Preventing Rogue Agents Improves Multi-Agent Collaboration

Ohav Barbi **Ori Yoran** **Mor Geva**
Blavatnik School of Computer Science and AI, Tel Aviv University
`{ohavbarbi@mail,ori.yoran.cs,morgeva@tauex}.tau.ac.il`

Abstract

Multi-agent systems, where specialized agents collaborate to solve a shared task hold great potential, from increased modularity to simulating complex environments. However, they also have a major caveat – a single agent can cause the entire system to fail. Consider a simple game where the knowledge to solve the task is distributed between agents, which share information in a communication channel. At each round, any of the agents can terminate the game and make the final prediction, even if they are uncertain about the outcome of their action. Detection of such rogue agents *before they act* may prevent the system's failure. In this work, we propose to *monitor* agents during action prediction and *intervene* when a future error is likely to occur. To test our approach, we introduce `WhoDunitEnv`, a multi-agent collaboration environment that allows modular control over task complexity and communication structure. Experiments on `WhoDunitEnv`, code generation tasks and the `GovSim` environment for resource sustainability show that our approach leads to substantial performance gains up to 17.4%, 2.5% and 20%, respectively. Thorough analysis shows that our monitors successfully identify critical points of agent confusion and our interventions effectively stop agent errors from propagating. We release `WhoDunitEnv` and our code for future studies on multi-agent collaboration at `https://github.com/Ohav/rogue-agents`.

1 Introduction

Recently there has been a growing interest in leveraging large language models (LLMs) to solve complex multi-agent tasks (Guo et al., 2024a; Tran et al., 2025), such as improving factuality and reasoning, (Du et al., 2024; Liang et al., 2024) simulating software development teams (Hong et al., 2024; Qian et al., 2024; Liu et al., 2024c), and human interactions (Park et al., 2023). A key requirement for succeeding at such cooperative tasks is the agent's ability to communicate effectively.

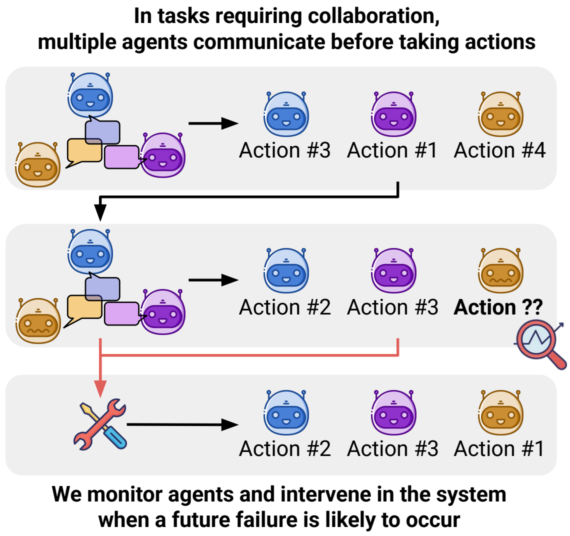

Figure 1: **An overview of our approach**. We propose to improve multi-agent collaboration by monitoring agent communication and applying interventions to the environment in case rogue agents are detected.

However, establishing effective communication is a major challenge for current LLMs, which often ignore critical information in the communication (Liu et al., 2024b; Levy et al., 2024), become distracted by irrelevant messages (Shi et al., 2023; Amayuelas et al., 2024), and introduce hallucinations (Xiao and Wang, 2021; Zhang et al., 2023; Jiang et al., 2024; Lin et al., 2024; Huang et al., 2025). Such failures introduce noise to the communication, which can be amplified as LLM responses are generated for many steps (Zhang et al., 2024a; Ivgi et al., 2024; Yoran et al., 2024; Jimenez et al., 2024). A single valid agent can be consequently affected by these issues and go *rogue*. This *rogue agent* could then cause the entire system to fail.

To improve multi-agent collaboration, recent works have proposed introducing changes to the communication protocol (Guo et al., 2024b; Mehta et al., 2024), augmenting the communication with

Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), pages 486–511
July 31, 2025

agent beliefs (Li et al., 2023b), and adding specific modules for reasoning and grounding (Agashe et al., 2023; Hong et al., 2024). While these approaches often yield improvements in the overall task performance, they do not prevent cases where one *rogue agent* introduces noise to the communication and drags the system into failure (D'Arcy et al., 2024; Hong et al., 2024).

In this work, we propose to improve multi-agent collaboration through live monitoring and interventions in agent communication. Specifically, we hypothesize that many failures can be detected and prevented early on — before the agent becomes non-functional and the task fails — based on the state of the agent and how it observes the environment (see Fig. 1 for an illustration). This approach is widely used in industrial systems, such as intrusion detection software and critical manufacturing processes (Stavropoulos et al., 2013; Baldauf et al., 2021; Olajiga et al., 2024; Woods and Seymour, 2023; Salem et al., 2024), and is also observed in natural systems such as the human immune system, which monitors faulty behavior and prevents infections from spreading throughout the body.

To test this hypothesis, we present a monitor and intervention framework (§2). To monitor an agent's state, we employ simple classifiers that detect rogue agents based on intrinsic signals of uncertainty, such as the prediction's entropy during action selection. This monitor then triggers an intervention mechanism, which prevents the communication from being distorted and the agents from failing at the task. Specifically, we experiment with simple interventions that roll back certain parts of the communication or the entire environment.

To evaluate our approach, we create a modular collaborative environment, called `WhoDunitEnv`, which allows analyzing different agent communication protocols at varying task-complexity levels (§3). `WhoDunitEnv` is inspired by the well-known game of *Guess Who*,[1] where players (here agents) collaborate to identify a culprit among a group of suspects based on their appearance properties.

We conduct experiments (§4) on two variants of `WhoDunitEnv`, a code generation environment `CodeGen`, and the `GovSim` resource sharing environment (Piatti et al., 2024). These environments cover symmetric and asymmetric agents, structured and free-form communication protocols, and varying levels of task complexities.

Our results (§5) show that, across all settings, monitoring rogue agents leads to substantial improvements in multi-agent collaboration. Specifically, we observe gains of up to 20%. Finally, we conduct ablations and qualitative analysis (§6), showing that strong monitors and interventions are needed to improve performance, and that our monitors effectively identify a range of *rogue* agent behaviors, including hallucination of information in the communication channel.

To summarize, our work makes the following contributions: (a) we propose the notion of live monitoring and interventions to prevent failures due to rogue agents in multi-agent systems, (b) we introduce `WhoDunitEnv`, a modular environment for studying collaborative multi-agent systems where agents are either symmetrical/asymmetrical, (c) we show that our approach leads to substantial performance gains on `WhoDunitEnv`, `CodeGen`, and `GovSim` across models and complexity levels, and (d) we show that improvements are due to strong monitors and interventions.

2 Monitoring and Intervening in Multi-Agent Systems

In multi-agent systems, agents collaborate in order to solve a task or use shared resources. The system comprises of agents $G = \{g_i\}$, a communication channel C that stores messages from the agents, and a shared task T. Agents possess knowledge K_i and perform actions A_i. For example, agents that simulate human behavior have different memories based on their experiences (Park et al., 2023), and different personas in a software development team have different actions, such as designing the code, programming, or writing tests (Qian et al., 2024).

Let C_j be the communication channel at the j-th turn, an agent g_i chooses their next action using a probability distribution over actions conditioned on their knowledge and the shared information:

$$P_{a \in A_i}(a, j, g_i, T) = P(\texttt{Action} = a \mid K_i, C_j, T)$$

Similarly, agents share information in the channel with a distribution over knowledge pieces:

$$P_{k \in K_i}(k, j, g_i, T) = P(\texttt{Share} = k \mid K_i, C_{j-1}, T)$$

We propose to perform *live*, mid-run interventions to prevent single agents from causing a system-wide failure (Fig. 1). Our approach consists of monitoring agent action predictions to detect

[1]`https://en.wikipedia.org/wiki/Guess_Who`

rogue agents, and intervening in the environment when a rogue agent is detected.

We view *monitoring* as a function that estimates the probability of succeeding at the task at every turn, based on the agent's probability distribution over actions. Namely, given P_{A_i} for agent g_i at turn j, we wish to estimate $P(\texttt{success} \mid P_{A_i}, j, g_i, T)$.

If a future failure is likely, namely, $P(\texttt{success} \mid P_{A_i}, j, g_i, T) < \tau$ for some threshold τ, we intervene to provide agents with an opportunity to reach a better state. An *intervention* is a causal operation that modifies the current state of the environment based on its current state and the monitoring output. For example, the intervention could revert the communication or augment it with additional content.

Monitoring agent uncertainty to predict failures Inspired by prior work on agents in Reinforcement Learning and NLP environments (Acharya et al., 2022; Liu et al., 2024a; Zhang et al., 2024b; Yoffe et al., 2024), we predict task success based on agent uncertainty. Namely, if the agent is "confused" in their action selection, they are likely to introduce noise which could fail the whole system.

Let $\mathbf{p}_i$ be the output probability distribution vector at position i.[2] We consider all the positions in the agent's generation that hold important information for its action selection. These include both the final action selection and the preceding thoughts, and differs between different environments (exact definition for each available in section §4). To estimate the agent's uncertainty, we compute the entropy, varentropy, and kurtosis of $\mathbf{p}_i$ (see exact definitions in §A) and take their maximum values over all selected positions, in addition to the current turn count, as features. We use the features to fit a simple polynomial ridge classifier with the goal of estimating the success probability $f : \mathbb{R}^m \to [0, 1]$, where $m \leq 4$ is the number of features used. This is done using boolean labels from training games at each intermediate game turn. During test time, these features are collected at every turn and are fed into the monitor, which outputs the probability of success. Further details are in §4.

Live interventions to prevent system failures When the monitor estimate is lower than τ, the monitor triggers an intervention. Notably, environments can have both *reversible* and *irreversible*. *Irreversible* actions include using a shared resource or committing to a solution, while *reversible* actions consist of only sharing information. When intervening, we undo *reversible* actions until the last *irreversible* action, to preserve the realism of our method and provide agents with another opportunity to collaborate.

Next, we introduce a multi-agent collaboration environment and evaluate our approach.

3 `WhoDunitEnv`: An Environment for Multi-Agent Collaboration

`WhoDunitEnv` is a modular multi-agent environment, where agents act as detectives working together to point out a culprit out of a suspect lineup. A game is comprised of N suspects, each with a unique set of attribute-value pairs that are randomly assigned from a predefined set. Attributes include clothing (e.g., a green shirt), accessories (e.g., a silver watch), and personality traits (e.g., mood). One suspect is randomly chosen as the culprit. Each agent receives partial information K_i about the suspects or the culprit, and must collaborate to find the culprit and accuse them. Turns move in a round robin fashion, and the game ends once either an agent accuses a suspect or a turn limit is reached. Actions are tuples (a, t), consisting of a prime action $a \in A$ and a target t to which a is applied. We provide two variants of the environment, asymmetric and symmetric, that differ by the action set A and information available to each agent K_i. See §F for prompts and §G for an example.

`WhoDunitEnv-Asym` (Fig. 2) This variant consists of exactly two agents – Accuser and Intel. $K_{accuser}$ contains the exact description of the culprit, but does not contain any information about the suspects. K_{intel} is the complete description of all suspects, without any indication of the culprit. The set of actions available to Accuser is $A_{accuser} = \{\texttt{request-specific}, \texttt{request-broad}, \texttt{accuse}\}$, which allows it to request information about a specific attribute of a suspect, request broad information from Intel for no specific suspect or attribute, and accuse a suspect, respectively. The set of actions available to Intel is $A_{intel} = \{\texttt{respond}, \texttt{respond-broad}\}$ which allows it to respond to Accuser's request with a yes/no answer or return a broad message. When returning a broad message, the agent decides on an specific attribute value, such as "green hat", then lists all the suspects that have this property. Thus,

[2] For proprietary models, where we cannot access to the full probability distribution, we approximate $\mathbf{p}_r$ with the top k tokens, setting $k = 10$.

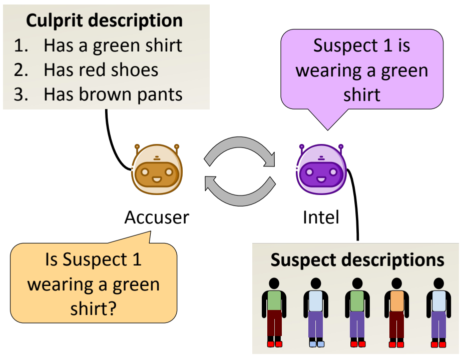

Figure 2: **An illustration of `WhoDunitEnv`-Asym** Accuser and Intel collaborate to identify the culprit from a lineup of suspects. Accuser, knowing the culprit's identity, can query and accuse. Intel chooses what and how much information to provide about the suspects.

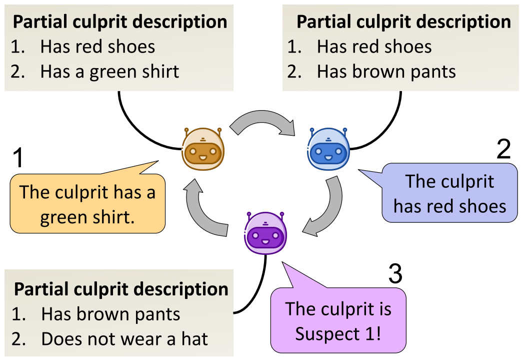

Figure 3: **An illustration of `WhoDunitEnv`-Sym.** Agents are tasked with identifying the culprit among a lineup of suspects by sharing information they posses. Information about the culprit is equally spread across the agents and all agents can accuse a suspect.

Intel can choose to provide different (broader) information than requested by Accuser.

`WhoDunitEnv`-Sym (Fig. 3) In the previous environment, the agents are asymmetric in terms of the actions they can perform. Here, we propose a variant where all agents are equal in their actions, but different in the knowledge they posses. Agents start with full knowledge of all the suspects and their attributes, but each agent is given a different set of starting facts about the culprit $K_i = \{f_i^{(1)}, f_i^{(2)}, f_i^{(3)}\}$, where every fact is an attribute value. In each turn, an agent chooses an action $a \in \{\texttt{share}, \texttt{accuse}, \texttt{skip}\}$. For $a = \texttt{share}$, the agent selects a fact from K_i and outputs it in a message to the rest of the group. For $a = \texttt{accuse}$, the agent decides a suspect to accuse of being the culprit and with that ends the game. With $a = \texttt{skip}$, the game simply moves to the next agent, spending the turn.

Task complexity `WhoDunitEnv` has different levers for increasing environment complexity, which allows adjusting it for evaluation of agents with different capabilities. These levers include: (a) *suspect count*: changing the suspect count can change how long the starting context is and the probability of having two very similar suspects, (b) *attribute count*: changing the number of attributes each suspect has can create more specific suspects that are harder to set apart, and (c) *turn count*: the game is set at a time limit, which affects the behavior of agents. By limiting their time, we force agents to use their available information better.

4 Experiments

We conduct experiments on `WhoDunitEnv` and two other environments. For each environment, we use train and validation sets to train our monitors and a separate test set to evaluate them, as detailed below.

4.1 Environments

`WhoDunitEnv` We wish to have a challenging yet feasible environment. To this end, we set the number of suspects, turn limit, and number of attributes in the asymmetric variant to 10, 31, and 11, respectively. For the symmetric variant we set them all to 20. Attributes include clothing and personality-related features, while each attribute has 2-3 possible values (see the full list in §B).

In our experiments, we observed that agents often struggle to perform the task when it is described with the words "accuse", "suspect" and "culprit", potentially due to alignment procedures. Therefore, we rephrased the task with the words "award", "character" and "winner", instead. This does not affect the task itself but only how it is presented to the agents. We prompt agents to generate thoughts before predicting an action with ReAct (Yao et al., 2023) (see §F for the exact prompts). For monitoring, we use all the positions where the agent generated a suspect ID. We treat information sharing as a *reversible* and accusations as *irreversible*, so an intervention consists of resetting the entire game, as long as an accusation was not made.

We use 210, 90 and 180 environment instances as training, validation and test sets, respectively. Sets are separated by suspect descriptions and the culprit choice. Performance is measured by the percentage of games that end in identifying the culprit, termed Success-Rate. Additionally, we measure Precision, i.e. Success-Rate when a character was accused, and Game-Length for the average number of dialog turns, including interventions.

CodeGen Code generation tasks are a popular practical application for LLMs.(Jimenez et al., 2024; Xia and Zhang, 2022; Min et al., 2024).Typically, agents are given a task description and a starting code snippet, and tasked to produce Python code that solves the task (Chen et al., 2021; Jain et al., 2024). Inspired by debate frameworks (Du et al., 2024; Liu et al., 2024d) we leverage a multi-agent debate system for code generation, CodeGen. We assign 4 different coding roles, a judge, and a unit test writer (see §C). The first round consists of each coding agent g_i outputting their solution s_i^0. Following rounds j consist of the judge and tester evaluating previous solutions S^{j-1} and providing feedback F^j and a list of tests T^j. Coding agents then receive (S^{j-1}, F^j, T^j) and attempt solving the task again, also outputting their measurement of how helpful F^j and T^j were. Passing information through the judge is a basic form of communication, which we monitor. We treat code generation as irreversible, with the intervention consisting of the judge and tester writing new (F^j, T^j).

We use the HumanEval benchmark (Chen et al., 2021) to train and validate our monitor with a 70-30 split. Final test evaluation is done on 279 problems from LiveCodeBench (Jain et al., 2024) that were collected between August 2024 and February 2025, to assess the generalization between similar tasks. We report Pass@1, which measures the percentage of problems correctly solved by the environment.

GovSim (Piatti et al., 2024) is a collaborative environment for resource sustainability, where agents share a renewable starting resource R_0 that they harvest to maximize their gains. At every round, agents harvest the resource and then discuss and decide their plans for future rounds. At the beginning of each round j, the remaining resources R_j^* are doubled up to the original maximum: $R_{j+1} = \min(2R_j^*, R_0)$, encouraging agents to be efficient while ensuring sustainability. The discussion stage allows free communication between agents. In our experiments, we focus on the fishing task. The set of actions at turn j is defined as the possible amounts of resources to harvest, that is $a \in [0, R_j]$. To calculate features for our monitor, we consider all the position corresponding to mentions of resource amounts. Since actions consume the shared resource, they are *irreversible*, and the intervention consists of resetting only the previous communication round and allowing the agents to discuss again, without any knowledge of the reset or any reflection made after the last discussion.

We follow Piatti et al. (2024) and report Survival-Time, Survival-Rate and Efficiency. Survival-Time measures the number of rounds in which the shared resource remains above a minimal threshold γ. Survival-Rate is a binary metric indicating whether the Survival-Time is above a maximal round threshold m. Efficiency measures how well agents consume the shared resource, i.e.,

$$\text{Efficiency} = 1 - \frac{\max(0, c - \sum_j^m \sum_i^n \mathrm{r}(g_i, j))}{c}$$

where $\mathrm{r}(g_i, j)$ is the amount of fish consumed by agent g_i at round j and $c = \frac{m \cdot R_0}{2}$. Following Piatti et al. (2024), we set $\gamma = 5, m = 12$. Training, validation and testing sets consist of 26, 14 and 20 instances, respectively, and differ by their starting resource R_0.[3] Exact R_0 values are given in §D.1.

4.2 Monitoring & Intervention

For every environment and agent type, we train classifiers with different feature combinations, polynomial degrees $d \in [1, 5]$ and monitoring threshold $\tau \in [0, 1]$. From these classifiers, we choose the one that best performs on the validation set (see details in §D.2 and Tab. 5). This results in a simple monitor that estimates $P(\texttt{success})$ at every turn. For the interventions, we define a cap on the number of triggers, setting it to either one or two in WhoDunitEnv and one in GovSim and CodeGen. Agents with different roles exhibit different behavior and so can each trigger up to the cap.

4.3 Models

For WhoDunitEnv, we experiment with two strong open-weight models LLAMA-3.1-70B (Grattafiori et al., 2024) and QWEN-2.5-72B (Yang et al., 2024), and one proprietary model – GPT-4O (Achiam et al., 2023). We report average scores and standard error over four runs with LLAMA and QWEN and three runs with GPT-4O.

For CodeGen, we use LLAMA-3.1-70B. In GovSim, the performance of LLAMA-3-70B and QWEN-1.5-72B is near-zero (Piatti et al., 2024),[4]

[3] We extend the evaluation by Piatti et al. (2024) from 5 to 20 games to obtain a better performance estimate.

[4] We observed similar results with the newer LLAMA-3.1-70B and QWEN-2.5-72B.

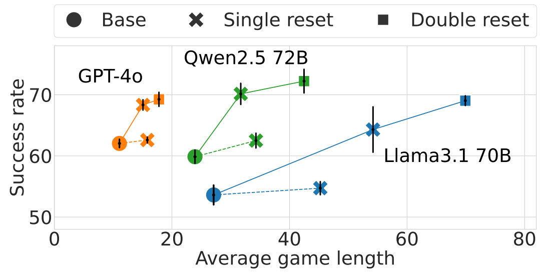

Figure 4: **Main results for `WhoDunitEnv-Asym`**. Live interventions lead to significant Success-Rate gains for three strong LLMs of up to 10 points, outperforming the random baseline (a cross connected by a dashed line). Resetting twice leads to additional, yet diminishing gains. Black lines indicate standard error.

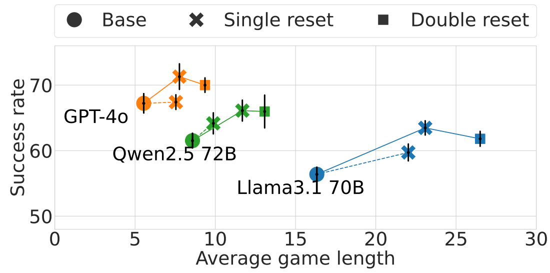

Figure 5: **Main results for `WhoDunitEnv-Sym`.** Dor all models, live interventions lead to significant improvement in Success-Rate of up to 7 points, outperforming the baseline (a cross connected by a dashed line). Interestingly, resetting twice does not offer further improvements. Black lines indicate standard error.

which leaves no positive examples for monitor training (we discuss applicability of our approach in the Limitations section). Thus, we use the stronger QWEN-1.5-110B (Bai et al., 2023) and GPT-4O (see more details in §D.3).

5 Results

We find that monitoring and interventions consistently improve system performance across all environments and models, leading to substantial gains of up to 17.4% in `WhoDunitEnv`, up to 20.0% in `GovSim` and up to 2.5% in `CodeGen`.

`WhoDunitEnv` Fig. 4 and 5 present the results for `WhoDunitEnv-Asym` and `WhoDunitEnv-Sym`, respectively. For `WhoDunitEnv-Asym` our method outperforms the base model by 6.1%, 10.6%, 10.3% for GPT-4O, LLAMA-3.1-70B, and QWEN-2.5-72B respectively, and by 4.1%, 7.0%, 4.5% for `WhoDunitEnv-Sym`. When one reset is allowed, our method outperforms the best baseline on average across models and the variants by 6.3%.

For `WhoDunitEnv-Asym`, more resets yield additional gains (11.8% vs 9.0% with one reset across models on average). Interestingly, the open-weight QWEN-2.5-72B and LLAMA-3.1-70B perform similarly to GPT-4O with two resets. However, a second reset did not further boost performance on `WhoDunitEnv-Sym`. We hypothesize that this is due to the structured form of communication in `WhoDunitEnv-Sym`, which would result in less communication-based mistakes.

Our interventions come at a cost of additional turns. The average number of turns increases by a factor of 1.9 and 1.6 with double and single reset for `WhoDunitEnv-Asym` and 1.6 and 1.4 for `WhoDunitEnv-Sym`. Detailed results per variant and model can be found at §D.4.

Method	HumanEval	LiveCodeBench
Zero-shot prompting	80.5%	18.2%
Multi-agent	81.6%	19.3%
Multi-agent + monitor	**83.5%**	**21.8%**

Table 1: **Performance on `CodeGen`** measured with Pass@1. Using LLAMA-3.1-70B with monitors trained for code generation on `HumanEval` shows gains that generalize to a similar task in `LiveCodeBench`.

`CodeGen` Tab. 1 presents the results for LLAMA-3.1-70B on `HumanEval` and `LiveCodeBench`, obtained with the monitors trained on `HumanEval`. Our method reaches a Pass@1 score of 83.5%, outperforming zero-shot prompting at 80.5% and a multi-agent network without monitoring which reaches 81.6%. Applying the same monitors to `LiveCodeBench`reaches a Pass@1 score of 21.6%, compared to 18.2% by zero-shot prompting and 20.4% in the multi agent system without monitoring. This shows that not only is our method effective for realistic tasks, but also that our monitors generalize well between similar tasks.

`GovSim` Tab. 2 presents the results for `GovSim`. With QWEN-1.5-110B, our method leads to an increase of 20.0% in Survival-Rate and 1.2 steps in Survival-Time. Efficiency remains similar, with 49.4% vs 48.8%. For GPT-4O, we increase Efficiency by 6.9% on average, while maintaining a maximal Survival-Rate. Overall, our method leads to better collaboration and performance on the task.

Model	Ours	Survival Rate	Survival Time	Efficiency
QWEN -1.5-110B	✗	35.0 ± 20.1	7.5 ± 0.7	$\mathbf{49.4} \pm 10.1$
	✓	$\mathbf{55.0} \pm 21.8$	$\mathbf{8.6} \pm 0.8$	48.8 ± 9.9
GPT-4O	✗	**100**	**12.0**	69.1 ± 6.6
	✓	**100**	**12.0**	$\mathbf{76.0} \pm 4.8$

Table 2: **Main results for `GovSim`.** Interventions lead to significant gains in Survival-Rate and Survival-Time with QWEN-1.5-110B and in Efficiency with GPT-4O. Numbers include 95% confidence intervals.

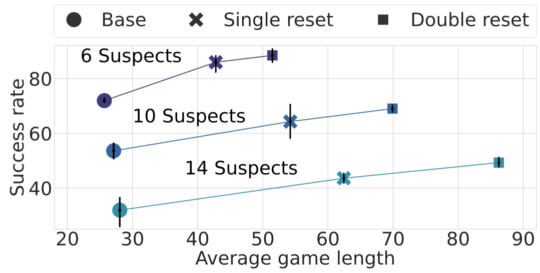

Figure 6: **Performance for `WhoDunitEnv-Sym` across difficulty levels**. Results are for LLAMA-3.1-70B with 6, 10, 14 suspects. Success-Rate decreases with more suspects, but gains are consistent.

6 Ablations and Analysis

We conduct additional analysis on `WhoDunitEnv-Asym`, demonstrating that our monitors capture meaningful signals of agent uncertainty and the effectiveness of our interventions.

6.1 Ablations

We conduct ablations to verify improvements are a result of our monitoring and intervening approach. For monitoring, we experiment with the classifiers that performed second best or the worst on the validation set to verify that classifier performance is correlated with final gains. For interventions, we experiment with resampling the agent upon trigger instead of resetting the communication channel.

Tab. 3 presents the results for QWEN-2.5-72B. For monitoring, the second best monitor leads to slightly lower gains (9.5% vs 10.3%) compared to the worst monitor with only 2% improvement, which is similar to the baseline at 2.6%. For interventions, resampling a single agent is ineffective, with a gain of 1.5%, which is 1.5% below the best random baseline. Overall, these results suggest that strong monitors and interventions are needed to improve performance.

Method	Success Rate	Average Length
Ours (Double reset)	$\mathbf{72.2} \pm 1.6$	42.5 ± 0.4
Ours (Single reset)	70.1 ± 1.7	31.7 ± 1.1
Monitoring: 2nd best monitor	69.3 ± 1.7	34.0 ± 1.5
Monitoring: Worst monitor	62.0 ± 1.8	30.6 ± 0.4
Intervention: Resample agent	61.3 ± 1.8	20.1 ± 0.4
Best baseline	62.5 ± 1.8	34.3 ± 0.7
No intervention	59.8 ± 1.8	23.9 ± 0.5

Table 3: **Ablations for monitoring and interventions.** Strong classifiers and interventions on the communication channel are needed to improve results. Experiments are with QWEN-2.5-72B and show standard error.

6.2 Performance by Task Complexity

Fig. 6 presents the results for LLAMA-3.1-70B on `WhoDunitEnv-Asym` with 6, 10, and 14 suspects, using the monitor trained for 10 suspects. We observe consistent gains of 14.0%, 10.7% and 11.7% in Success-Rate with a single reset and 16.5%, 15.4% and 17.4% with two resets across the different complexities. The average game length increases with complexity, from 1.9 with six suspects to 3.1 with fourteen suspects. This shows our method generalizes across task complexities with no required monitor retraining. Similar analysis for `WhoDunitEnv-Sym` is provided in §D.5.

6.3 Qualitative Analysis

To understand what phenomena are captured by our monitors, we perform a qualitative analysis of 50 examples in which monitors were triggered in `WhoDunitEnv-Asym` with LLAMA-3.1-70B. We observed four categories of monitor triggers: (a) *hallucination* (48% of reviewed cases): the information shared was incorrect, (b) *collapse* (16%): an agent repeatedly asks about the same suspect, even when they already received the relevant information, (c) *role loss* (8%): the agent loses track of their role in the game, and (d) *recall failure* (4%): Accuser failed to recall key information that was previously shared by the Intel. Fig. 7 presents an example for each category. In the remaining cases, we could not identify a concrete issue in the game trajectory, suggesting they may be instances of a *false trigger*.

Overall, 76% of the monitor triggers belong to one of the four main categories, with hallucinations being the most frequent trigger. Our analysis shows that our monitors successfully detect a wide range of problematic trajectories, and gains are due to

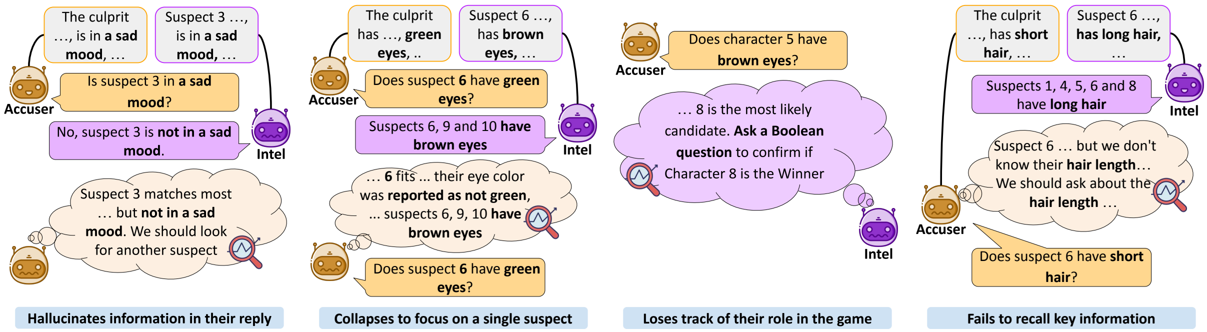

Figure 7: **Example monitor triggers for WhoDunitEnv-Asym with Llama-3.1-70B**. We categorize triggers into the following categories: hallucination (48%), agent collapse (16%), losing track of their role (8%) and recall failure (4%). The information relevant to the mistake is presented in bold text.

high monitoring accuracy in intervened games.

7 Related Work

Multi-agent communication Communication between agents have long been suggested in order to develop strong modular systems (Kraus, 1997; Dorri et al., 2018; Sukhbaatar et al., 2016; Foerster et al., 2016; Mikolov et al., 2018; Lazaridou and Baroni, 2020; Lowe et al., 2017). One of the most exciting applications of language agents is environments where agents autonomously communicate, with examples of improving factuality and reasoning via agent debate (Du et al., 2024; Liang et al., 2024), cooperation between embodied robots (Mandi et al., 2024; Chen et al., 2024b), simulating software development teams (Li et al., 2023a; Hong et al., 2024; Qian et al., 2024; Liu et al., 2024c), and gaming environments (Mukobi et al., 2023; Xu et al., 2024). Thus, several frameworks have been proposed to enable simple development of multi-agent environments (Li et al., 2023a; Wu et al., 2024; Hong et al., 2024). We contribute to this line of work by introducing `WhoDunitEnv`, a modular environment for multi-agent communication, which allows easy configuration of difficulty levels and communication structures.

Robustness of multi-agent systems Previous work has showed that multi-agent collaboration is susceptible to adversarial attacks (Wang et al., 2025; Amayuelas et al., 2024) and that Theory of Mind can improve collaboration in simple gaming environments (Lim et al., 2020). Our method of monitoring and intervention can complement existing methods by improving agent communication, and potentially prevent attacks such as poisoned information or prompt injections (Nazary et al., 2025; Greshake et al., 2023), which can be viewed as agents momentarily going *rogue*.

Uncertainty estimation in language modeling Uncertainty estimation has been shown useful in detecting and mitigating hallucinations in knowledge-intensive tasks (Kadavath et al., 2022; Yona et al., 2024; Ivgi et al., 2024), including in *agentic retrieval* where an external search engine is used (Jiang et al., 2023; Han et al., 2024). It has also been recently applied to language agents in order to increase exploration (Rahn et al., 2024), improve performance on bandit tasks (Felicioni et al., 2024), or make agents output textual estimates to help debates (Yoffe et al., 2024). In this work, we bridge uncertainty estimation and multi-agent collaboration by training monitors to predict the probability of task failures given agents uncertainty.

Aggregating multiple generations Aggregating over multiple generations is a popular method to increase performance (Wang et al., 2023; Yoran et al., 2023; Du et al., 2024; Chen et al., 2024a; Min et al., 2024). However, post-hoc aggregation is not directly applicable in agentic settings, where actions can be *irreversible*. Additionally, majority voting (Wang et al., 2023) requires at least three generations, while our approach increases turn account by less than twofold on average in our main experiments with `WhoDunitEnv`. Our work differs by resetting the communication channel *before* a problematic action was taken, rather than aggregating *after* the final prediction.

8 Conclusion

We present live monitoring and mid-run interventions for multi-agent systems. We demonstrate that monitors based on simple statistical measures can effectively predict future agent failures, and these failures can be prevented by restarting the communication channel. Experiments across multiple environments and models show consistent gains of up to 17.4%-20% in system performance, with an addition in inference-time compute. Our work also introduces `WhoDunitEnv`, a new environment for studying multi-agent cooperation.

Limitations

Environments In this work, we focus on `WhoDunitEnv`, a new framework for multi-agent communications, in addition to GovSim (Piatti et al., 2024). We believe that adapting our approach to additional environments is an exciting direction for future work. We note that our approach can be directly extended only to environments where agents achieve non-zero accuracy, as otherwise interventions will have no affect on performance. In addition, we showed that intervening by restarting an agent without the communication is less effective (§6). Our work can potentially be extended to single-agent settings by developing novel monitors and interventions.

Sample size For `WhoDunitEnv`, we experiment with 180 test environments . While this is a relatively small number, it is in line with recent evaluation sets that have 125-300 examples such as HumanEval (Chen et al., 2021), DrawBench (Saharia et al., 2022), Bamboogle (Press et al., 2023), and SWE-Bench (Jimenez et al., 2024). For `GovSim`, we extend from a single starting resource $R_0 = 100$ in (Piatti et al., 2024) to 20 choices. To reduce noise, we run every experiment between three and four times and report confidence intervals or standard deviation. Nevertheless, smaller evaluation sets also have some advantage as they require less compute to evaluate, thus having a smaller environment footprint (Schwartz et al., 2020) and allowing more research teams to experiment with our environments.

Data collection for training monitors To train our monitors to predict if an intermediate state of an agent will result in a task failure, we assume access to a set of train tasks. Realistically, collecting labels can be expensive, especially when task failures incur high costs. An exciting direction for future work is to frame our monitors as an online learning problem (Littlestone and Warmuth, 1994; Zinkevich, 2003; Park et al., 2024), where one has to learn to detect rogue agents whilst minimize the number of system failures.

Ethical Implications and Broader Impact

While multi-agent collaboration is an exciting direction for future research, it also entails significant risks. Strong multi-agent systems can potentially solve tasks beyond the reach of current AIs, a potential risk if used by a malicious user. In addition, multi-agent collaboration can have major economic and social impact. For example, a strong multi-agent system for autonomous software development can significantly increase productivity of engineering teams.

Another exciting future direction for multi-agent collaborations is operating embodied robots (Mandi et al., 2024; Chen et al., 2024b) and simulating human behavior (Park et al., 2023), which can cause significant social impact. In these domains, monitoring when future errors are likely and intervening when monitors are triggered can be especially important, as safe deployment presents a major challenge. Moreover, environmental considerations should also be taken into account, as multi-agent systems often incur high costs (seen from our experiments in D.3).

Acknowledgments

We thank Amit Elhelo for valuable feedback. This research was supported in part by AMD's AI & HPC Fund, the Google PhD Fellowship program, Len Blavatnik and the Blavatnik Family foundation. Figures 1, 2, 3, and 7 use images by Rank Sol on IconScout and from Freepik. The authors used AI models, specifically ChatGPT and Gemini, for implementing helper functions.

References

Aastha Acharya, Rebecca Russell, and Nisar R. Ahmed. 2022. Uncertainty quantification for competency assessment of autonomous agents. *Preprint*, arXiv:2206.10553.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany

Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

Saaket Agashe, Yue Fan, Anthony Reyna, and Xin Eric Wang. 2023. Evaluating multi-agent coordination abilities in large language models. *Preprint*, arXiv:2310.03903.

Alfonso Amayuelas, Xianjun Yang, Antonis Antoniades, Wenyue Hua, Liangming Pan, and William Yang Wang. 2024. MultiAgent collaboration attack: Investigating adversarial attacks in large language model collaborations via debate. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6929–6948, Miami, Florida, USA. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report.

Matthias Baldauf, Sebastian Müller, Arne Seeliger, Tobias Küng, Andreas Michel, and Werner Züllig. 2021. Human interventions in the smart factory – a case study on co-designing mobile and wearable monitoring systems with manufacturing staff. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan

Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2024a. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*.

Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2024b. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4311–4317.

Mike D'Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. 2024. Marg: Multi-agent review generation for scientific papers. *Preprint*, arXiv:2401.04259.

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

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Nicolò Felicioni, Lucas Maystre, Sina Ghiassian, and Kamil Ciosek. 2024. On the importance of uncertainty in decision-making with large language models. *Transactions on Machine Learning Research*.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2137–2145.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew

Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. *Preprint*, arXiv:2302.12173.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024a. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8048–8057. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Xudong Guo, Kaixuan Huang, Jiale Liu, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang, Thomas L. Griffiths, and Mengdi Wang. 2024b. Embodied llm agents learn to cooperate in organized teams. *Preprint*, arXiv:2403.12482.

Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. 2024. Towards uncertainty-aware language agent. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6662–6685, Bangkok, Thailand. Association for Computational Linguistics.

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

In *The Twelfth International Conference on Learning Representations*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Maor Ivgi, Ori Yoran, Jonathan Berant, and Mor Geva. 2024. From loops to oops: Fallback behaviors of language models under uncertainty. In *Second NeurIPS Workshop on Attributing Model Behavior at Scale*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

Xuhui Jiang, Yuxing Tian, Fengrui Hua, Chengjin Xu, Yuanzhuo Wang, and Jian Guo. 2024. A survey on large language model hallucination via a creativity perspective. *Preprint*, arXiv:2402.06647.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Sarit Kraus. 1997. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1):79–97. Economic Principles of Multi-Agent Systems.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Angeliki Lazaridou and Marco Baroni. 2020. Emergent multi-agent communication in the deep learning era. *Preprint*, arXiv:2006.02419.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Bangkok, Thailand. Association for Computational Linguistics.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. CAMEL: communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

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.

Terence X. Lim, Sidney Tio, and Desmond C. Ong. 2020. Improving multi-agent cooperation using theory of mind. *Preprint*, arXiv:2007.15703.

Leilei Lin, Yumeng Jin, Yingming Zhou, Wenlong Chen, and Chen Qian. 2024. Mao: A framework for process model generation with multi-agent orchestration. *Preprint*, arXiv:2408.01916.

N. Littlestone and M.K. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108(2):212–261.

Hao Liu, Zi-Yi Dou, Yixin Wang, Nanyun Peng, and Yisong Yue. 2024a. Uncertainty calibration for tool-using language agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16781–16805, Miami, Florida, USA. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024c. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024d. A dynamic llm-powered agent network for task-oriented agent collaboration. *Preprint*, arXiv:2310.02170.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6379–6390.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299.

Nikhil Mehta, Milagro Teruel, Xin Deng, Sergio Figueroa Sanz, Ahmed Awadallah, and Julia Kiseleva. 2024. Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1306–1321, St. Julian's, Malta. Association for Computational Linguistics.

Tomas Mikolov, Armand Joulin, and Marco Baroni. 2018. A roadmap towards machine intelligence. In *Computational Linguistics and Intelligent Text Processing*, pages 29–61, Cham. Springer International Publishing.

Marcus J. Min, Yangruibo Ding, Luca Buratti, Saurabh Pujar, Gail Kaiser, Suman Jana, and Baishakhi Ray. 2024. Beyond accuracy: Evaluating self-consistency of code large language models with identitychain. In *The Twelfth International Conference on Learning Representations*.

Gabriel Mukobi, Hannah Erlebach, Niklas Lauffer, Lewis Hammond, Alan Chan, and Jesse Clifton. 2023. Welfare diplomacy: Benchmarking language model cooperation. *Preprint*, arXiv:2310.08901.

Fatemeh Nazary, Yashar Deldjoo, and Tommaso di Noia. 2025. Poison-rag: Adversarial data poisoning attacks on retrieval-augmented generation in recommender systems. *Preprint*, arXiv:2501.11759.

Oladiran Olajiga, Emmanuel Ani, Kehinde Olu-lawal, Danny Montero, and Adeniyi Adeleke. 2024. Intelligent monitoring systems in manufacturing: Current state and future perspectives. *Engineering Science & Technology Journal*, 5:750–759.

Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. 2024. Do llm agents have regret? a case study in online learning and games. *Preprint*, arXiv:2403.16843.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA. Association for Computing Machinery.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Giorgio Piatti, Zhijing Jin, Max Kleiman-Weiner, Bernhard Schölkopf, Mrinmaya Sachan, and Rada Mihalcea. 2024. Cooperate or collapse: Emergence of sustainable cooperation in a society of LLM agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Nate Rahn, Pierluca D'Oro, and Marc G Bellemare. 2024. Controlling large language model agents with entropic activation steering. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. 2022. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*.

Aya Salem, Safaa Azzam, O. Emam, and Amr Abohany. 2024. Advancing cybersecurity: a comprehensive review of ai-driven detection techniques. *Journal of Big Data*, 11.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):54–63.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

P. Stavropoulos, D. Chantzis, C. Doukas, A. Papacharalampopoulos, and G. Chryssolouris. 2013. Monitoring and control of manufacturing processes: A review. *Procedia CIRP*, 8:421–425. 14th CIRP Conference on Modeling of Machining Operations (CIRP CMMO).

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2244–2252.

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. *Preprint*, arXiv:2501.06322.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. 2025. G-safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. *Preprint*, arXiv:2502.11127.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Daniel W. Woods and Sezaneh Seymour. 2023. Evidence-based cybersecurity policy? a meta-review of security control effectiveness. *Journal of Cyber Policy*, 8(3):365–383.

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. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Chunqiu Steven Xia and Lingming Zhang. 2022. Less training, more repairing please: revisiting automated program repair via zero-shot learning. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE '22, page 959–971. ACM.

Yijun Xiao and William Yang Wang. 2021. On hallucination and predictive uncertainty in conditional language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2734–2744, Online. Association for Computational Linguistics.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2024. Exploring large language models for communication games: An empirical study on werewolf. *Preprint*, arXiv:2309.04658.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Luke Yoffe, Alfonso Amayuelas, and William Yang Wang. 2024. Debunc: Mitigating hallucinations in large language model agent communication with uncertainty estimations. *Preprint*, arXiv:2407.06426.

Gal Yona, Roee Aharoni, and Mor Geva. 2024. Can large language models faithfully express their intrinsic uncertainty in words? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7752–7764, Miami, Florida, USA. Association for Computational Linguistics.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. AssistantBench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8938–8968, Miami, Florida, USA. Association for Computational Linguistics.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5942–5966, Singapore. Association for Computational Linguistics.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2024a. How language model hallucinations can snowball. In *Forty-first International Conference on Machine Learning*.

Ruiyi Zhang, Xaq Pitkow, and Dora E. Angelaki. 2024b. Inductive biases of neural network modularity in spatial navigation. *Science Advances*, 10(29):eadk1256.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models. *Preprint*, arXiv:2309.01219.

Martin Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 928–936. AAAI Press.

A Uncertainty estimation for monitors

In our monitoring, we use 3 statistical measures over the probability vector: Entropy, Varentropy and Kurtosis. Given a probability vector P, Entropy, Varentropy and Kurtosis are defined as:

$$\text{Entropy}(P) = -\sum_{p \in P} log(p) \cdot p$$

$$\text{Varentropy}(P) = -\sum_{p \in P} p \cdot (log(p) + \text{Entropy}(P))^2$$

$$\text{Kurtosis}(P) = \frac{\sum_{p \in P} p(-log(p) - \text{Entropy}(P))^4}{\text{Varentropy}(P)^2}$$

B WhoDunitEnv

Attribute list The suspects in WhoDunitEnv are defined by a collection of different attributes. These attributes are modular and can be swapped as need arises. In our experiments, we used 12 attributes for the asymmetric variant and 20 attributes for the symmetric variant. This was done to create a challenging yet feasible task for WhoDunitEnv-Sym, which is easier. The complete list of attributes and their possible values is given in Tab. 4:

C CodeGen

CodeGen is a multi-agent coding environment, heavily drawing from the debate frameworks presented in Du et al. (2024) and Liu et al. (2024d). Parts of the environment code of Liu et al. (2024d) was used to construct our coding generation debate environment, CodeGen, including the coding role implementation of 'PythonAssistant', 'AlgorithmDeveloper', 'ComputerScientist' and 'Programmer'. The dynamic network aspect of Liu et al. (2024d) was not used in our experiments.

Attribute	Values
hat	brown, black
mood	happy, sad
shirt color	pink, green
hobby	basketball, dancing
pants	long, short
pants color	brown, black
eye color	blue, brown, green
eye glasses	circular, square
shirt	button-up, tee
shoe color	red, white
hair	long, short
watch	bronze, silver
socks	dotted, white
jacket	yellow, jean
height	short, tall
age	young, old
build	medium, muscular
personality	introverted, extroverted
interests	sports, arts
occupation	professional, student

Table 4: **List of attributes used in the WhoDunitEnv variants experiments**. The top half appears in experiments of both variants, while the bottom one only appears in the symmetric variant experiments. This list is completely modular and can be changed to affect task complexity.

D Experiments

D.1 Data Splits

For WhoDunitEnv we define games by suspect attributes. For GovSim we make use of the starting resource, R_0. For Train and validation we shuffle and split $\{105 + 5k | k \in [0, 19]\}$ into two exclusive parts. Test is simply $R_0 \in \{[100], [210, 300]\}$. We make sure $R_0 = 100$ stays in the test set to align with results from Piatti et al. (2024).

D.2 Monitor Design

The monitor is made up of a classifier function together with a threshold. To train the classifier we first split the data set into a train set and a validation set. We use the training set to min-max normalize the data to the [-1, 1] range, then create polynomial features over input features with degree D, and finally fit a Ridge function over the resulting polynomials. These are done using the scikit-learn library (Pedregosa et al., 2011). We do this for $D \in [1, 2, 3, 4, 5]$ and for every combination of the 3 features. Turn count is always included in the model input. To find the optimal degree, feature combination and final classifier threshold $\tau \in [0, 1]$, we use the validation set.

For every game, we use the monitor to test triggers against the golden task success labels from

Model	Type	Features	Degree	Gain	τ
QWEN-2.5 -72B	Accuser	Var	1	5.8	0.51
	Intel	Ent	1	4.6	0.36
LLAMA-3.1 -70B	Accuser	Var Kur	5	11.6	0.55
	Intel	Ent Var	4	11.3	0.5
GPT-4O	Accuser	Ent	3	4.9	0.37
	Intel	Ent Var	5	5.3	0.43

Table 5: **WhoDunitEnv-Asym best monitors**. Best monitors for different models had different features and degrees selected.

Model	Full name
QWEN-2.5-72B	Qwen/Qwen2.5-72B-Instruct
LLAMA-3.1-70B	meta-llama/Llama-3.1-70B-Instruct
GPT-4O	gpt-4o-2024-08-06
QWEN-1.5-110B	Qwen/Qwen1.5-110B-Chat-GPTQ-Int4

Table 6: **Full model names used across experiments.**

the run. We consider the monitor successful on a game if even one sample of a failed game is over the threshold, since that would trigger a reset. We consider the monitor unsuccessful on a game if either the classifier was never over the threshold for a failed game, or was over the threshold for any turn on a successful game. Finally, we pick the monitor which had the maximum **gain** over the validation set. This would be the monitor we use for testing. See Tab. 5 for monitors used in WhoDunitEnv-Asym and their predicted gain. §E.1 shows a histogram of monitor turn triggers in a game without interventions.

For baselines, we used a random monitor with a set probability of triggering p. We applied the monitor to the validation set and chose $p \in [0, 1]$ that maximized gains.

D.3 Models

For the open-source models, we required at most 320GB of GPU memory to load the model and run the experiments. A load of 180 games took on average 6 hours. Using GPT-4O took about one hour for 180 games, and cost about 30$. When using GovSim, QWEN-1.5-110B required 160GB of total GPU memory, and each game took about 3 hours of run time (depending on how long the agents last in the environment). GPT-4O took between one and two hours per game, and cost about 8$ per a game. See table 6 for specific model names (HuggingFace for open weights). We make use of the vLLM library for open source model loading (Kwon et al., 2023). All model queries are done with the default model parameters, with the exception of temperature which is always set to 0.

D.4 Full Experimental Results

Due to the high variance we encountered during runs, we run a total of 180 test games for each variant of WhoDunitEnv. Due to cost, we only run GPT-4O three times on each game. The full results of each experiment, including Success-Rate and Precision can be seen at Tab. 7 for the asymmetric variant and at Tab. 8 for the symmetric.

D.5 WhoDunitEnv-Sym complexity analysis

As a complement to the WhoDunitEnv-Asym complexity analysis seen in §5, Fig. 8 shows a complexity analysis of WhoDunitEnv-Sym. Since our main results for WhoDunitEnv-Sym have 20 suspects (see §4), we conduct analysis with an easier 15 suspect setting and a harder 25 suspect setting. Similarly to WhoDunitEnv-Asym, we use the original monitor, trained with 20 suspect games. After a single reset, the 15 and 25 suspect settings saw a gain of 0.86% and 3.69% over their base experiments, respectively. With an additional reset, the 15 suspect setting only gained 1.83% over the base, while the 25 suspect setting saw an increased gain of 6.47%. The 25 suspect setting compares to the 20 suspect setting, which has an increase of 7.08% and 5.42%. Interestingly, the decrease seen after a double reset was not seen in the 25 suspect setting. The negligible gains in the 15 suspect setting fit with our analysis that the symmetric environment is easier when the number of suspects is small, hinting that communication problems are not very common in this setting.

E Analysis

E.1 Monitor trigger turns

Fig. 9 shows a histogram of the turn count for a monitor trigger on WhoDunitEnv-Asym, across all 3 models and all experiments. We see that triggers when the game ended up succeeding usually happen early. We hypothesize these are a result of a communication mistake that agents manged to recover from. On the other hand, communication mistakes later in the conversation often lead to task failure. 20% of games failed without a monitor trigger, leaving us further room for monitor improvement and gain.

Model	Int. #	Success -Rate	Precision
QWEN-2.5-72B	0	57.8	73.8
		62.8	79.0
		58.9	81.5
		60.0	77.1
	1	67.8	78.2
		73.9	87.5
		66.7	79.5
		72.2	81.2
	2	67.2	72.9
		72.8	77.5
		76.7	80.7
		72.2	77.4
	Baseline	58.9	72.6
		63.9	76.7
		63.9	81.6
		63.3	78.1
LLAMA-3.1-70B	0	56.1	73.7
		51.1	69.2
		50.6	78.4
		56.7	78.5
	1	55.6	71.4
		66.1	75.8
		62.2	74.2
		73.3	82.0
	2	69.4	77.2
		67.8	75.8
		71.1	77.6
		67.8	77.2
	Baseline	53.9	75.2
		57.8	72.2
		54.4	74.2
		52.8	72.0
GPT-4O	0	63.3	63.7
		61.7	61.7
		61.1	61.5
	1	70.0	70.4
		67.2	68.4
		67.8	67.8
	2	70.9	66.1
		69.8	65.0
		67.0	62.6
	Baseline	61.7	61.7
		63.3	63.3
		62.8	62.8

Table 7: **List of different experiments done with `WhoDunitEnv`-Asym** comparing normal running, using our method for one and two resets, and a baseline random classifier that can reset once. Each of these is an average over the exact same 180 games, which were defined with 10 suspects. For each run we bring the Success-Rate and Precision.

Model	Int. #	Success -Rate	Precision
QWEN-2.5-72B	0	63.3	64.0
		58.3	58.3
		62.8	63.5
		61.7	62.0
	1	62.2	62.9
		70.0	70.4
		65.6	65.9
		66.7	67.0
	2	71.1	71.9
		69.4	69.4
		61.7	62.4
		61.7	61.7
	Baseline	67.2	67.2
		66.7	67.0
		61.7	62.0
		61.1	61.5
LLAMA-3.1-70B	0	53.9	57.7
		55.6	59.9
		57.2	61.3
		58.9	63.5
	1	61.7	64.5
		66.7	71.9
		62.8	64.9
		62.8	64.9
	2	63.9	65.0
		63.3	65.5
		58.9	59.9
		61.1	63.6
	Baseline	60.0	65.1
		62.2	66.7
		56.1	61.6
		60.6	62.3
GPT-4O	0	65.0	65.0
		70.0	70.0
		66.7	66.7
	1	68.3	68.3
		75.0	75.0
		70.6	70.6
	2	68.9	68.9
		72.2	72.2
		68.9	68.9
	Baseline	69.4	69.4
		67.2	67.2
		65.6	65.6

Table 8: **List of different experiments done with `WhoDunitEnv`-Sym** using our method, including no resets and baseline random classifiers. Each of these is an average over the exact same 180 games, which were defined with 20 suspects. For each run we bring the Success-Rate and Precision.

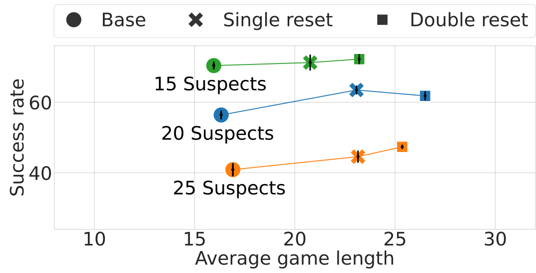

Figure 8: **Performance for WhoDunitEnv-Sym across difficulty levels**. Results are for LLAMA-3.1-70B with 15, 20, 25 suspects. Success-Rate decreases with more suspects. Gains with 15 suspects is negligible, however gains with 25 suspects are similar to the original 20 suspect setting.

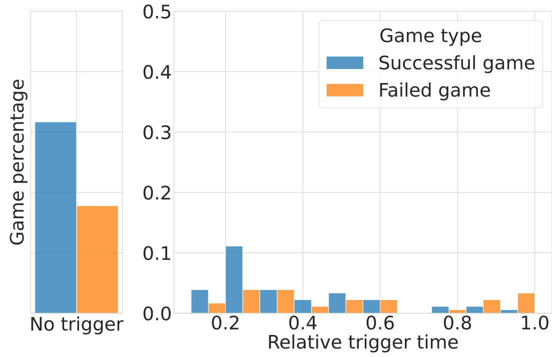

Figure 9: **Histogram of monitor triggers** for the WhoDunitEnv-Asym test set using GPT-4O, normalized by the 180 game count.

F WhoDunitEnv Prompts

We make here available the full text of WhoDunitEnv prompts. For WhoDunitEnv-Sym this includes System prompts for all agents (Fig. 10), user prompts (Fig. 11), and message formats (Fig. 12). For WhoDunitEnv-Asym this includes System prompts for Accuser (Fig. 13) and Intel (Fig. 14), User prompts (Fig. 15) and message formats (Fig. 16).

G WhoDunitEnv Full Example - Hallucination

For better understanding, we bring here the full text of the hallucination example, shown in §6. Due to length, the example is split into 2 figures, the system prompt (Fig. 17) and the user prompt and assistant answer (Fig. 18).

System Prompt

Introduction:
- Your name is NAME, and you are a helpful AI assistant participating in a cooperative game with other players. There are AGENT_COUNT players in total: AGENT_NAMES.

Game Overview:
- The game involves characters, one of which is the Winner and will receive an award! Each player has information about all characters and partial knowledge about who the Winner is.
- You will receive information in the following format:

1. A list of characters with their descriptions.
2. A list of starting facts about the Winner, describing specific properties. These are only visible to you.
3. The current communication channel between you and other players, containing shared facts.

Game Mode:
- The game proceeds in rounds. In each round, you can either award the prize to a character you believe is the Winner, share one of your starting facts or skip your turn.
- If there is only one character matching the criteria according to your information, you should award them the prize. Otherwise, if there's a fact that adds to your team's understanding, you should share a fact. If you believe you have nothing to add, you can skip your turn.
- The goal is to find the correct Winner in the minimum number of rounds. Awarding an incorrect Winner will lose points.
- **Output Format in Game Mode:**

```
{
  "thoughts": "Thoughts describing your optimal action at this point.
  Describe whether to share a fact, award a character or skip your turn,
  and why you chose this action.",
  "action": 1, 2 or 3,  // 1 to share a fact, 2 to award a character
  or 3 to skip your turn
  "fact": If sharing, fact number to share. This must be a number.
  "character": If accusing, character number of accuse. This must be a number.
}
```

Instructions Recap:
- Focus on minimizing rounds to find the Winner.
- Share informative facts to narrow down the possibilities.
- Award a character only when sure of their identity as the Winner.
- Follow the specified JSON format for output in Game mode.

SUSPECT_INFORMATION

Figure 10: **WhoDunitEnv-Sym system prompts.**

User prompt

You received the following starting facts about the Winner (these are the facts you may share):
FACTS

The current communication channel between you and the other players is:
COMM_CHANNEL

The current turn is TURN_COUNT out of MAX_TURN_COUNT.
It's your turn. What would you do?

Figure 11: **WhoDunitEnv-Sym user prompts.**

User Prompt and Messages

You received the following starting facts about the Winner (these are the facts you may share):
FACTS

The current communication channel between you and the other players is:
COMM_CHANNEL

The current turn is TURN_COUNT out of MAX_TURN_COUNT.
It's your turn. What would you do?

Player NAME has decided to share a fact about the Winner: FACT.

The winner is VALUE.

Figure 12: **WhoDunitEnv-Sym User prompt, message and fact formats.**

Accuser System Prompt

Introduction:
- Your name is NAME, and you are a helpful AI assistant participating in a cooperative game with another player: PARTNER_NAME.

Game Overview:
- The game involves SUSPECT_NUM characters, identified by numbers 1 to SUSPECT_NUM. One of the characters has been selected as the Winner and will be awarded a prize.
- You and your partner have access to different information. You know the Winner's description, while your partner knows the description of every character but doesn't know which one is the Winner.
- There is only one Winner, and only that character will exactly fit the Winner's description.
- You will receive information in the following format:

1. A list of properties about the Winner.
2. The current communication channel between you and your partner, containing shared properties.

Game Modes:

1. Game Mode:
- The game proceeds in rounds. In each round, you can either award the prize to a character, making them the Winner, or ask your partner for information.
- If there is only one character matching the criteria according to your information, you should award them the prize. Otherwise, ask your partner for the most helpful information.
- You can ask for information in two ways:

1. Ask if a specific character has a specific value for a property, and get a boolean answer. You must include both a property and a value for this action.
2. Ask for general info, receiving a property and value combination, and a list of all characters that fit that combination. Your partner will decide what property to share.

- The goal is to find the correct Winner in a minimum number of rounds. Once you award a character, the game will end.
- **Output Format in Game Mode:**

```
{
 "thoughts": "Thoughts describing your optimal action at this point.
 Describe whether to request information or award a character, and why you chose this action.",
 "action": 1, 2 or 3, // 1 to request specific information, 2 to request general information,
 and 3 to award a character
 "character": Character id to request information about or award, depending on your action.
 This must be a number.,
 "property": "Relevant if action is 1. The property to ask for information about.",
 "value": "Relevant if action is 1. Value of the property that is asked about"
}
```

2. Analysis Mode:
- In this mode, you will answer questions about the task. Rounds do not count towards the goal, and you will not request or award.
- A different output format will be provided.

Instructions Recap:
- Focus on minimizing rounds to find the Winner.
- Award as soon as you are sure of who the Winner is.
- Request informative facts to narrow down the possibilities.
- Follow the specified JSON format for output in Game mode.
- Use your knowledge to eliminate characters who cannot be the Winner and to narrow down the possibilities.

Possible attributes for Characters are: CHARACTER_ATTRIBUTES

The Winner is described as:
WINNER_DESCRIPTION

Figure 13: **WhoDunitEnv-Asym accuser system prompt.**

Intel System Prompt

Introduction:
Your name is NAME, and you are a helpful AI assistant participating in a cooperative game with another player, PARTNER_NAME.

Game Overview:
The game involves characters, with one character identified as the Winner.
You have descriptions for all characters but do not know which one is the Winner.
Beth has the description of the Winner, but not the character descriptions.
You must help your partner discover which of the characters is the winner.
You will receive information in the following format:

1. A list of characters with their descriptions.
2. The current communication channel between you and Beth, containing shared properties.

Game Modes:

1. Game Mode:
The game proceeds in rounds. In each round, you will receive a request from your partner.
The request would either be a Boolean Question about one of the characters or a request for a Broad Message.
Boolean question: Given the property, value and one possible character, answer whether that combination is true or false.
Broad Message: Share the value of a single property and a list of characters that have that value for the property.
Where sharing a broad message, you choose what property to share. Try sharing the most efficient property and value combination.
The goal is to identify the Winner in the minimum number of rounds.

Output Format in Game Mode:

```
{
"thoughts": "Describe your optimal action at this point.
Explain what is your response and why.",
"action": 1 or 2 // 1 to answer a request, 2 for a broad message,
"value": If the action is 1, this should be a boolean True/False value.
If the action is 2, this should be a combination of a property name and a value separated by a -:
property-value
"character": "Relevant only if action is 2.
List of all characters who have the property and value combination shared."
}
```

2. Analysis Mode:
In this mode, you will answer questions about the task.
Rounds in this mode do not count towards the goal, and you will not make requests or accusations.
A different output format will be provided in this mode.

Instructions Recap:
Focus on minimizing rounds to identify the Winner.
Fulfill Beth's requests or broadcast your own messages to narrow down possibilities.
Follow the specified JSON format for output in Game Mode.
Possible attributes for Characters are: CHARACTER_ATTRIBUTES

Following is information about the Characters:
SUSPECT_INFORMATION

Figure 14: **WhoDunitEnv-Asym Intel system prompts.**

User Prompt

The current communication channel between you and the other players is:
`COMMUNICATION_CHANNEL`

The current turn is `TURN_COUNT` out of `MAX_TURN_COUNT`
It's your turn. What would you do?

Figure 15: **`WhoDunitEnv`-Asym User Prompt, which is identical between Accuser and Intel.**

Agent Messages

Request Specific

Agent `NAME` has requested information: `BOOLEAN_QUESTION`

Request Broad

Agent `NAME` has asked for general information (a broad message)

Specific Answer

Agent `NAME` has responded that character `CHARACTER` [is | is not] `ANSWER`

Broad message

Agent `NAME` has decided to return a broad message: For characters `CHARACTER_LIST`, the property `PROPERTY` is `ANSWER`

Figure 16: **`WhoDunitEnv`-Asym message formats.**

Hallucination example - System prompt. Culprit is character 3

```
System:
**Introduction:**
- Your name is `Beth`, and you are a helpful AI assistant participating in a cooperative game with another player: Alex.

**Game Overview:**
- The game involves 10 characters, identified by numbers 1 to 10.
One of the characters has been selected as the Winner and will be awarded a prize.
- You and your partner have access to  different information.
   You know the Winner's description, while your partner knows the description of every character
   but doesn't know which one is the Winner.
- There is only one Winner, and only that character will exactly fit the Winner's description.
- You will receive information in the following format:
  1. A list of properties about the Winner.
  2. The current communication channel between you and your partner, containing shared properties.

**Game Modes:**

1. **Game Mode:**
- The game proceeds in rounds. In each round, you can either award the prize to a character,
making them the Winner, or ask your partner for information.
- If there is only one character matching the criteria according to your information, you should award them the prize.
  Otherwise, ask your partner for the most helpful information.
- You can ask for information in two ways:
  1. Ask if a specific character has a specific value for a property, and get a boolean answer.
    You must include both a property and a value for this action.
  2. Ask for general info, receiving a property and value combination, and a list of all characters that fit that combination.
    Your partner will decide what property to share.

- The goal is to find the correct Winner in a minimum number of rounds. Once you award a character, the game will end.
- **Output Format in Game Mode:**
```json
{
 "thoughts": "Thoughts describing your optimal action at this point.
   Describe whether to request information or award a character, and why you chose this action.",
 "action": 1, 2 or 3, // 1 to request specific information, 2 to request general information, and 3 to award a character
 "character": Character id to request information about or award, depending on your action. This must be a number.,
 "property": "Relevant if action is 1. The property to ask for information about.",
 "value": "Relevant if action is 1. Value of the property that is asked about"
}
```

Instructions Recap:
- Focus on minimizing rounds to find the Winner.
- Award as soon as you are sure of who the Winner is.
- Request informative facts to narrow down the possibilities.
- Follow the specified JSON format for output in Game mode.
- Use your knowledge to eliminate characters who cannot be the Winner and to narrow down the possibilities.
Possible attributes for Characters are: eye_color, eye_glasses, hair, hat, hobby, mood,
  pants, pants_color, shirt, shirt_color, shoe_color, watch.

The Winner is described as:
their eye color is blue, they have circular eye glasses, their hair is long, is wearing a brown hat, their hobby is basketball,
  is in a sad mood, they have red shoes, they have a bronze watch,
  they have short, black pants and a pink, button-up shirt.

The current communication channel between you and the other players is:
Agent Beth has requested information: is property eye_glasses of character 1 circular?
Agent Alex has responded that Character 1 is not wearing circular eye glasses
Agent Beth has asked for general information (a broad message)
Agent Alex has decided to return a broad message: For characters: [1, 3, 5, 8, 9], the property hobby is basketball
Agent Beth has requested information: is property eye_glasses of character 3 circular?
Agent Alex has responded that Character 3 is wearing circular eye glasses
Agent Beth has requested information: is property eye_color of character 3 blue?
Agent Alex has decided to return a broad message: For characters: [1, 2, 4, 5, 6, 7, 9, 10], the property eye_color is brown
Agent Beth has asked for general information (a broad message)
Agent Alex has decided to return a broad message: For characters: [1, 5, 8, 10], the property hat is black
Agent Beth has requested information: is property eye_color of character 3 blue?
Agent Alex has decided to return a broad message: For characters: [3, 8], the property eye_color is blue
Agent Beth has requested information: is property hat of character 3 brown?
Agent Alex has decided to return a broad message: For characters: [2, 3, 6, 10], the property eye_glasses is circular
Agent Beth has requested information: is property hat of character 3 brown?
Agent Alex has decided to return a broad message: For characters: [1, 2, 3, 4, 5, 6, 7, 9], the property shoe_color is red
Agent Beth has requested information: is property hat of character 3 brown?
Agent Alex has responded that Character 3 is wearing a brown hat
Agent Beth has requested information: is property hair of character 3 long?
Agent Alex has decided to return a broad message: For characters: [1, 4, 8, 9], the property mood is happy
Agent Beth has requested information: is property hair of character 3 long?
Agent Alex has responded that Character 3's hair is long
Agent Beth has requested information: is property mood of character 3 sad?
Agent Alex has responded that Character 3 is not in a sad mood

The current turn is 25 out of 31. It's your turn. What would you do?
```

Figure 17: **Hallucination** example - System prompt.

Hallucination example - User prompt and answer. Culprit is character 3

The current communication channel between you and the other players is:
Agent Beth has requested information: is property eye_glasses of character 1 circular?
Agent Alex has responded that Character 1 is not wearing circular eye glasses
Agent Beth has asked for general information (a broad message)
Agent Alex has decided to return a broad message: For characters: [1, 3, 5, 8, 9], the property hobby is basketball
Agent Beth has requested information: is property eye_glasses of character 3 circular?
Agent Alex has responded that Character 3 is wearing circular eye glasses
Agent Beth has requested information: is property eye_color of character 3 blue?
Agent Alex has decided to return a broad message: For characters: [1, 2, 4, 5, 6, 7, 9, 10], the property eye_color is brown
Agent Beth has asked for general information (a broad message)
Agent Alex has decided to return a broad message: For characters: [1, 5, 8, 10], the property hat is black
Agent Beth has requested information: is property eye_color of character 3 blue?
Agent Alex has decided to return a broad message: For characters: [3, 8], the property eye_color is blue
Agent Beth has requested information: is property hat of character 3 brown?
Agent Alex has decided to return a broad message: For characters: [2, 3, 6, 10], the property eye_glasses is circular
Agent Beth has requested information: is property hat of character 3 brown?
Agent Alex has decided to return a broad message: For characters: [1, 2, 3, 4, 5, 6, 7, 9], the property shoe_color is red
Agent Beth has requested information: is property hat of character 3 brown?
Agent Alex has responded that Character 3 is wearing a brown hat
Agent Beth has requested information: is property hair of character 3 long?
Agent Alex has decided to return a broad message: For characters: [1, 4, 8, 9], the property mood is happy
Agent Beth has requested information: is property hair of character 3 long?
Agent Alex has responded that Character 3's hair is long
Agent Beth has requested information: is property mood of character 3 sad?
Agent Alex has responded that **Character 3 is not in a sad mood**

The current turn is 25 out of 31.
It's your turn. What would you do?

Answer: '"'json{ "thoughts": "Character 3 matches most of the Winners description, **but we found out they are not in a sad mood**. We should now look for another character that matches the description of having blue eyes, circular glasses, long hair, a brown hat, and being in a sad mood. Since we already know that character 3 has blue eyes, circular glasses, long hair, and a brown hat, we should look for another character that also has these properties **and is in a sad mood.**", "action": 2, "character": 0, "property": "", "value": "" } "' I am asking for general information because we need to find another character that matches most of the Winners description and is in a sad mood. We can then ask more specific questions to narrow down the possibilities.'

Figure 18: **Hallucination** example - User prompt and response.