025), or whether the combination of individual components has emergent capabilities (Liang et al., 2024). Understanding these mechanisms requires a systematic evaluation, specifically code that enables adjusting one variable of the MAD at a time to measure its effect.

Recent work (Guo et al., 2024; Tran et al., 2025; Tillmann, 2025) has identified several key aspects influencing multi-agent discussions. We focus on the following three main components: (1) **agents** define "who" is participating in the debate, meaning the personas of agents and their response style (Wang et al., 2023; Xu et al., 2023); (2) **discussion paradigms** determine "how" the debate is taking place, including agent response order and turn boundaries, structuring information flow (Yin et al., 2023); (3) **decision protocols** choose "what" the debate result will be, meaning deciding when discussions end and determining a final answer (Chen et al., 2023a; Kaesberg et al., 2025). As later evaluations will demonstrate, each of these components is crucial in downstream tasks. Adjusting them individually is particularly important for systematic investigations.

To satisfy the growing demand for MAD applications, many frameworks have been developed (Wu et al., 2023; Gao et al., 2024; Hong et al., 2023). Yet these frameworks intertwine the definitions of multiple components, such as agents and discussion paradigms (Wu et al., 2023) or do not allow for adjusting specific parts, such as discussion paradigms or decision protocols (Hong et al., 2023), hindering independent experimentation. Existing approaches often constrain experimentation by using predefined agents, discussion paradigms, or decision protocols (Zhuge et al., 2024; Gao et al., 2024). Some frameworks also specialize in particular use cases, such as tool usage (OpenAI, 2024), and typically lack integrated evaluation pipelines for analyzing individual MAD components systematically (Wu et al., 2023; Gao et al., 2024). To the best of our knowledge, no current framework exists to evaluate the three core components of related

[*]Equal contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 418–439
November 4-9, 2025

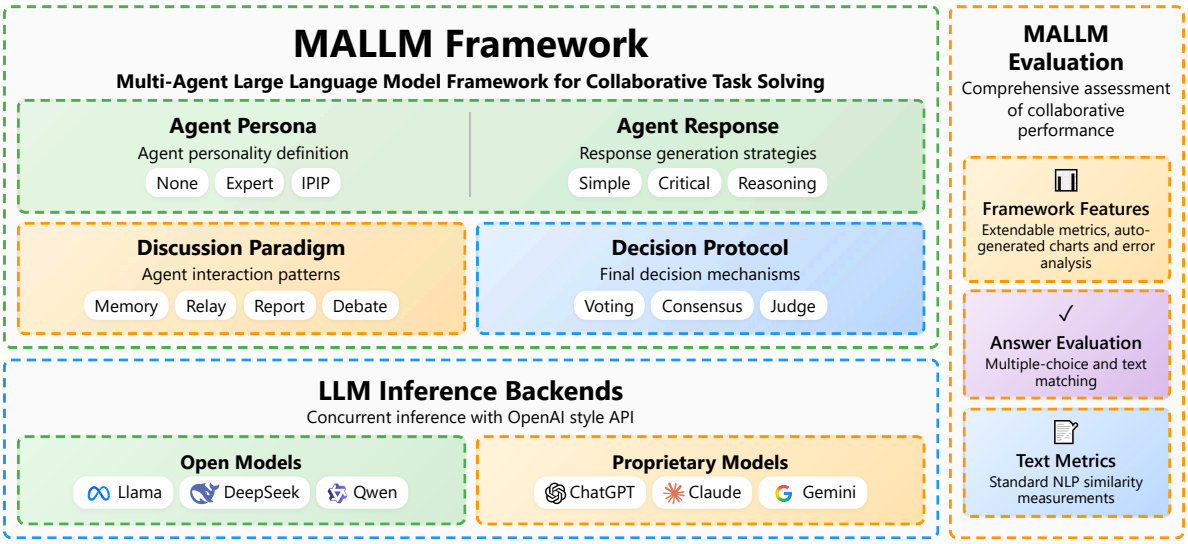

Figure 1: Overview of the MALLM framework and its components.

works and their interactions for MAD: agents, discussion paradigms, and decision protocols.

We propose the open-source Multi-Agent LLM (MALLM) framework to address these limitations (see Figure 1 for an overview). MALLM integrates several MAD components from previous research. Researchers can individually configure their debate setup through parameter settings and easily extend the framework by inheriting existing classes or using template functions. Without additional programming effort, MALLM supports more than 144 distinct MAD configurations. Thus, it enables researchers to reproduce prior MAD experiments, apply MAD methods to new datasets or tasks, and systematically ablate individual MAD components to analyze their impact. We present MALLM's capabilities on our demo website[1].

The key contributions are:

- We propose MALLM, an open-source framework that supports studies of MAD by enabling controlled variation of agents, discussion paradigms, and decision protocols.
- We allow researchers to experiment with 144 existing MAD configurations on a wide range of text-based tasks through automated dataset loading, preprocessing, and evaluation.
- We provide abstract classes and template functions to implement new MAD components and tasks, allowing others to understand the best configuration for their specific research goals.

[1] mallm.gipplab.org

2 Related Work

We identify three key components of related work that are commonly discussed in MAD. Wang et al. (2023); Xu et al. (2023) use **agents** with varying personas and response styles. Yin et al. (2023); Li et al. (2024) define **discussion paradigms** that determine turn order and information flow. Chen et al. (2023a); Yang et al. (2024) explain variations in **decision protocols** that aggregate agent outputs into a single solution.

Existing MAD frameworks vary across the core components described previously: agents, discussion paradigms, and decision protocols. AutoGen supports multi-turn interactions between customizable agents but does not separate discussion paradigms from agent definitions, hindering systematic studies of each component (Wu et al., 2023). It also does not support integrated evaluation pipelines, which need to be coded externally. Similarly, MetaGPT assigns tasks to specialized agents that follow standard operating procedures, tightly coupling agent roles and response styles, but restricting experimentation with alternative discussion paradigms or decision protocols (Hong et al., 2023).

Other frameworks offer alternative abstractions. GPTSwarm models agent interactions as optimizable computational graphs, focusing on information flow rather than the modular comparison of agents or decision protocols (Zhuge et al., 2024). AgentScope simplifies interactions using predefined pipelines (Gao et al., 2024), constraining discussion paradigms and

limiting evaluation of agent personas or decision protocols. The OpenAI Agents SDK coordinates tool-using agents (OpenAI, 2024), prioritizing agent functionality but lacking customizable decision protocols for MAD.

A common limitation across existing frameworks is the tight coupling among agents, discussion paradigms, and decision protocols, which hinders the analysis of each component independently or in combinations of choice. This makes the investigation of which specific components contribute and should be used for particular use cases difficult. Most frameworks provide fixed orchestration setups, restricting experimentation with alternative decision protocols or agent configurations (OpenAI, 2024; Zhuge et al., 2024), and few include integrated evaluation pipelines (Wu et al., 2023; Gao et al., 2024; Smit et al., 2023). No current framework explicitly supports the systematic analysis of individual MAD components and their interactions. Our proposed framework, MALLM, addresses these limitations with a modular architecture clearly separating agents, discussion paradigms, and decision protocols into interchangeable modules. This design enables a systematic study of each component independently and combined, supported by an integrated evaluation pipeline. A comparison of MALLM and other frameworks for MAD is included in Table 5 of Appendix C.

3 MALLM Framework

We propose MALLM, a framework for MAD. It coordinates agents to solve text-based tasks (Guo et al., 2024). MALLM receives an input task and outputs a solution after performing a MAD. Figure 1 illustrates the components of MALLM.

MALLM implements three **agent personas** (`None`, `Expert`, `IPIP`), three **agent response generators** (`Simple`, `Critical`, `Reasoning`), four **discussion paradigms** (`Memory`, `Relay`, `Report`, `Debate`), and three main **decision protocols** (`Voting`, `Consensus`, `Judge`). Each of the component variants can be parameterized individually, allowing systematic comparison of individual setups without additional code.

The agents participating in the debate can use most proprietary and open models, as MALLM supports any OpenAI-compatible API endpoint for inference. The integrated evaluation pipeline can be used to analyze the large amounts of data generated by MAD in a unified way and directly generate comparative charts to visualize performance across different configurations. We provide more details on the framework parameters in Appendix E and prompts in Appendix G.

With MALLM, users can explore the effects of changing components within MAD. The effects of each variation in MAD can be measured towards solving various text-based problems, such as mathematical reasoning (Cobbe et al., 2021), ethical question-answering (Hendrycks et al., 2021a), and more. Our public demo includes three persona generators, three response generators, four discussion paradigms, and four decision protocols[2]. Thus, users can explore 144 MAD configurations directly, observing the effect of parameter combinations. A screenshot of our interactive demo is in Figure 6 of Appendix B.

3.1 Agents

An **agent** is defined by its role (persona generator) and its answer style (response generator).

Persona Generator. The persona specifies agent behavior by their system prompt, e.g., expertise or personality (Xu et al., 2023; Wang et al., 2023). Personas are created iteratively to be complementary and unique. We include three persona types: `None`, `Expert`, and `IPIP`.

None: Disables the persona generation. It assigns each agent a generic name ("Participant N") for baseline experiments. **Expert**: Creates domain-specific personas aligned with the task description (Xu et al., 2023). Examples are an "Educator" for machine learning explanations, a "Software Developer" for app development, or a "Chef" for cooking tasks. **IPIP**: Based on the Big Five personality traits, using the open-source IPIP-NEO classification (Costa and McCrae, 1992; Goldberg, 1999). They cover Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness. The items originate from the International Personality Item Pool (IPIP), frequently used in Psychology (Maples et al., 2014). This enables the detailed modeling of psychological diversity (Serapio-García et al., 2023; Sorokovikova et al., 2024).

Response Generator. The response generator produces agent responses in a specified format or style, influencing how agents interact (e.g., neutral

[2] mallm.gipplab.org

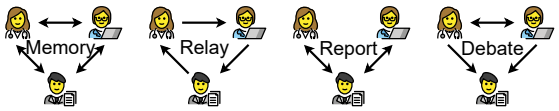

Figure 2: Overview of four discussion paradigms.

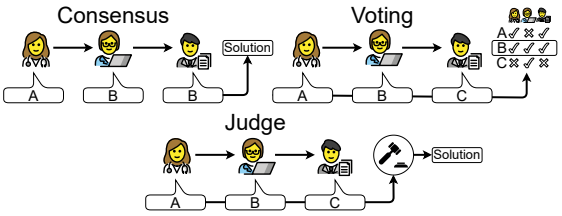

Figure 3: Overview of three main decision protocols.

or critical) (Mizrahi et al., 2024). They vary in how the agents are prompted to generate a response in the debate. MALLM includes the `Simple`, `Reasoning`, and `Critical` response generators. We include their prompts in Appendix G.3.

Simple: Produces free-text responses in a neutral tone, explicitly indicating agreement or disagreement. **Reasoning**: Responds step-by-step, including analysis, alternatives, and conclusions. Agents share their reasoning but not solutions, encouraging independent idea generation. **Critical**: Tasks the agent to identify weaknesses, question assumptions, and suggest alternative approaches.

3.2 Discussion Paradigm

The **discussion paradigm** defines the structure of agent interaction. It specifies turn-taking and information access rules for the MAD. We implement four paradigms (Yin et al., 2023): `Memory`, `Report`, `Relay`, and `Debate`. Each paradigm differs in information flow and visibility, as illustrated in Figure 2.

Memory: All agents have full visibility into each other's messages across turns. **Relay:** Information is passed sequentially between agents in a chain, with only the last message visible to the next. **Report:** Agents independently solve the tasks and report back to a central agent. **Debate:** Agents argue in pairs, taking turns to debate intermediate conclusions before a central agent is consulted.

3.3 Decision Protocol

Each agent in a multi-agent system generates solution drafts. **Decision protocols** systematically determine when discussions end and combine agent-generated solutions into a final decision.

The MALLM framework implements three decision protocol families: `Consensus`, `Voting`, and `Judge`, as illustrated in Figure 3. Debates can run with a fixed number of turns or perform early stopping upon a successful decision.

Consensus: Consensus decision protocols decide on an answer by having the agents converge on one solution. The solution is selected when a required level of agreement among agents is reached (Yin et al., 2023). MALLM includes three agreement levels: `Majority Consensus` (over 50%), `Supermajority Consensus` (over 66%), and `Unanimity Consensus` (100%). **Voting**: Uses a fixed number of discussion rounds (default three, following findings by Du et al. (2023)) before agents vote. In the event of a tie, we run an additional round of debate and voting. Variants are inspired by Yang et al. (2024) and include `Simple Voting` (i.e., each agent votes for their preferred solution), `Approval Voting` (i.e., multiple acceptable solutions per agent), `Ranked Voting` (i.e., agents rank solutions, best cumulative rank selected), and `Cumulative Voting` (i.e., agents allocate up to 25 points across solutions, highest total points selected). **Judge**: Relies on one agent reviewing all solutions, choosing either a preferred one or synthesizing a new solution. The effectiveness of the Judge protocol depends on the model's reasoning capabilities (Zheng et al., 2023).

3.4 Evaluation

The MALLM framework includes a pipeline for evaluating MAD configurations, producing statistics and charts for a dataset and its metrics.

Datasets. The pipeline provides integrated loaders for reasoning tasks (e.g., WinoGrande (Sakaguchi et al., 2020), StrategyQA (Geva et al., 2021)) and knowledge tasks (e.g., GPQA (Rein et al., 2023), MMLU-Pro (Wang et al., 2024c)) as well as core tasks of text generation (Becker et al., 2024), such as paraphrasing (Kovatchev et al., 2018) or summarization (Narayan et al., 2018). It further supports any textual Hugging Face dataset for problem-solving. Researchers can also add their own dataset via subclassing. All datasets are converted into a unified format for processing.

Metrics. We include question-answering and free-text evaluations. For question-answering, we compute accuracy by comparing selected response letters against reference solutions via regex. For free-text tasks, we include BERTScore (Zhang et al., 2020) and textual overlap measures (BLEU (Papineni et al., 2002), ROUGE-1/2/3/L (Lin,

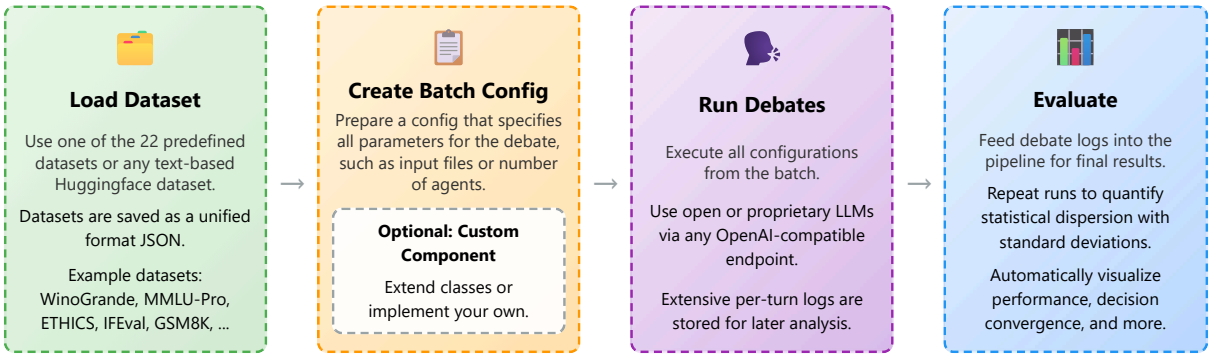

Figure 4: Example workflow of experimenting with MAD using MALLM. First, we can load a dataset. Second, a config file defines the MAD. Third, the debates run and produce output logs. Last, the debates are evaluated. While MALLM already comes with many parameters and components to test, researchers can optionally incorporate their own components, which are tailored to their specific experiment.

2004), and METEOR (Banerjee and Lavie, 2005)).

Statistical variance. We find that several studies on MAD do not account for statistical variance (Wu et al., 2023; Talebirad and Nadiri, 2023), yet it can have marked impacts on MAD (Smit et al., 2023). MALLM enables repeated experiments and calculates standard deviations between them, thereby quantifying statistical dispersion.

Automatic charts. MALLM can visualize the evaluation results of MAD configurations. For this, researchers can pass a single file or a directory with multiple evaluation results directly to an automatic chart generator. Examples of the generated charts by the evaluation pipeline can be seen in Figures 7 to 10 of Appendix F.

4 Application

We explain the workflow for using MALLM to evaluate a specific MAD configuration. First, we can load any text-based Hugging Face dataset for problem-solving via our dataset loader (e.g., MMLU-Pro (Wang et al., 2024c)). Second, MALLM comes with a configuration file (cf. Appendix H) that the user can change to adjust parameters and the components for the desired experiment (e.g., using `Llama-3.3-70B-Instruct` as a model and changing the configuration to `four` debating agents, `Relay` discussion paradigm, and `Unanimity Consensus` decision protocol). Third, we can run the experiment specified by the configuration file. MALLM keeps extensive logs of all debates. They include each agent's messages, votes cast, used components, and configuration parameters. Lastly, we can directly feed the logs into our evaluation pipeline, computing metrics (e.g., accuracy) and leveraging automatic chart generation (cf. Figures 7 to 10 in Appendix F).

Each component (agent, discussion, decision) comes with abstract base classes to describe the pipeline of MAD. If necessary, a custom component could be provided by inheriting features from a component's abstract base class. For example, Sketch-of-Thought (Aytes et al., 2025) is a variant of Chain-of-Thought (Wei et al., 2023) that restructures how models express intermediate steps of their reasoning. To integrate with MALLM, we would create a custom class `SoTResponseGenerator` inheriting from the abstract base class `ResponseGenerator`. Then, the components of MAD can be defined by a configuration file as usual. Figure 4 provides an overview of the workflow for conducting and evaluating MAD with MALLM.

4.1 Use Cases

The following examples illustrate possible research directions that can be realized using MALLM.

Agents on paradigms. The number of agents for the MAD is modifiable. Researchers can test the impact of this on the various discussion paradigms. For example, a study could compare two, three, four, or five agents on the memory and relay paradigm. As these paradigms differ in their information flow, it would be interesting to see how this impacts task performance.

Testing new task. JailbreakBench (Chao et al., 2024) measures the safety of LLMs against jailbreaks. One avenue could be the comparison of multi-agent safety with a single-agent setup. For this, we can use MALLM, which is plug-and-play with a template configuration set.

Simple	Critical	Reasoning
58.6±1.6	**61.4**±3.3	52.2±2.8

Table 1: Comparison of accuracy averaged over all voting-based decision protocols using the simple, critical, and reasoning response generators on the StrategyQA dataset. Best is bolded. ± shows standard deviation over three runs.

CoT	Memory	Relay	Report	Debate
56.9±1.8	60.8±2.6	**62.9**±1.6	60.9±3.1	61.9±1.1

Table 2: Accuracy of MAD on StrategyQA with different discussion paradigms. Best and worst are bolded. ± shows standard deviation over five runs.

Finetuned agents. MALLM works with any proprietary and open-source LLM, which means that we can also provide our own model for the agents. A promising direction is to finetune an agent to enhance its argumentation skills, thereby improving task performance on reasoning tasks such as StrategyQA (Geva et al., 2021).

Moderated paradigm. A dynamic moderator can be implemented by subclassing the abstract base class `DiscussionParadigm`. We can define logic for an LLM-based moderator agent to adjust speaking order based on previous agent contributions. This enables an investigation into the effects of adaptive moderation.

4.2 Example Experiments with MALLM

MALLM can be applied to various use cases. To demonstrate the opportunities for experimental setups, we provide some example investigations. Supplementary information, such as used models and parameters, can be found in Appendix E.

Agents. Kaesberg et al. (2025) use MALLM to experiment with response generators on the StrategyQA dataset (Geva et al., 2021). Table 1 presents the average accuracy across all voting-based decision protocols, using the `Memory` discussion paradigm and `Expert` personas with the `Simple`, `Critical`, and `Reasoning` response generators. The `Critical` response generator slightly improves performance by encouraging agents to critically evaluate responses from others, resulting in a 2.8% point increase. The `Reasoning` response generator decreases performance by 6.4% points, likely because it imposes a strict response structure. Strictly structured responses can degrade the task performance of MAD, a characteristic that was previously noted in single-agent setups (Tam et al., 2024). To summarize, invoking strict response patterns from agents can harm task performance, while prompting agents to think critically can boost it.

Dataset	Voting	Consensus
Knowledge-Based		
MMLU	51.7 ± 2.4	**54.0** ± 2.7
MMLU-Pro	31.1 ± 3.5	**36.0** ± 1.8
GPQA	29.7 ± 2.5	**31.0** ± 2.4
Reasoning-Based		
SQuAD 2.0	**56.7** ± 1.6	43.6 ± 1.5
StrategyQA	**58.6** ± 2.0	58.4 ± 1.6
MuSR	**54.8** ± 1.9	28.4 ± 2.6

Table 3: Mean performance for voting and consensus decision protocols on knowledge and reasoning tasks. Best is bolded. ± shows standard deviation over three runs.

Discussion Paradigms. Becker (2024) compares discussion paradigms on the StrategyQA dataset (Geva et al., 2021), using `Expert` personas, the `Simple` response generator, and `Majority Consensus`. MAD runs until the agent agrees to a solution or until a maximum of seven turns is reached. We find that, using `Majority Consensus`, most debates reach an agreement and end within the first three turns. Thus, seven turns provide a reasonable headroom for this experiment. The results reveal two findings. First, Table 2 compares the discussion paradigms of MAD against a single LLM baseline with Chain-of-Thought (Wei et al., 2023). All paradigms outperform a single LLM with Chain-of-Thought prompting on StrategyQA, improving accuracy by up to 4.0%. Second, we investigate the impact of information transparency on convergence speed for MAD. We find that the very transparent paradigm `Memory` enables faster consensus (avg. 1.75 turns), while limited visibility between agents in `Relay` slows it down (avg. 2.61 turns). Thus, information transparency can lead to quicker convergence in MAD without sacrificing task performance. In summary, MAD can outperform Chain-of-Thought on reasoning tasks, such as StrategyQA, while the information transparency of the discussion paradigm impacts the convergence speed of MAD.

Decision Protocols. Investigations by Kaesberg et al. (2025) compare the average of all `Voting` and `Consensus` decision protocols across knowledge and reasoning tasks, summarized in Table 3. They use the `Simple` response generator and

the `Memory` discussion paradigm. `Consensus` consistently outperforms `Voting` on knowledge tasks (MMLU (Hendrycks et al., 2021b), MMLU-Pro (Wang et al., 2024c), GPQA (Rein et al., 2023)), achieving approximately 2.8% higher accuracy due to repeated verification steps. `Voting` protocols significantly improve accuracy by about 13.2% on reasoning-intensive tasks (SQuAD 2.0 (Rajpurkar et al., 2018), StrategyQA (Geva et al., 2021), MuSR (Sprague et al., 2024)), benefiting from diverse reasoning paths. To summarize, the selection of the decision protocol depends on the specific task. When chosen correctly, it can notably improve task performance.

Creating Demo Examples. For the demonstration of MALLM, we create a set of 144 different example configurations using MALLM's batch feature, called DEBATE (Diverse Exchanges Between Autonomous Talking Entities). We release the DEBATE dataset publicly[3]. Potential uses for this data could be to (1) study how agents debate as a proxy to humans; (2) study the structural reasons why MAD can fail in some scenarios, as identified by Becker et al. (2025); (3) explore how prompting agents to assess prior messages critically affects the speed of consensus-building. More details on the DEBATE dataset are in Appendix D.

5 Epilogue

We proposed MALLM, a framework specialized in conversational problem-solving for MAD. MALLM enables users to configure debates for their specific problems and research objectives. More specifically, our framework supports the analysis of multiple components, including agent personas, response generators, discussion paradigms, and decision protocols.

MALLM works with most proprietary and open models and can load any text-based Hugging Face dataset for problem-solving. MALLM's evaluation pipeline offers pre-implemented metrics (e.g., Accuracy, BLEU, BERTScore) and automatic chart generation, while accounting for the statistical variance of MAD. A demo for the capabilities of MALLM is publicly available[4].

We described four potential use cases that can be further developed: the number of agents and their impact, the resilience of MAD against jailbreak attacks, finetuning specialized agents, and adaptive moderation. Future work could also expand MALLM with additional functionalities, such as evaluating debates through any Hugging Face metric. Beyond serving as a testbed for MAD itself, MALLM provides researchers with an environment to assess how variations in agents, discussion paradigms, and decision protocols can impact their specific problems.

Limitations

We provide the MALLM framework with pre-implemented variants for personas, response generators, discussion paradigms, and decision protocols. While our goal was to include diverse variants backed by literature (e.g., voting (Yang et al., 2024), consensus (Yin et al., 2023), and judge (Zheng et al., 2023) for decision protocols), we could not account for all possible patterns of agent orchestration. There is the possibility that niche use cases with MAD and future developments are not captured by our selection, e.g., decisions by confidence-weighted voting (Chen et al., 2023b). We publish our framework as open-source, allowing and encouraging researchers to develop custom components via subclassing and apply MALLM to their specific use cases.

6 Acknowledgements

This work was supported by the Landeskriminalamt NRW, the Lower Saxony Ministry of Science and Culture and the VW Foundation.

[3] huggingface.co/datasets/Multi-Agent-LLMs/DEBATE
[4] mallm.gipplab.org

References

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *ArXiv preprint*, abs/2503.05179.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Jonas Becker. 2024. Multi-Agent Large Language Models for Conversational Task-Solving. *ArXiv preprint*, abs/2410.22932.

Jonas Becker, Lars Benedikt Kaesberg, Andreas Stephan, Jan Philip Wahle, Terry Ruas, and Bela

Gipp. 2025. Stay focused: Problem drift in multi-agent debate. *ArXiv preprint*, abs/2502.19559.

Jonas Becker, Jan Philip Wahle, Bela Gipp, and Terry Ruas. 2024. Text generation: A systematic literature review of tasks, evaluation, and challenges. *ArXiv preprint*, abs/2405.15604.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better llm-based evaluators through multi-agent debate. In *International Conference on Representation Learning*, volume 2024, pages 9079–9093.

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. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 55005–55029. Curran Associates, Inc.

Huaben Chen, Wenkang Ji, Lufeng Xu, and Shiyu Zhao. 2023a. Multi-agent consensus seeking via large language models. *ArXiv preprint*, abs/2310.20151.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023b. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *ArXiv preprint*, abs/2309.13007.

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. 2021. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168.

Paul T Costa and Robert R McCrae. 1992. Normal personality assessment in clinical practice: The neo personality inventory. *Psychological assessment*, 4(1):5.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. *ArXiv preprint*, abs/2305.14325.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Zeyu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. AgentScope: A Flexible yet Robust Multi-Agent Platform. *ArXiv preprint*, abs/2402.14034.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics*, 9:346-361. Place: Cambridge, MA Publisher: MIT Press.

Lewis R Goldberg. 1999. A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. *Personality psychology in Europe*, 7(1):7–28.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. *arXiv preprint*. ArXiv: 2402.01680 [cs].

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

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. 2023. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. *ArXiv preprint*, abs/2308.00352.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. 2025. Voting or consensus? decision-making in multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11640–11671, Vienna, Austria. Association for Computational Linguistics.

Venelin Kovatchev, M. Antònia Martí, and Maria Salamó. 2018. ETPC - a paraphrase identification corpus annotated with extended paraphrase typology and negation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. 2024. Improving multi-agent debate with sparse communication topology. *ArXiv preprint*, abs/2406.11776.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jessica L. Maples, Li Guan, Nathan T. Carter, and Joshua D. Miller. 2014. A test of the international personality item pool representation of the revised neo personality inventory and development of a 120-item ipip-based measure of the five-factor model. *Psychological Assessment*, 26(4):1070–1084.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

OpenAI. 2024. openai-agents-python. `https://github.com/openai/openai-agents-python`. Accessed: 2025-05-27.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *ArXiv preprint*, abs/2311.12022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740.

Greg Serapio-García, Mustafa Safdari, Clément Crepy, Luning Sun, Stephen Fitz, Peter Romero, Marwa Abdulhai, Aleksandra Faust, and Maja Matarić. 2023. Personality traits in large language models. *ArXiv preprint*, abs/2307.00184.

Andries Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D. Barrett, and Arnu Pretorius. 2023. Should we be going mad? a look at multi-agent debate strategies for llms. *ArXiv preprint*, abs/2311.17371.

Aleksandra Sorokovikova, Sharwin Rezagholi, Natalia Fedorova, and Ivan P Yamshchikov. 2024. Llms simulate big5 personality traits: Further evidence. In *Proceedings of the 1st Workshop on Personalization of Generative AI Systems (PERSONALIZE 2024)*, pages 83–87.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *Preprint*, arXiv:2310.16049.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *ArXiv preprint*, abs/2306.03314.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *ArXiv preprint*, abs/2408.02442.

Arne Tillmann. 2025. Literature review of multi-agent debate for problem-solving. *ArXiv preprint*, abs/2506.00066.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *ArXiv preprint*, abs/2501.06322.

J. Wahle, T. Ruas, S. M. Mohammad, N. Meuschke, and B. Gipp. 2023. Ai usage cards: Responsibly reporting ai-generated content. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 282–284, Los Alamitos, CA, USA. IEEE Computer Society.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024a. Soft self-consistency improves language models agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 287–301, Bangkok, Thailand. Association for Computational Linguistics.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024b. Rethinking the Bounds of LLM Reasoning: Are Multi-Agent Discussions the Key? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024c. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. *ArXiv preprint*, abs/2406.01574.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing Cognitive Synergy in Large Language Models: A

Task-Solving Agent through Multi-Persona Self-Collaboration. *arXiv preprint*. ArXiv: 2307.05300 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint*. ArXiv: 2201.11903 [cs].

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. *ArXiv preprint*, abs/2308.08155.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. ExpertPrompting: Instructing Large Language Models to be Distinguished Experts. *arXiv preprint*. ArXiv: 2305.14688 [cs].

Joshua C. Yang, Damian Dailisan, Marcin Korecki, Carina I. Hausladen, and Dirk Helbing. 2024. Llm voting: Human choices and ai collective decision-making. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 7(1):1696–1708.

Yongjin Yang, Euiin Yi, Jongwoo Ko, Kimin Lee, Zhijing Jin, and SeYoung Yun. 2025. Revisiting multi-agent debate as test-time scaling: A systematic study of conditional effectiveness. *ArXiv preprint*, abs/2505.22960.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-Thought: Enhancing Large Language Model Capabilities through Cross-Model Communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15135–15153, Singapore. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Language Agents as Optimizable Graphs. *ArXiv preprint*, abs/2402.16823.

Appendix

A MALLM Workflow

We provide an example of how MALLM could be used to conduct experiments on MAD in Figure 4. An example discussion can be seen in Figure 5.

B System Demonstration

We provide an interactive demonstration website for MALLM. It is available under mallm.gipplab.org. A screenshot can be seen in Figure 6.

C Comparison With Other Frameworks

We compare the functionality of other commonly used frameworks for MAD with MALLM. The comparison can be seen in Table 5.

D DEBATE Examples

To demonstrate MALLM's capabilities, we construct a set of examples called DEBATE (Diverse Exchanges Between Autonomous Talking Entities), comprising 14,400 strategic problem-solving debates based on the StrategyQA dataset (Geva et al., 2021), generated using 144 distinct MALLM configurations. Each configuration combines specific settings of the framework's modular components listed in Table 4.

Parameter	Values
Response Generators	Simple, Critical, Reasoning
Persona Generators	None, Expert, IPIP
Discussion Paradigms	Memory, Relay, Report, Debate
Decision Protocols	Majority Consensus, Unanimity Consensus, Simple Voting, Approval Voting

Table 4: Parameters used for creating DEBATE. The example set comprises diverse data for each possible combination of parameters.

For example, one setup uses the `Simple` response generator, no personas, the `Memory` discussion paradigm, and the `Majority Consensus` decision protocol. Our rationale for selecting the parameters is to ensure diversity in the agent orchestration (e.g., two voting and two consensus approaches for decision-making), while keeping the computational effort manageable. All debates involve three agents, up to seven turns, and use the meta-llama/Llama-3.3-70B-Instruct

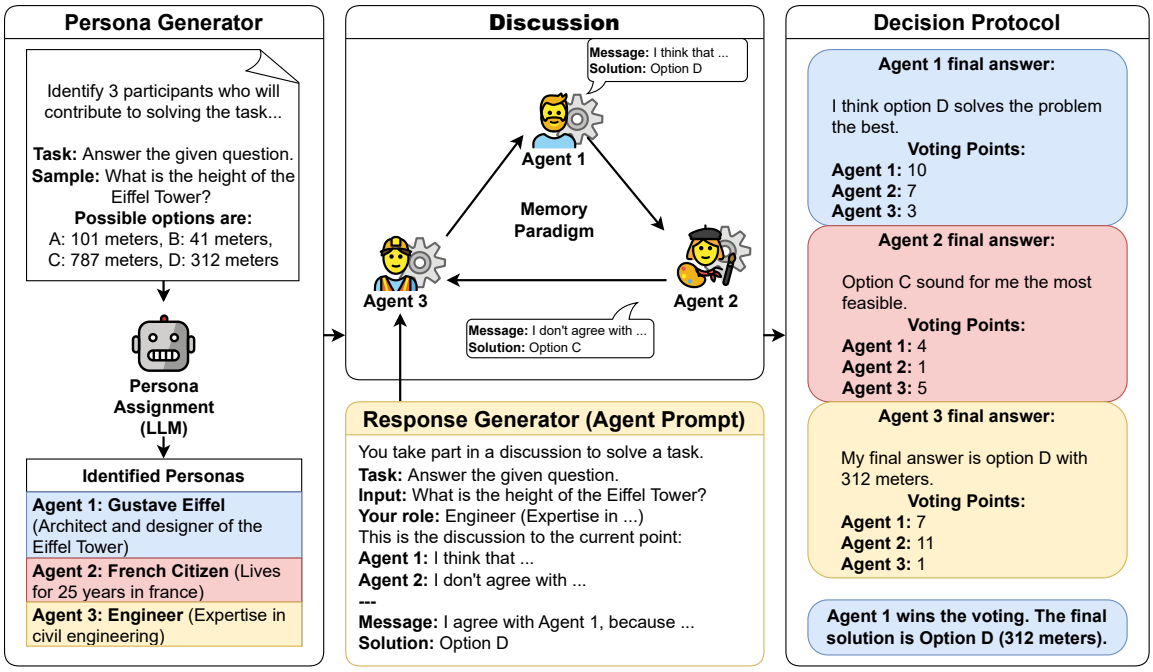

Figure 5: Example multi-agent discussion conducted in the MALLM framework. It showcases the functionality of the four modules and how they work together to get an improved final solution. First, we use the `Expert` persona generator to create three agents with different expertise. These agents discuss according to the `Memory` discussion paradigm and use the `Simple` response generator to formulate their answers. After the third turn, they begin voting using the `Cumulative Voting` decision protocol until they reach a final solution.

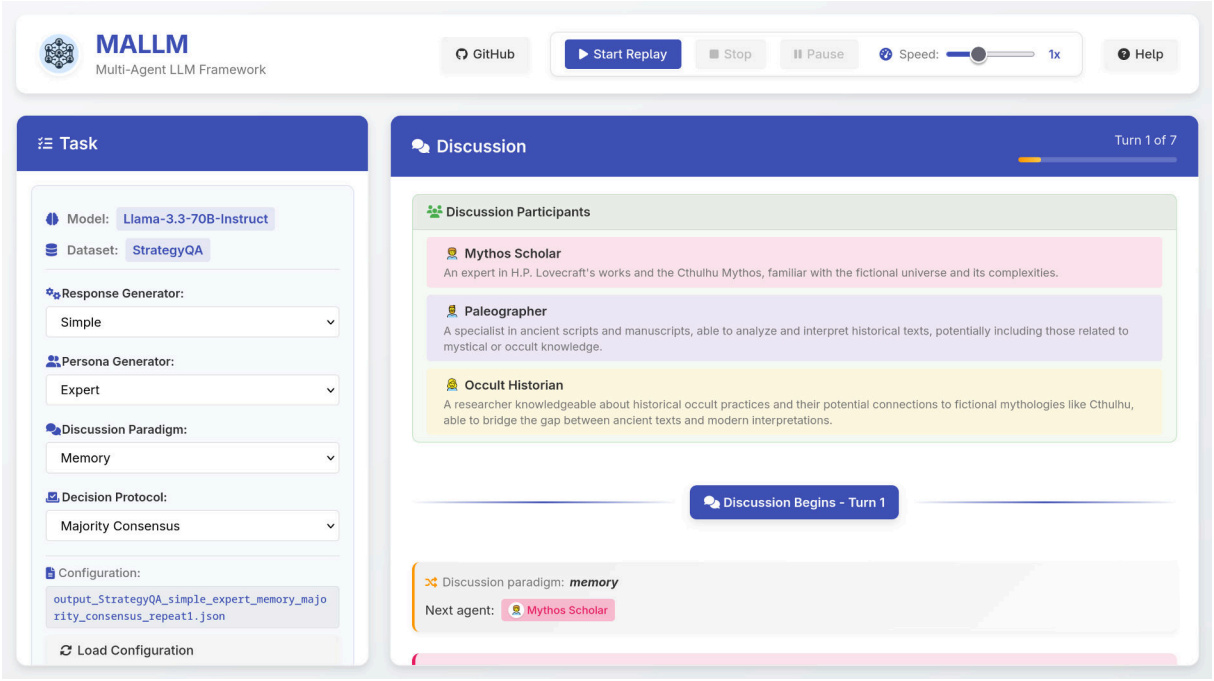

Figure 6: A screenshot of the demonstration website. One of 144 configurations for MAD can be selected on the left panel. MAD is conducted to solve the task, visible on the right panel. The top header provides functions to pause the replay or adjust the simulation speed.

model. Each of the 144 configurations runs 100 debates. For the creation of the DEBATE examples, we utilize eight NVIDIA A100 GPUs, each equipped with 40GB of VRAM, to host a meta-llama/Llama-3.3-70B-Instruct model for 8 days, 5 hours, and 42 minutes. The data is available on Hugging Face[5].

E Parameters

We adhere to default parameters for the models we used, using langchain 0.1.16 and openai 1.25.0 for the implementation of the MALLM framework.

- `temperature = 1.0`
- `top_p = 1.0`
- `presence_penalty = 0.0`
- `frequency_penalty = 0.0`
- `max_tokens = 1024`

E.1 Agent Experiments

The setup and parameters for this experiment are described in Kaesberg et al. (2025). They use *meta-llama/Meta-Llama-3-8B-Instruct* as a model for all agents with the following fixed parameters:

- Persona generator: `Expert`
- Discussion paradigm: `Memory`
- Decision protocol: Average of `Simple Voting`, `Ranked Voting`, `Cumulative Voting` and `Approval Voting`

Each experiment is repeated three times, and the average performance and standard deviation across the runs are reported.

E.2 Discussion Experiments

We use *meta-llama/Meta-Llama-3-70B-Instruct* as a model for all agents. We further report the parameters that are set fixed for this experiment:

- Persona generator: `Expert`
- Response generator: `Simple`
- Decision protocol: `Majority Consensus`

To ensure the reliability of our findings, we follow the prior work of Wang et al. (2024a) and conduct each experiment five times, reporting the average performance and standard deviation across the runs.

E.3 Decision Experiments

The setup and parameters for this experiment are described in Kaesberg et al. (2025). They use *meta-llama/Meta-Llama-3-8B-Instruct* as a model for all agents with the following fixed parameters:

[5] huggingface.co/datasets/Multi-Agent-LLMs/DEBATE

Customizable Feature	Agent Personas	Agent Responses	Discussion Paradigms	Decision Protocol	Evaluation Pipeline
AutoGen (Wu et al., 2023)	✗	✗	✗	✗	✗
GPTSwarm (Zhuge et al., 2024)	✗	✗	✓	✓	✗
OpenAI Agents SDK (OpenAI, 2024)	✓	✗	✓	✗	✓
MetaGPT (Hong et al., 2023)	✓	✓	✗	✗	✗
AgentScope (Gao et al., 2024)	✓	✗	✓	✗	✗
AutoGPT (Talebirad and Nadiri, 2023)	✓	✓	✓	✗	✗
MALLM (this work)	✓	✓	✓	✓	✓

Table 5: Comparison of customizable features across commonly used frameworks for MAD. MALLM enables the modification of agent personas, agent responses, discussion paradigms, and decision protocols. It also comes with an integrated evaluation pipeline. To the best of our knowledge, no other framework offers the same level of configurability for these main components of MAD.

- Persona generator: `Expert`
- Discussion paradigm: `Memory`
- Response Generator: `Simple`

Each experiment is repeated three times, and the average performance and standard deviation across the runs are reported.

F Evaluation Pipeline

Figures 7 to 10 show example charts generated by the MALLM evaluation pipeline. They are presented to demonstrate the pipeline's automated analysis and visualization capabilities. These specific examples are generated from experiments on the StrategyQA dataset using the *meta-llama/Meta-Llama-3-8B-Instruct* model, with error bars representing the standard deviation across three runs.

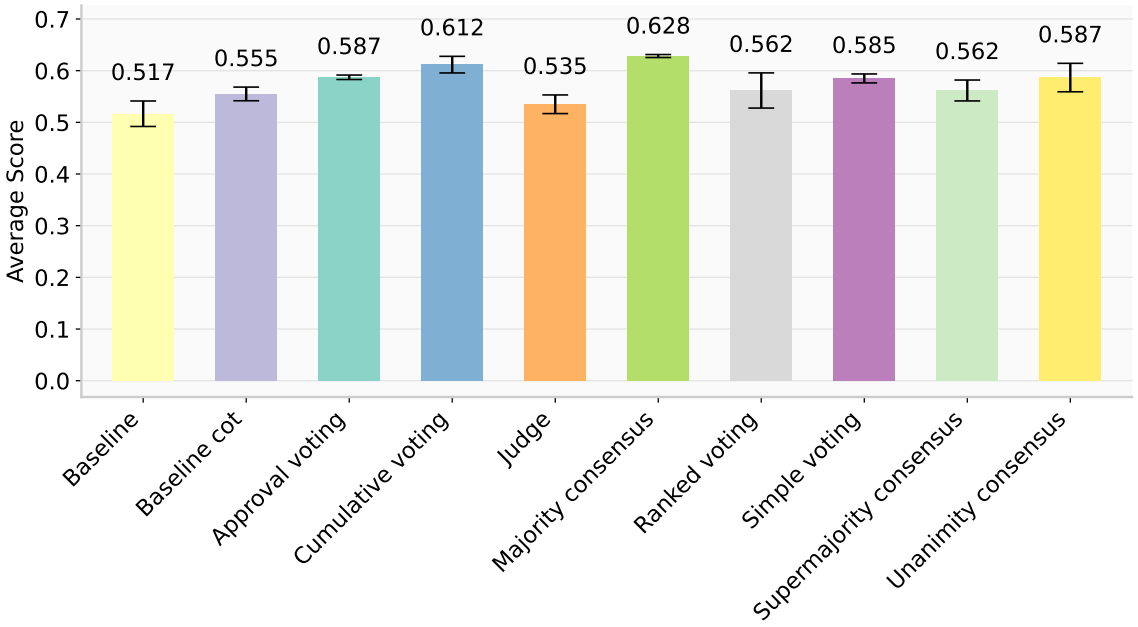

Figure 7: An example chart automatically generated by the MALLM evaluation pipeline, comparing the average performance scores of various decision protocols on the StrategyQA dataset. Error bars indicate the standard deviation over three experimental runs.

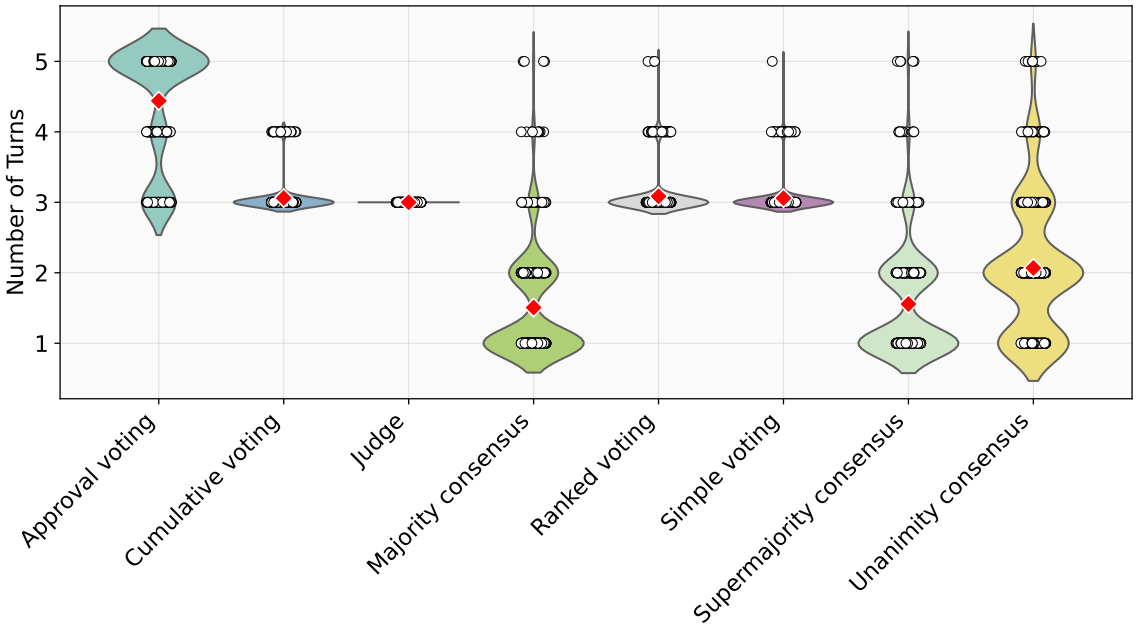

Figure 8: An example visualization from the MALLM evaluation pipeline, showing the distribution of the number of turns required for different decision protocols to converge on the StrategyQA dataset. The plot's width illustrates the frequency of turn counts, and the red marker shows the mean.

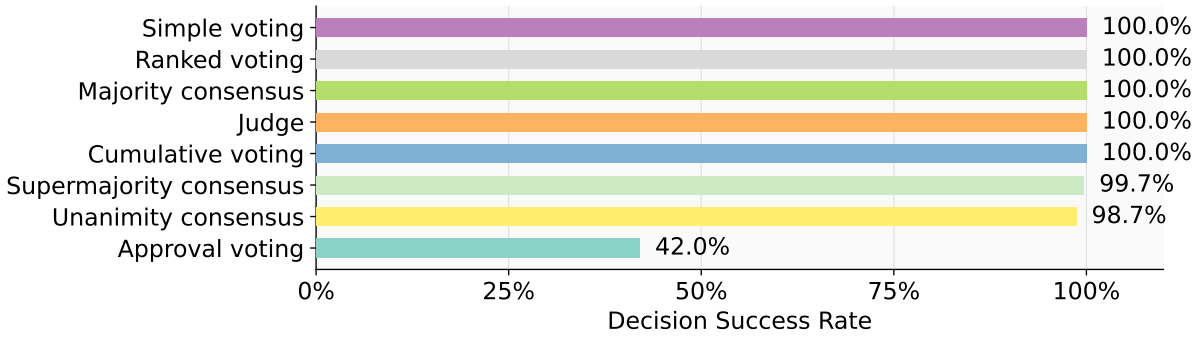

Figure 9: An example chart from the MALLM evaluation pipeline showing the decision success rates for each protocol on the StrategyQA dataset. The decision success rate explains how many of the debates reach a final solution according to the decision protocol (e.g., $> 50\%$ for `Majority Consensus`).

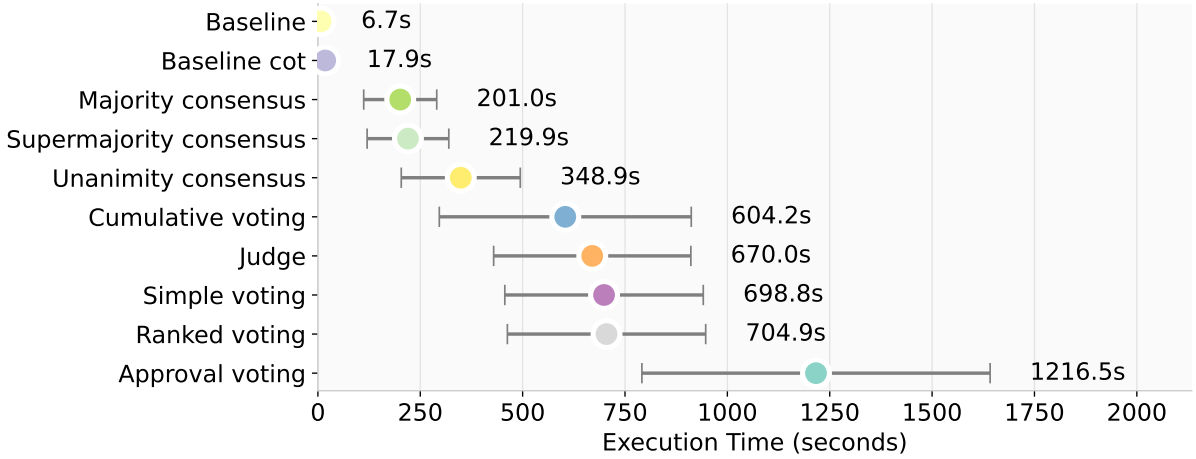

Figure 10: An example of an automatically generated chart from the MALLM evaluation pipeline, comparing the average wall clock time (in seconds) for each decision protocol on the StrategyQA dataset.

G Experiment Prompts

We provide the prompts that the MALLM framework uses to conduct MAD, which are relevant for our experiments. Appendix G.1 shows the prompt used across all configurations. Prompts specific to the persona experiments are provided in Appendix G.2, while prompts for the response generators are detailed in Appendix G.3. Additionally, prompts related to the discussion paradigm are included in Appendix G.4, and prompts for the various decision protocols are available in Appendix G.5.

G.1 General Debate

System Prompt:

You take part in a discussion to solve a task.

Your role: `<persona name>` (`<persona description>`)
Task: `<instruction>`
Input: `<example>`
Context: `<optional information>`
Current Solution: `<most recent draft>`
Discussion so far: `<agent memory>`

Figure 11: Prompt used with the `Simple` response generator for an agent participating in collaborative debate. If this is the first message of the discussion, we write "Nobody proposed a solution yet. Please provide the first one." instead of the most recent draft and agent memory.

G.2 Persona Experiments

These are the prompts used for the persona experiments.

System Prompt:

Solve the following task: `<task instruction>`
Input: `<input str>`
Make absolutely sure to provide your solution in the end: 'FINAL SOLUTION: <Letter>'.

User Prompt:

Answer the following question.

Figure 12: Base prompt used for agents participating in a GPQA experiment.

System Prompt:

When faced with a task, begin by identifying the participants who will contribute to solving the task. Provide role and description of the participants, describing their expertise or needs, formatted using the provided JSON schema.
Generate one participant at a time, complementing the existing participants to foster a rich discussion.
Example 1: `<example 1>`
Example 2: `<example 2>`
Example 3: `<example 3>`

User Prompt:

Now generate a participant to discuss the following task:
Task: `<task description>`. Please use the following examples to generate a useful persona for the task! Only answer with the JSON for the next persona.

Figure 13: Prompt used for the Expert agent generator, which creates unique personas for each example.

System Prompt:

When faced with a task, begin by identifying the participants who will contribute to solving the task. Provide role and fixed characteristics of the participant, formatted using the provided JSON schema. Generate one participant at a time, complementing the existing participants to foster a rich discussion.
You must choose the following characteristics for the participant, in JSON format:
`<characteristics and options>`
You absolutely must stick to the JSON format and the characteristics and options provided.
Example 1: `<example 1>`
Example 2: `<example 2>`

User Prompt:

Now generate a participant to discuss the following task:
Task: `<task description>`. Only answer with the JSON for the next persona! Ensure your new participant is unique.

Figure 14: Prompt used for the IPIP agent generator, which creates unique personas for each example.

G.3 Response Generator Experiments

We provide the prompts for each response generator used.

G.3.1 Simple Response Generator

User Prompt:

Based on the provided feedback, carefully re-examine your previous solution. Provide a revised solution.

Figure 15: Prompt with the `Simple` response generator instructing an agent to create a new solution based on received feedback. It is appended to the system prompt in Figure 11.

User Prompt:

Improve the current solution.
If you agree with the current solution, answer with `[AGREE]`.
Else, answer with `[DISAGREE]`, explain why, and provide an improved solution.
Let's think step-by-step.

Figure 16: Prompt with the `Simple` response generator to an agent for contributing to the current solution draft. The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

User Prompt:

Improve the current solution.
Based on the current solution, give constructive feedback. If you agree, answer with `[AGREE]`, else answer with `[DISAGREE]` and explain why.
Let's think step-by-step.

Figure 17: Prompt with the `Simple` response generator to an agent for giving feedback to the current solution draft (without directly proposing a new solution). The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

G.3.2 Critical Response Generator

User Prompt:

Re-examine the current solution critically based on the feedback provided. Ensure your revision addresses any identified weaknesses or areas for improvement. Submit a revised and improved solution.

Figure 18: Prompt with the `Critical` response generator instructing an agent to create a new solution based on received feedback. It is appended to the system prompt in Figure 11.

User Prompt:
Improve the current solution. Identify specific areas that need enhancement and propose unique solutions based on your persona. If you see no room for improvement, answer with `[AGREE]`, otherwise, answer with `[DISAGREE]` and provide a clear, solution.

Figure 19: Prompt with the `Critical` response generator to an agent for contributing to the current solution draft. The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

User Prompt:
Critically evaluate the current solution. Identify potential weaknesses or areas of improvement. If you believe the solution is flawless, answer with `[AGREE]`, otherwise answer with `[DISAGREE]` and provide constructive feedback with suggestions for improvement.

Figure 20: Prompt with the `Critical` response generator to an agent for giving feedback to the current solution draft (without directly proposing a new solution). The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

G.3.3 Reasoning Response Generator

User Prompt:
Based on the provided feedback, carefully re-examine your previous solution. Provide a revised solution.

Figure 21: Prompt with the `Reasoning` response generator instructing an agent to create a new solution based on received feedback. It is appended to the system prompt in Figure 11.

User Prompt:
Improve the current steps of the argument by referring to the other participants in the discussion. Be critical and answer short and concise. Repeat only the reasoning steps that you think are the most important. If you think there is enough information to create a final answer also answer with `[AGREE]` else answer with `[DISAGREE]`. Don't provide a final solution yet.

Figure 22: Prompt with the `Reasoning` response generator to an agent for contributing to the current solution draft. The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

User Prompt:
Based on the current solution, give constructive feedback. If you agree, answer with `[AGREE]`, else answer with `[DISAGREE]` and explain why.

Figure 23: Prompt with the `Reasoning` response generator to an agent for giving feedback to the current solution draft (without directly proposing a new solution). The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

G.4 Discussion Paradigm Experiments

For the experiments on discussion paradigms, just one more prompt for the Chain-of-Thought baseline is used. We use the general prompts described in Appendix G.1 for MAD.

System Prompt:
Solve the provided task. Do not ask back questions. Clearly indicate your final solution after the text 'Final Solution:'.

Task: `<task instruction>`
Input: `<input str>`

User Prompt:
Let's think step-by-step.

Figure 24: Prompt used for the Chain-of-Thought baseline.

G.5 Decision Protocol Experiments

These are all prompts used for the decision-making protocols. The final answer extraction prompt can be seen in Figure 25. The prompt for the voting-based decision protocols can be seen in Figure 26 (Simple Voting), Figure 27 (Approval Voting), Figure 29 (Ranked Voting), and Figure 28 (Cumulative Voting) decision protocols (Figure 26 to Figure 30). The consensus decision protocol has no special prompt, as it terminates when a consensus is found, and then the final answer extraction prompt is used. Voting also utilizes the final answer extraction prompt to obtain the final answer from each agent that is used during the voting process.

G.5.1 Final Answer Extraction

System Prompt:

Your role: `<persona>` (`<persona description>`)

User Prompt:

You are tasked with creating a final solution based on the given input and your previous response.
Task: `<task>`
Input: `<input sample>`
Your previous response: `<previous answer>`
Extract the final solution to the task from the provided text. Remove statements of agreement, disagreement, and explanations. Do not modify the text. Do not output any text besides the solution. If there is no solution provided, just copy the previous response.

Figure 25: Prompt used to extract the final answer of a given agent from its previous response.

G.5.2 Voting Prompts

System Prompt:

Your role: `<persona>` (`<persona description>`)

User Prompt:

You are tasked with voting for the best solution from the list provided below based on the given task.
Task: `<task>`
Question: `<input sample>`
Here are the possible solutions:
Solution 1: `<agent 1 final answer>`
Solution 2: `<agent 2 final answer>`
Solution 3: `<agent 3 final answer>`
Based on the above solutions, please provide the number of the solution you are voting for. Answer only with the number.

Figure 26: Prompt used to get a vote from each agent for the Simple Voting decision protocol.

System Prompt:

Your role: `<persona>` (`<persona description>`)

User Prompt:

You are tasked with approving any number of solutions from the list provided below based on the given task.
Task: `<task>`
Question: `<input sample>`
Here are the possible solutions:
Solution 1: `<agent 1 final answer>`
Solution 2: `<agent 2 final answer>`
Solution 3: `<agent 3 final answer>`
Based on the above solutions, please provide the numbers of the solutions you are approving, separated by commas. Answer only with the numbers.

Figure 27: Prompt used to get a vote from each agent for the Approval Voting decision protocol.

System Prompt:

Your role: <persona> (<persona description>)

User Prompt:

You are tasked with distributing 10 points among the provided solutions based on the given task.
Task: <task>
Question: <input sample>
Here are the possible solutions:
Solution 1: <agent 1 final answer>
Solution 2: <agent 2 final answer>
Solution 3: <agent 3 final answer>
Based on the above solutions, please distribute 10 points among the solutions. Provide your points allocation as a JSON dictionary where keys are solution numbers (as int) and values are the points. The total points should sum up to 10. Answer only with the JSON dictionary.

Figure 28: Prompt used to get a vote from each agent for the Cumulative Voting decision protocol.

System Prompt:

Your role: <persona> (<persona description>)

User Prompt:

You are tasked with ranking the solutions from the most preferred to the least preferred based on the given task.
Task: <task>
Question: <input sample>
Here are the possible solutions:
Solution 1: <agent 1 final answer>
Solution 2: <agent 2 final answer>
Solution 3: <agent 3 final answer>
Based on the above solutions, please provide the rankings of the solutions separated by spaces. Example: '0 2 1' if you prefer Solution 0 the most, then Solution 2, and finally Solution 1. Provide up to 5 rankings. Only answer with the rankings.

Figure 29: Prompt used to get a vote from each agent for the Ranked Voting decision protocol.

G.5.3 Judge Prompt

User Prompt:

Task: <task>
Question: <input sample>
Please provide a decision on the following solutions and combine them in a single answer to solve the task. Only answer with the solution:
Solution 1: <agent 1 final answer>
Solution 2: <agent 2 final answer>
Solution 3: <agent 3 final answer>

Figure 30: Prompt used to get a final decision from the Judge decision protocol. No alterations are applied.

H MALLM Configuration File

Example MALLM batch configuration file for running an experiment with fixed response generator, persona generator, and discussion paradigm, but varying decision protocols. The "repeats" field defines how many times each run is repeated, which is later relevant for evaluating for robustness by the standard deviation between experiment runs. The "common" field describes parameters that are considered for all runs. The "runs" field defines the parameters unique for each individual run.

```
{
  "repeats": 3,
  "name": "<DATASET NAME>",
  "common": {
    "task_instruction_prompt_template": "<DATASET NAME>",
    "endpoint_url": "<LLM API HOSTNAME>",
    "api_key": "<LLM API KEY>",
    "model_name": "<MODEL NAME>",
    "input_json_file_path": "data/datasets/<DATASET NAME>.json",
    "concurrent_api_requests": 200,
    "num_samples": "<NUMBER OF SAMPLES>",
    "max_turns": 5,
    "response_generator":"simple"
  },
  "runs": [
    {
      "output_json_file_path": "results/baseline-cot.json",
      "use_baseline": true
    },
    {
      "output_json_file_path": "results/baseline.json",
      "use_baseline": true,
      "use_chain_of_thought": false
    },
    {
      "output_json_file_path": "results/approval.json",
      "decision_protocol": "approval_voting"
    },
    {
      "output_json_file_path": "results/cumulative.json",
      "decision_protocol": "cumulative_voting"
    },
    {
      "output_json_file_path": "results/majority_consensus.json",
      "decision_protocol": "majority_consensus"
    },
    {
      "output_json_file_path": "results/supermajority_consensus.json",
      "decision_protocol": "supermajority_consensus"
    },
    {
      "output_json_file_path": "results/unanimity_consensus.json",
      "decision_protocol": "unanimity_consensus"
    },
    {
      "output_json_file_path": "results/voting.json",
      "decision_protocol": "simple_voting"
    },
    {
      "output_json_file_path": "results/ranked.json",
      "decision_protocol": "ranked_voting"
    }
  ]
}
```

I AI-Usage Card

AI Usage Card

PROJECT DETAILS	PROJECT NAME MALLM: Multi-Agent Large Language Models Framework	DOMAIN Paper	KEY APPLICATION Multi-Agent Debate
CONTACT(S)	NAME(S) Jonas Becker	EMAIL(S) jonas.becker@uni-goettingen.de	AFFILIATION(S) University of Göttingen, LKA NRW
MODEL(S)	MODEL NAME(S) ChatGPT o3, 4.5, 4o, 5		
LITERATURE REVIEW	FINDING LITERATURE	FINDING EXAMPLES FROM KNOWN LITERATURE OR ADDING LITERATURE FOR EXISTING STATEMENTS	COMPARING LITERATURE
WRITING	GENERATING NEW TEXT BASED ON INSTRUCTIONS	ASSISTING IN IMPROVING OWN CONTENT OR PARAPHRASING RELATED WORK ChatGPT 4.5, 5	PUTTING OTHER WORKS IN PERSPECTIVE
CODING	GENERATING NEW CODE BASED ON DESCRIPTIONS OR EXISTING CODE ChatGPT o3, 4o	REFACTORING AND OPTIMIZING EXISTING CODE ChatGPT o3, 4o	COMPARING ASPECTS OF EXISTING CODE
ETHICS	WHY DID WE USE AI FOR THIS PROJECT? Efficiency, Speed, Knowledge	WHAT STEPS ARE WE TAKING TO MITIGATE ERRORS OF AI?	WHAT STEPS ARE WE TAKING TO MINIMIZE THE CHANCE OF HARM OR INAPPROPRIATE USE OF AI?

THE CORRESPONDING AUTHORS VERIFY AND AGREE WITH THE MODIFICATIONS OR GENERATIONS OF THEIR USED AI-GENERATED CONTENT

AI Usage Card v2.0 https://ai-cards.org (Wahle et al., 2023)