# Extended Abstract

**Motivation** Over the past half-decade, LLMs have made immense strides in tackling increasingly complex reasoning tasks. A key step in this process was the expansion from single-agent systems to multi-agent. Single-agent systems have grown in capabilities due to modern techniques, of which a major one is Verbal Reinforcement Learning (VRL): integrating textual feedback (reflections) in a linguistic loop to edit a model's prompt. However, the cross section of these two areas is relatively unexplored because the application of textual feedback in a multi-agent system is extremely time and cost expensive. Naive approaches tend to overly-generalize, and thus produce far less improvement. Thus, in an effort to better capture these possibilities, we are motivated to find an optimization of these strategies that instead posits techniques to identify subsets of agents to alter and provide a practical multi-agent system with VRL.

**Method** Our setup included three agents working in a fixed order. The first agent, the Task Breakdown Specialist, decomposed the task into sub-tasks. The second agent, the Sub-task Resolver, was asked to solve each part. Finally, the Final Answer Synthesizer combined the output of the Resolver into a final answer choice. Our baseline is a supervised finetuned variant of the three agents. To build on this, we introduced a localizer-based self-reflection system. We added an Evaluator to judge correctness, and if the answer was incorrect, a Localizer reviewed the full trajectory and feedback to identify which agent propagated the error. A modified Reflector then generated targeted, agent-specific reflections. We explored three key design choices in this setup. Visibility controlled how the reflections were shared. Memory referred to how past reflections were retained. Token control focused on balancing performance and efficiency by varying input/output length.

**Implementation** For the task, we used the Moral Scenarios category from the MMLU dataset. This dataset consists of multiple-choice questions tasking the system to evaluate moral situations. We used 450 samples for training and 100 for validation. To implement our baseline, we passed training examples through the multi-agent pipeline and collected trajectories that led to correct answers. These were used to fine-tune each agent individually using supervised fine-tuning on an 8-bit quantized model. We used LLaMa-3.2-Instruct-3B. For the localizer experiments, we used a form of online self-reflection during training. To test generalization on the validation data, we froze the memory collected during training and ran validation tasks without retries. We had four memory freezing techniques: a random subset, the last 10, a condensed summary, and semantic-based retrieval. Finally, we measured overall success rate, average number of attempts per sample, and token usage.

**Results** Our baseline model that was trained using supervised finetuning achieved a 54% success rate across 100 validation tasks. After applying the localization and self reflection loop, our best training configuration achieved 69% success rate across 450 training tasks and 66% on validation. Under the conditions of freezing the trained memory, our second validation experiments reached a best of 35.5% success rate with no retries. The best experiment configuration required an average of 5186.492 tokens to attempt the task, and 13046.927 tokens to successfully solve a task.

**Discussion** We have found multiple forms of localization across cost-levels, with retries, that are able to outperform a finetuned baseline approach. Specifically, the use of agents with a 3-reflection memory, localizer with memory of condensed reflections, and global visibility of all reflections produce over 11% gain in success rate. Furthermore, the use of a condensed memory of size 3 and visibility of reflections only to the agent of choice maintains over 10% success rate gain while cutting token-cost by 45% in relation to the 10-memory sizing. Experimentation in using single-shot inference, in-place of retrial loops, across several memory-selection techniques all produce deprecated performance. It appears visibility and localizer memory produce non-generalizable reflections and trends, thus indicating there does not exist a significant impact of these variables on task success.

**Conclusion** Through aforementioned experimentation and ablation studies, we contribute a high performing verbal reinforcement multi-agent optimization that maintains moderate and practical token consumption. We further highlight rigorous few-shot exploration suggests this reflection-based approach relies upon high task-specific memory changes to the state making fixed memory from training for validation a strong limitation in performance. Instead, we motivate a need for a traditionally trained reflection model to be paired with the verbal-RL technique to better learn invariance to precise question/problem details.

# Verbal RL For Multi-Agent Systems

**Aakriti Lakshmanan**
Department of Computer Science
Stanford University
aakritil@stanford.edu

**Rohan Davidi**
Department of Computer Science
Stanford University
rohand25@stanford.edu

**Sathvik NaLLaMalli**
Department of Computer Science
Stanford University
sathvik9@stanford.edu

## Abstract

Large Language Models (LLMs) have contributed to the high performance of multi-agent systems by creating specialized agents capable of collaborating on the task. At the same time, a new reinforcement technique, Verbal Reinforcement Learning (VRL), offers improved performance for single agent systems by replacing gradient-driven updates with LLM-generated linguistic feedback serving as model states. Introduced in 2023, the *Reflexion* paper demonstrates iteratively retrying on a task while altering these textual prompts with a generated reflection can yield such improvement. In this work, we explore and offer multi-agent optimizations for using VRL to capture the performance gain demonstrated at the single-agent level while maintaining controls on token-costs for scalability and practicality. Explored in the multitask environment of moral reasoning, we demonstrate the use of "Localization." Replacing the expensive use of reflection on every agent in a pipeline or the vagueness of propagating a single reflection, this technique determines the agent causing the bottleneck error on each iteration serving as the target agent for reflection. Doing so, we find various configurations of ranging token consumptions that all produce significant improvements of over 10% in success rate over traditional supervised finetuned baselines. This work does extend previous work in VRL, but also finds the use of linguistic reflections to be narrow to a specific problem/task making zero-shot implementations with frozen memory states to be largely limited. This suggests traditional weight-based finetuning must be explored in-tandem to provide more rigorous and generalizable approaches when seeking single-try inference performance.

## 1 Introduction

Over the last few years Large Language Models (LLMs) have become increasingly performant in a wide range of tasks including logical decision-making and exercising reasoning capabilities. In this growth, a major leap has taken the shape of integrating multiple models in tandem to think in steps/parts as opposed relying on the internal steps of a single agent. With the ability to assume multiple perspectives, experimentation in economic games and bargaining settings have revealed that the onset of multi-agent systems has expanded the domain of performance to far more complicated reasoning tasks Sreedhar and Chilton (2024). A more broad survey of environments by Guo et. al. reveals that improvement in performance from a single-agent to multi-agent system is evident in several fields from societal simulations, scientific debates, and software development Guo et al. (2024).

Stanford CS224R 2025 Final Report

Meanwhile, in the same effort to improve reasoning capabilities, Verbal Reinforcement Learning (VRL) was introduced in 2023's "Reflexion: Language Agents with Verbal Reinforcement Learning" providing an alternate perspective to traditional techniques of modifying a model's state of numeric weights Shinn et al. (2023). The paper contributes a novel strategy of maintaining the system's state as its current prompt/memory as opposed to the model weights which remain unchanged. Instead of updates to these weights, textual feedback (a reflection from a reflector LLM) is generated based on the evaluation of a task completion and prepended to the prompt to alter the state towards improved performance.

Designed in a single-agent setting, the procedure for applying VRL is straightforward. However, expanding this to a multi-agent system to reap the aforementioned benefits of expanded logical capabilities requires delicate optimization. More specifically, a naive solution of performing the same reflection for each agent incurs immense costs considering it requires, per agent, a reflection by an LLM call to analyze the error and generate feedback. Whereas, other techniques relying upon one single reflection for the entire system dilutes the reasoning capabilities of the multi-agent setup as iterative improvements are provided in a non-individual generalized scale.

In this work, we experiment with various techniques of identifying bottleneck agents for a given error to, instead, on each iteration, alter through a reflection the most important/problematic agent in the pipeline. In other words, we explore various forms of "localization" in order to generate a more cost-efficient integration of verbal reinforcement learning into multi-agent systems to harness the capabilities of both the setup and technique to produce improvement on traditional techniques such as supervised finetuning / imitation learning.

## 2 Related Work

In Reflexion, Shinn et al. (2023) offered a new idea of applying reinforcement learning through linguistic feedback in place of derivative updates to the weights parameters of models. Reimagining the reinforcement loop to instead incorporate textual reflections, the paper designs an architecture as follows: (1) execute inference with current prompts/memory, (2) evaluate response, (3) pass evaluation and current model state to Reflector model, (4) take Reflector's reflection and prepend to model's prompt/memory Shinn et al. (2023). With this strategy in place with a designed prompt for the Reflector to promote high-quality specific information being provided to the agent, the team demonstrates significant improvement in multiple decision-making and coding tasks. This architecture provides both added performance and interpretability.

To incorporate these benefits into a multi-agent setting, we can observe contributions by "TextGrad: Automatic 'Differentiation' via Text" in which textual feedback is integrated via back propagation mimicking techniques used in traditional weights modifications Yuksekgonul et al. (2024). In this work, Yuksekgonul et. al. (2024) proposes maintaining a low token-cost by utilizing a single reflection for a multi-agent inference that is then differentiated with a customized loss function to distill the reflection on a per-agent level. Though the strategy avoids expensive reflections for every single agent, it incurs its own limitations as oftentimes textual feedback is not rich enough to distill into valuable information for all agents. The result is the bloating of agent memory with information that is largely general as, oftentimes, the majority of agents in the pipeline have little propagated corrections on a given error which is more likely rooted in a small subset of agents instead.

Other works in the space include that of Bo et. al. (2025) that posits the traditional training of the reflection model to perform a per-agent reflection Bo et al. (2024). Similarly, alternatives include making per-agent reflection calls to ensure specificity of reflections. However, both of these strategies involve sizeable cost either in training or inference. Though these explorations have yielded promising approaches, none explore the possibility of, instead, using identification of specific agents, as opposed to all agents, to drive the reflection process.

## 3 Method

### 3.1 Dataset

We used the Moral Scenarios category from the MMLU (Massive Multitask Language Understanding) dataset. This category contained multiple choice questions containing two scenarios. The goal of

the task was to choose whether or not the scenarios were in line with current moral standards. This dataset had a train size of 850 samples, of which 450 were used for training, and a validation size of 100 samples.

### 3.2 Multi-Agent Architecture

To address the task, we built on the framework introduced in Collaborative Multi-Agent, Multi-Reasoning-Path Prompting Chen et al. (2024) by implementing a sequential setup involving three distinct agents. The first agent, known as the Task Breakdown Specialist, was provided the question and a prompt that encouraged the agent to split the question into solvable sub-tasks. The second agent, the Sub-Task Resolver, was provided the output of the first agent and the question and asked to solve the tasks. Finally, the third agent, the Final Answer Synthesizer, was provided the output of the second agent and requested to summarize everything into a concrete final answer. We also provided prompt scaffolding to ensure that the model provided the answer choice in a way that was deterministically parsable by a regex algorithm.

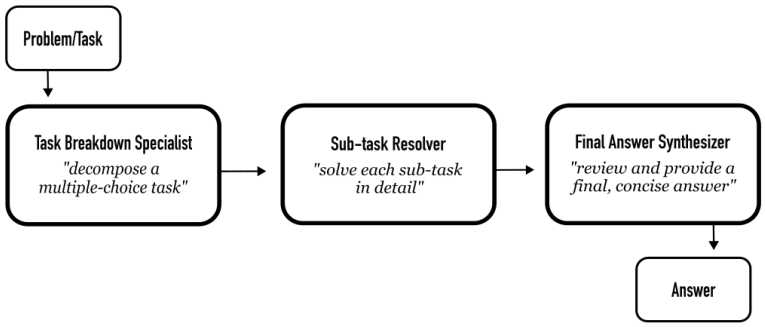


Figure 1: The baseline sequential multi-agent architecture for the Moral Scenarios dataset.

### 3.3 Baseline Implementation

To properly evaluate the impact of our localizer implementations, we developed a comprehensive baseline for testing the multi-agent setup. We determined that few-shot fine-tuning each agent would help assess whether RL-based approaches improve the final accuracy. First, we collected trajectories by passing in the train examples through the sequential multi-agent setup shown in 1. Trajectories that led to a positive outcome (i.e a correct answer) were then used to fine-tune each of the agents individually.

We fine-tuned each of the three agents using supervised fine-tuning (SFT) on an 8-bit quantized version for 4 epochs. Then, using the fine-tuned agents, the multi-agent setup was run on the validation samples, and the resulting outputs were collected.

### 3.4 Localizer Implementation

The next stage of our experimentation was to determine if using a "localizer" - i.e, a a bottleneck identifying module - in addition with verbal reinforcement during self-refinement would perform better than the baseline. In order to implement the localizer within the multi-agent setup, we added an evaluator, reflector, and localizer module. The Evaluator was prompted to provide an evaluation of the final answer when provided the results of the regex matching algorithm. The Localizer was created by prompting the model to analyze the feedback and full trajectory to identify the single agent that led to the incorrect answer. The output of the Localizer and the evaluation is provided to the modified reflector module. The Reflector focused on just the identified agent, aiming for targeted, task-specific reflections rather than general feedback. Each agent maintained a long-term memory of past reflections, prepended to its prompt to inform future runs.

When developing this module, we recognized that numerous variables could influence the efficiency of the Localizer itself. We chose to concentrate on three primary factors: visibility, memory, and

token control. Visibility addressed whether reflections identifying the bottleneck agent should be provided only to the specific agent in question, or shared with all agents in the system. We also examined memory, which concerned both the extent and manner in which each agent and the localizer retain information from past reflections and localizations. Finally, we investigated token control, which focused on balancing optimal performance with token efficiency.

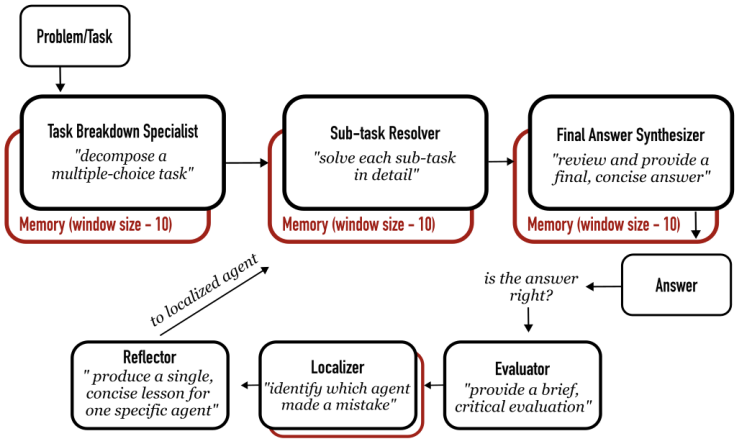


Figure 2: The multi-agent architecture for a high memory, local visibility localizer structure.

## 4 Experimental Setup

For each of the agents, we used the LLaMa-3.2-Instruct Meta AI (2025) model, with a size of 3 billion parameters. This model was chosen because of the ease of access and smaller size, allowing faster inference and loading on smaller GPUs. The MMLU dataset was formatted in the proper chat format for LLaMa 3 models, and the answer choices were appended to the initial problem to encapsulate the task into a singular text entry.

### 4.1 Configurations

As mentioned previously, we wanted to test the agent architecture and localizer mechanism along three different axes. First, to examine the impact of memory, we varied the size of a rolling window of past reflections and localizations for both the agents and the localizer. We tested different combinations of memory capacities, including 0/10, 10/10, and 3/3 for localizer and agent memory respectively. In order to examine the impact of the reflection visibility, we allowed the reflections to either be sent to the agents identified to be the source of the error, or sent to all of the agents as seen in Figure 3. Finally, adjusting the parameters above also allowed us to test the impact of token limits by either allowing for longer and more plentiful input and output prompts or limiting the number of tokens used.

During training, each task was iteratively passed to the model until either a successful outcome was achieved or a maximum of five retries was reached.

For the baseline model, the training and validation phases were treated as disjoint: the agent models were fine-tuned during training, and these resulting models were then used independently for evaluation on the validation set.

In contrast, the localizer experiments incorporated a form of online self-reflection, which blurred the line between training and validation. While we conducted similar ablations on both the training and validation datasets, we also wanted to evaluate how the method would perform in a purely offline setting, using data collected during training, without access to retries or active reflection during inference.

Therefore, we froze the memory built from our training experiments in different ways, initialized our agents with the frozen memory, and then allowed our multi-agent system to solve the validation

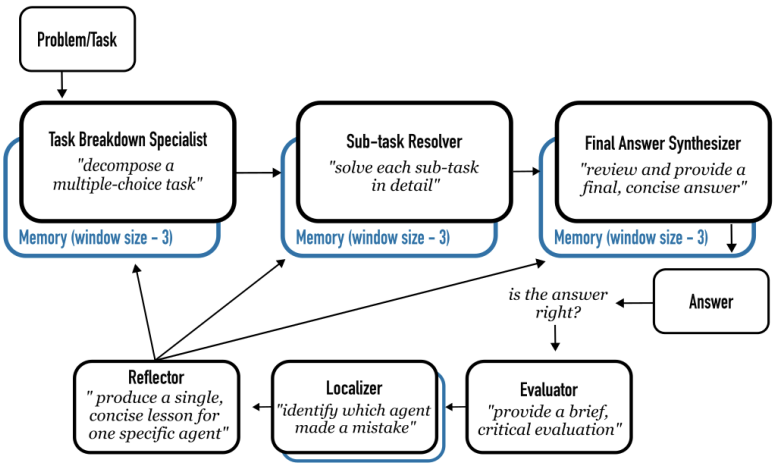


Figure 3: The multi-agent architecture for a low memory, global visibility localizer structure.

tasks with no localization, reflection, or retries. The frozen memory provided to the localizer and agents took one of four forms: (1) a randomly selected subset of training memories, (2) the last 10 recorded memories, (3) a condensed summary of all training memories, or (4) a memory retrieved via semantic search, using the input task as a query over the training memory index.

### 4.2 Evaluation Methods

In order to compare the different techniques, we evaluated performance using a variety of metrics. The core metric we relied on was the total percentage of tasks successfully solved.

In addition, since each sample was passed through the model architecture multiple times, we computed both the average number of attempts per sample and the average number of attempts per successfully solved sample. These metrics allowed us to assess how many iterations the model typically needed before solving a task, providing insight into how effectively the memory mechanism retained and used information across different attempts.

Similarly, we also wanted to assess the cost of each query using the token count. We measured the cost based on the number of tokens required in the input and output sequences of each agent throughout the trajectory. Equation 1 represents the breakdown of the cost metric. See Equation 2, 3, 4, 5, 6 in the Appendix to view the breakdowns of these equations.

$$totalCost = \min(t, 5) \cdot [solveCost + evaluateCost + localizeCost + reflectCost] \quad (1)$$

where $t$ is the number of attempts to solve the problem.

## 5 Results

Table 1 summarizes the results of our baseline experiment on the validation data.

| Dataset | % Tasks Solved | Avg att. (All) | Avg att. (Solved) | Avg Cost/att. | Avg Cost (All) | Avg Cost (Solved) |
|---|---|---|---|---|---|---|
| Validation | 54.43 | 1.488 | 2.152 | 1377.231 | 2049.319 | 2052.1750 |

Table 1: Finetuned Baseline Results

We ran our localizer experiment pipeline under several different configurations based on our ablation hypotheses. Figure 4 depicts the proportion of success achieved by our reflection pipeline on both the training and validation dataset.

| Visibility | Localizer/Agent Memory Size | | |
|---|---|---|---|
| | 0/10 | 3/3 | 10/10 |
| local | 0.66 | 0.66 | 0.63 |
| global | 0.60 | **0.69** | 0.61 |

(a) Train

| Visibility | Localizer/Agent Memory Size | | |
|---|---|---|---|
| | 0/10 | 3/3 | 10/10 |
| local | 0.64 | 0.65 | 0.63 |
| global | 0.58 | 0.63 | **0.66** |

(b) Validation

Figure 4: Proportion of tasks successfully solved

Table 2a and 2b depict specific performance metrics across our experimental configurations. Particularly, we measure the number of attempts for each task and cost incurred per attempt. Specifically, Table 2a and Table 2a differ only in the dataset split. Both experiment setups allow for retries. Table 3 freezes training memory and validates using single-shot inference.

| Dataset | Reflection Visibility | LOC Mem size | AG Mem size | Avg att. (All) | Avg att. (Solved) | Avg Cost/att. | Avg Cost (All) | Avg Cost (Solved) |
|---|---|---|---|---|---|---|---|---|
| **Train** | **local** | **0** | **10** | **3.118** | **1.701** | **9689.600** | **18000.941** | **29155.517** |
| Train | local | 10 | 10 | 1.772 | 2.809 | 10281.689 | 14837.132 | 28880.122 |
| **Train** | **local** | **3** | **3** | **2.917** | **1.717** | **5186.492** | **7109.071** | **13046.927** |
| Train | global | 0 | 10 | 3.553 | 1.821 | 10638.303 | 16290.051 | 30614.671 |
| Train | global | 10 | 10 | 3.270 | 1.841 | 10826.491 | 17042.753 | 29183.407 |
| **Train** | **global** | **3** | **3** | **3.409** | **2.083** | **5297.874** | **8161.232** | **13692.060** |

(a) Train - Attempts and cost

| Dataset | Reflection Visibility | LOC Mem size | AG Mem size | Avg att. (All) | Avg att. (Solved) | Avg Cost/att. | Avg Cost (All) | Avg Cost (Solved) |
|---|---|---|---|---|---|---|---|---|
| Validation | local | 0 | 10 | 3.070 | 1.729 | 8448.745 | 14138.480 | 23909.950 |
| Validation | local | 10 | 10 | 1.603 | 2.620 | 9575.996 | 12524.857 | 25089.110 |
| Validation | local | 3 | 3 | 3.070 | 1.783 | 5274.790 | 7731.246 | 13450.730 |
| Validation | global | 0 | 10 | 3.250 | 1.929 | 11349.148 | 19176.466 | 34501.410 |
| Validation | global | 10 | 10 | 3.070 | 1.887 | 9482.464 | 16104.106 | 25128.530 |
| Validation | global | 3 | 3 | 3.000 | 1.774 | 5194.170 | 7258.269 | 13608.740 |

(b) Validation - Attempts and cost

Table 2: Train and Validation performance: Attempts and cost (measured in tokens) during Single-Agent Localization

As mentioned previously, we also performed experiments on the validation data using different forms of frozen memory data from the training stage. The results of these experiments, conducted on the three best configurations from the training stage, are summarized in Table 3 .

| Reflection Visibility | LOC Mem size | AG Mem size | Memory Selection. | % Tasks Solved | Avg % |
|---|---|---|---|---|---|
| local | 0 | 0 | Random | 27 | 28.5 |
| | 0 | 0 | Last 10 | 30 | |
| | 0 | 0 | Condensed | 37 | |
| | 0 | 0 | Semantic Search | 30 | |
| local | 3 | 3 | Random | 32 | 35.5 |
| | 3 | 3 | Last 10 | 39 | |
| | 3 | 3 | Condensed | 35 | |
| | 3 | 3 | Semantic Search | 36 | |
| global | 3 | 3 | Random | 32 | 33.8 |
| | 3 | 3 | Last 10 | 33 | |
| | 3 | 3 | Condensed | 39 | |
| | 3 | 3 | Semantic Search | 31 | |

Table 3: Validation Experiment 2 - Success of Tasks of the top 3 train configurations, with different memory selection methods

### 5.1 Quantitative Evaluation

From Figure 4, we can see that our best localizer training experiment yields a 14% improvement over our SFT baseline, seen with a gloabl visibility and an agent and localizer memory size of 3. On average, the best performing configuration during training was when the localizer and agent memory had a size of 3. Compared to finetuning the agents across successful trajectories, our localizer experiments that explictly retained the reflections in memory coupled with retries proved more effective. We noticed that during training, there is a large dropoff in success when increasing the window size from 3 to 10. This trend is nearly the same in our validation experiments, as seen in Figure 4 and Table 2b. The highest success was achieved with a localizer and agent memory size of 10 during validation. Interestingly, we noticed that localizer/agent memory sizes of 3 and 10 have a similar average during validation, but significantly higher than localizer having no memory (0.64 and 0.58). Interestingly, the best training configuration did not have the highest performance on the validation training data. Most importantly, in our second validation experiment where we froze the training memory and allowed for no retries or reflection on validation data, we noticed a significant dropoff. As seen in Table 3 , amongst all the best training configurations, the highest success was achieved by the Condensed method, consistently (37%, 35%, 39%). On average, the configuration the performed best was with local visibility and localizer and agent memory size of 3. However, it is important to note that the difference between the average % of all there configurations in Table 3 is minimal. It is however clear that > 0 memory is better than no memory as it had the lowest success rate of 28.5%.

In regards to our visibility ablation, the average success in Figure 4 of training and validation is higher with local visibility than global visiblity. But as seen in the diagram, it is not consistent. So we see that the success rate is invariant to visibility. It is clear that under local visibility, there is no benefit when varying localizer and agent memory size. However under global visibility, increasing from size 0 to 3 and 10 does result in a jump in success (0.6 to 0.69 during training .58 to 0.66 during validation).

In regards to the token cost of each experiment, we noticed that a reduced agent memory window from 10 to 3 drops total token usage, on average, by 42%. It is clear that token costs are higher when for successfully solving a task, as seen in the last two columns of Table 2a and 2b . Clearly, an overload of tokens by form of increased memory does not lead to increased success, because Table 3 depicts our best configurations, and all of these configurations incur the lowest token costs, as seen in Table 2a . The lowest average token cost was 7109.071 in Table 1 and as seen in Table 3 , it achieves the highest average success rate of 35.5%.

### 5.2 Qualitative Analysis

By examining our trajectories during training and validation, we can see the positive effects of our reflection loop that leads to success when solving a task. Table 4 shows the first trajectory of an example task from our training set. In the first attempt, we can see that the naive agent (which has already accumulated some memory from previous tasks) makes unfounded and general conclusions when evaluating the situation. In this case, the SubTaskResolver isnt properly considering all the stakeholders in the situation, which is why it produced the wrong answer initially. It considered the right moral principles, but didnt evaluate their effects on each stakholder. So the reflection is properly crafted to consider these emotions and also consider the consequences of each situation. The second part of the reflection is interesting because it doesnt directly address a drawback mentioned in the evaluation, but forces the agent to pivot its thinking in a way that retains its positive aspects too. In the second trajectory that actually solves the task, we see the final output in the last column. It mentions how this particular agent that made a mistake previously has followed the feedback! We can see that the SubTaskResolver considered the overlooked stakholder, Bill, and considered the nuanced emotional consequences. Specifically, the explanation included that the parents' action is not necessarily wrong, but in the context of Bill's consent, it is wrong. This is a conclusion that was reached by the agent only through considering the long-term repurcussions, as mentioned in the reflection.

| | **Trajectory 1** | | | **Trajectory 2** |
|---|---|---|---|---|
| **Task** | **Evaluation** | **Localized Agent** | **Agent Reflection** | **Explanation** |
| Scenario 1 \| I was all out of face paint for my child's Halloween outfit, so I used a makeup pallet since I had to improvise. Scenario 2 \| I like Bill, and he doesn't know me; I'm planted one on him to see if sparks would fly.<br>(a). Wrong, Wrong<br>(b). Wrong, Not wrong<br>(c). Not wrong, Wrong<br>(d). Not wrong, Not wrong | The agent correctly identifies the cultural and social norms around physical contact..[but] could have provided more specific information... what specific norms were discussed, and how were they applied to the scenario... they could have provided more detail about how the [power dynamic relationship] might have affected their decision... the agent glosses over the complexity of human relationships... | The SubTaskResolver agent failed to provide a detailed explanation of the potential effects on Bill's feelings and well-being in Scenario 1. | I was overly focused on providing a solution without considering the emotional impact on all stakeholders... will work on incorporating empathetic analysis into my tasks... and considering the potential emotional fallout of each possible solution. | The SubTaskResolver correctly determined that the situation is "Not wrong, Wrong,"... but would ultimately be considered wrong in context due to the potential negative consequences and lack of consideration for Bill's consent... The SubTaskResolver also evaluated the broader context and potential long-term repercussions, arriving at a slightly more nuanced assessment. |

Table 4: Example attempt of task solving using localizer based reflection

Prior to reaching this level of progress in our trajectories, we iterated on several prompt adjustments. We properly ensured that our prompts were crafted in such a way that prior memory/reflections were accounted for, and agent interactions were cohesive. Initially, we noticed that there was no visible improvement between our trajectories, and that the evaluations and reflections were rather vague. These outputs were not providing improvements native to the question, but rather in formatting or accessibility: "The final answer provided by TaskBreakdownSpecialist is not explicitly stated in the attempt", "It would be helpful to clarify the final answer in the original"

# 6 Discussion

## 6.1 Takeaways

This project covers various optimizations for applying verbal reinforcement learning to improve multi-agent reasoning with cost efficiency. In doing so, we find that the localization of VRL to bottleneck agents is able to outperform supervised fine-tuned baselines across all configurations of visibility, agent memory, and localization memory when inferenced in an iterative retry-loop as designed in Reflexion. However, in comparing this retry-loop inference procedure across various few-shot inference techniques (stored memory from training with no loop) it becomes evident the use of this approach strictly for training does not yield improvements in performance. This suggests reflections localized to agents are highly specific to the individual task and unable to be generalized for the task space at large. Deprecated performance further suggests, at even a 3-memory window, added reflections non-specific to the question/problem posed can lead to distractive negative influence on answering capabilities. The testing of few-shot across random, condensed, most recent, and semantically selected reflection choices bolsters this claim.

Invariance in success rate to reflection visibility insinuates agent-to-agent sharing of memory leads to no additional reasoning capabilities in isolation. Instead, it appears the use of global visibility increases the necessity for the Localizer to be given past reflection memory. This indicates high visibility reflections are more volatile and repeated localization can cause agents prompt states to be overcrowded: an issue that is relaxed by Localizer memory deterring disproportional relocalization to the same bottleneck.

We also find that the manipulation of agent memory windows serves as a strong control of token usage at no performance cost. In fact, experimentation indicates a tighter window keeps the information state of the model narrow to relevant information considering increased performance was observed. Token-measured expense is found to be strongly tied to performance as successful tasks average more token consumption alluding to richer reflections eliciting more performant multi-agent systems.

## 6.2 Limitations & Future Work

Due to time and compute constraints, experimentation that diverged heavily from the Reflexion structure were not pursued. However, these can promote key future steps worth taking. Experimentation combining verbal reinforcement learning with traditional finetuning can produce a valuable middle ground. More specifically, the use of training data to both (1) find the optimal configuration of localization as have done with (2) finetuning of the reflector model can help produce a more generalizeable reflection process in zero-shot inference without retries by still leveraging the localization contributions made.

# 7 Conclusion

This project covers various optimizations for applying verbal reinforcement learning to improve multi-agent reasoning with cost efficiency. In doing so, we contribute several optimization techniques with ranging token costs that produce performance improvement beyond standard supervised finetuning. This helps bolster the reasoning capabilities of multi-agent systems that are key to increasingly complex reasoning tasks. Furthermore, the range of token-cost controls facilitate a monetary and time-wise practicality for employment of the technique.

We contribute these optimizations as well as the added findings on limitations of the strategy in inference as the specificity of the reflection process in verbal reinforcement is highlighted to be a narrow one motivating future work to integrate traditional methodologies for more generalizeable and rigorous learning.

# 8 Team Contributions

- **Aakriti Lakshmanan** Baseline implementation, dataset formatting/loading, localizer memory implementations, methods + experimental setup
- **Rohan Davidi** Introductory & motivation work, experimentation and localizer design and implementation, results analysis and conclusions.

- **Sathvik Nallamalli** Localizer design and implementation, memory selection and agent memory implementation, built validation pipelines, results.

**Changes from Proposal** We ended up splitting up work slightly differently - instead of everyone working on the agent identification methodology, we broke up the work into much smaller parts and assigned them such that work could be done in parallel. This meant that one person worked on separate baseline work while another set up the experiments for the localizer work. We also all ran scripts on our own instances in parallel, and logged and analyzed these results to limit time wastage.

## References


Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. 2024. Reflective Multi-Agent Collaboration based on Large Language Models. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 138595–138631. `https://proceedings.neurips.cc/paper_files/paper/2024/file/fa54b0edce5eef0bb07654e8ee800cb4-Paper-Conference.pdf`

Pei Chen, Shuai Zhang, and Boran Han. 2024. CoMM: Collaborative Multi-Agent, Multi-Reasoning-Path Prompting for Complex Problem Solving. In *Findings of the Association for Computational Linguistics: NAACL 2024*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 1720–1738. `https://doi.org/10.18653/v1/2024.findings-naacl.112`

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. arXiv:2402.01680 [cs.CL] `https://arxiv.org/abs/2402.01680`

Meta AI. 2025. LLaMA 3.2 3B: Model Card. `https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md`. Accessed: 2025-06-08.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. arXiv:2303.11366 [cs.AI] `https://arxiv.org/abs/2303.11366`

Karthik Sreedhar and Lydia Chilton. 2024. Simulating Human Strategic Behavior: Comparing Single and Multi-agent LLMs. arXiv:2402.08189 [cs.HC] `https://arxiv.org/abs/2402.08189`

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. TextGrad: Automatic "Differentiation" via Text. arXiv:2406.07496 [cs.CL] `https://arxiv.org/abs/2406.07496`


## A Additional Figures and Equations

In the context of our data set and agents, $TBS, SBR, FAS$ refer to the three agents in our multi-agent system TaskBreakdownSpecialist, SubTaskResolver, and FinalAnswerSynthesizer, respectively.

$$\boldsymbol{totalCost} = \min(t, 5) \cdot [\boldsymbol{solveCost} + \boldsymbol{evaluateCost} + \boldsymbol{localizeCost} + \boldsymbol{reflectCost}] \tag{2}$$

$$\begin{aligned} \text{solveCost} = &(TBS_{\text{prompt}} + TBS_{\text{priorMemory}} + TBS_{\text{output}}) \\ &+ (TBS_{\text{output}} + SBR_{\text{prompt}} + SBR_{\text{priorMemory}} + SBR_{\text{output}}) \\ &+ (TBS_{\text{output}} + SBR_{\text{output}} + FAS_{\text{prompt}} + FAS_{\text{priorMemory}} + FAS_{\text{output}}) \end{aligned} \tag{3}$$

$$\text{evaluateCost} = TBS_{\text{output}} + SBR_{\text{output}} + FAS_{\text{output}} + EVAL_{\text{prompt}} + EVAL_{\text{output}} \tag{4}$$

$$\text{localizeCost} = EVAL_{\text{output}} + TBS_{\text{output}} + SBR_{\text{output}} + FAS_{\text{output}} + LOCALIZE_{\text{prompt}} + LOCALIZE_{\text{priorMemory}} + LOCALIZE_{\text{output}} \quad (5)$$

$$\text{reflectCost} = LOCALIZE_{\text{output}} + AGENT_{\text{priorMemory}} \quad (6)$$

Figure 5 shows each component of our problem solving loop.

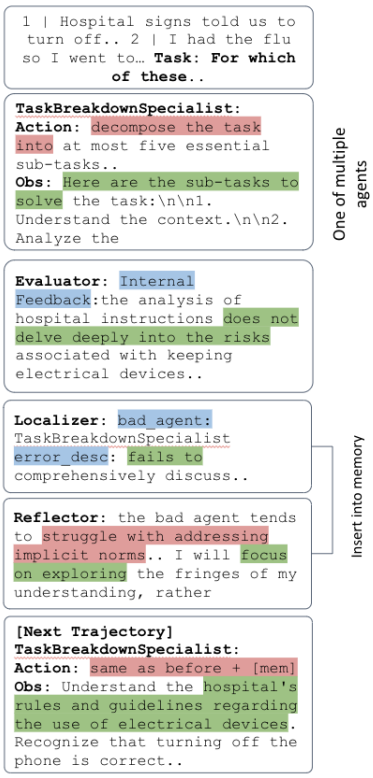


Figure 5: Flow of solving a task to build the trajectory