Conformal Agent Error Attribution

Naihe Feng*
Dalhousie University
Halifax, NS, Canada
nfeng@dal.ca

Yi Sui*
Layer 6 AI
Toronto, ON, Canada
amy@layer6.ai

Shiyi Hou
Layer 6 AI
Toronto, ON, Canada
gloria@layer6.ai

Ga Wu
Dalhousie University
Halifax, NS, Canada
ga.wu@dal.ca

Jesse C. Cresswell
Layer 6 AI
Toronto, ON, Canada
jesse@layer6.ai

Abstract

When multi-agent systems (MAS) fail, identifying where the decisive error occurred is the first step for automated recovery to an earlier state. Error attribution remains a fundamental challenge due to the long interaction traces that large language model-based MAS generate. This paper presents a framework for error attribution based on conformal prediction (CP) which provides finite-sample, distribution-free coverage guarantees. We introduce new algorithms for filtration-based CP designed for sequential data such as agent trajectories. Unlike existing CP algorithms, our approach predicts sets that are contiguous sequences to enable efficient recovery and debugging. We verify our theoretical guarantees on a variety of agents and datasets, show that errors can be precisely isolated, then use prediction sets to rollback MAS to correct their own errors. Our overall approach is model-agnostic, and offers a principled uncertainty layer for MAS error attribution.

1 Introduction



Figure 1: Conformal agent error attribution isolates the decisive error in a failed MAS trajectory within a conformal prediction set, providing statistical guarantees of coverage.

Advances in large language models (LLMs) have driven the widespread adoption of multi-agent systems (MAS) for complex tasks requiring decomposition, coordination, and tool use [11], with strong empirical performance in domains such as software engineering [15, 16], scientific discovery [9], and financial decision making [31]. However, the increased system complexity and rich interactions in MAS make them prone to errors from incorrect intermediate decisions, miscoordination among agents, and long-horizon dependencies [5, 20]. While detecting *overall* task failure is often straightforward, understanding *why and where* a failure originated remains challenging yet critical for debugging, and self-correction. Identifying the decision step that constitutes the decisive error point has emerged as a central challenge for improving MAS.

Most existing MAS error attribution approaches, including naive LLM-as-a-judge methods [33], structured reasoning pipelines [14, 34, 32], and fine-tuned attribution models [18], ultimately produce a point prediction, committing to a single responsible step. In practice, point predictions provide limited actionable insight for practitioners as they offer no principled form of uncertainty quantification to assess reliability, undermining the trustworthiness of error attribution systems [26]. Conformal

*Equal Contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

prediction (CP) offers a promising direction for addressing this limitation through the generation of *prediction sets*. CP enables reliable decision-making under uncertainty by providing statistical guarantees across a range of applications [1, 7].

Motivated by these developments, we propose an uncertainty-aware framework for error attribution in MAS based on CP, which provides finite-sample, distribution-free coverage guarantees. Rather than predicting a single step, our methods identify a localized region of the execution trace that is guaranteed to contain the decisive error at a user-specified confidence level (see Figure 1). We introduce novel methods for filtration-based CP which are adapted to sequential data structures, like agent trajectories. Unlike existing CP approaches that produce arbitrary sets, our methods produce *contiguous* sets, aligning with the inherent ordinal structure of sequential data. Finally, we use conformal sets to rollback the MAS to before the decisive error, allowing the agent to restart and fix its own mistakes. Our approach is model-agnostic and can wrap existing black-box attribution scores, while providing a principled uncertainty layer for error attribution in real-world MAS.

2 Background & Related Work

2.1 Post-hoc Multi-agent System Error Attribution

Recent work on error attribution in MAS has primarily studied post-hoc localization of errors using execution traces. Early approaches used *naive LLM-as-a-judge*, where a single LLM directly predicts the responsible step from a failed trace [33]. Subsequent work improved attribution quality through more sophisticated LLM pipelines, including *context engineering* as well as multi-LLM frameworks. For example, ECHO [3] improves attribution by organizing long traces into hierarchical contexts and aggregating evaluations via consensus, while RAFFLES [34] employs a multi-turn, multi-LLM architecture that iteratively proposes and critiques candidate error steps. Additionally, CORRECT [32] incorporates retrieval to localize the error step based on similar events. A complementary line of work *fine-tunes specialized LLM judges* for error attribution. In particular, AEGIS [18] constructs large-scale labeled failure traces via controlled error injection for fine-tuning LLMs on the task.

In our experiments we compare the efficacy of these three main classes of evaluators: naive, context-engineered, and fine-tuned LLM judges. We note that all of the above research assumes a single decisive error, whereas in practical MAS applications small errors may accumulate into large ones. Due to the lack of labeled datasets with more nuanced error definitions, we follow current work and focus on the decisive error setting.

2.2 Conformal Prediction

For a classification problem where inputs $x \in \mathcal{X}$ and ground-truth values $y^* \in \mathcal{Y} = [\ell] := (1, \ldots, \ell)$ are drawn jointly from a distribution $(x, y^*) \sim \mathbb{P}$, CP first calibrates a threshold $\hat{q}$ from a set of held-out data. Then for a new datapoint x_{n+1}, CP outputs a set of classes $C(x_{n+1}; \hat{q}) \subseteq \mathcal{Y}$ which contains y^* with high probability $\mathbb{P}[y^*_{n+1} \in C(x_{n+1}; \hat{q})] \geq 1 - \alpha$. This *coverage* guarantee is distribution-free and valid in finite samples, while also allowing the user to set their own error tolerance α [28, 27].

To perform CP, one first defines a *conformal score* function $S : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, which should take smaller values when $y = y^*$ is the correct label for x. In practice, $S(x, y)$ often leverages the predictions of a pre-trained classification model $f : \mathcal{X} \to \mathcal{Y}$. Using a set of n calibration datapoints, CP computes the scores $\{S_i\}_{i=1}^n = \{S(x_i, y_i^*)\}_{i=1}^n$, and finds the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile which is set as the threshold $\hat{q}$. Then prediction sets can be generated by including all classes for which the score is greater than $\hat{q}$, $C(x_{n+1}; \hat{q}) = \{y \in \mathcal{Y} \mid S(x_{n+1}, y) \leq \hat{q}\}$. When x_{n+1} is exchangeable with the calibration data, the coverage guarantee is valid. Exchangeability is a mild assumption that automatically holds when data is i.i.d., and hence is reasonable for many machine learning contexts, including the agent error attribution task as we show in our experiments. At equal coverage levels $1 - \alpha$, smaller prediction sets are considered more useful both for uncertainty quantification over the predictions of f_θ [25, 2, 17], and in downstream tasks [7, 8].

2.3 Conformal Prediction for Sequential Data

Another common setting is where data is sequential, $x = (c_1, \ldots, c_\ell)$, with variable length ℓ, where the ground truth $y^* \subset x$ is a subset of elements. Following Kuwahara et al. [19], the principles of

CP can be used to calibrate a threshold $\hat{q}$, and predict a subset $C(x_{n+1}; \hat{q}) \subseteq x_{n+1}$ which retains the ground truth elements with high probability, $\mathbb{P}[y^*_{n+1} \subseteq C(x_{n+1}; \hat{q})] \geq 1 - \alpha$. In some settings, y^* will consist of multiple elements, and the predicted set $C(x_{n+1}; \hat{q})$ need not be contiguous. For agent error attribution we take x to be the agent's trajectory, and y^* to be the single decisive error—one of the c_i. As we will discuss, for downstream applications including automated rollbacks of the agent's state it is desirable to predict sets of *consecutive* elements, rather than arbitrary subsets. Hence, we develop novel CP algorithms that satisfy a coverage guarantee using contiguous prediction sets,

$$\mathbb{P}[y^*_{n+1} \in C(x_{n+1}; \hat{q})] \geq 1 - \alpha, \quad C(x_{n+1}; \hat{q}) = (c_j, ..., c_k). \tag{1}$$

The only existing CP algorithms that produce contiguous sets were designed for hierarchical classification [22]. We describe one such algorithm in Section 3.1.2 and adapt it for sequential data.

3 Conformal Agent Error Attribution

For the remainder of this work we take $x = (c_1, \ldots, c_\ell)$ to be an agent trajectory which has failed to complete the desired task. Each step c_j can contain any available information such as the environment's state, action taken, and observed response. One of the steps $y^* \in x$ is labeled as the decisive error—the earliest error that the MAS cannot recover from. The aim is to produce a prediction set $C(x_{n+1}) \subseteq x_{n+1}$ that gives valid coverage, where smaller sets are preferred.

Applying CP to agent error attribution requires two components: an algorithm which takes a calibration dataset and generates a prediction set for x_{n+1} with valid coverage; and a scoring function $g(C(x))$ acting on sets of steps, which quantifies the likelihood that $y^* \in C(x)$. We design these two components separately so that they are interchangeable, and discuss the pros and cons of each option.

3.1 Conformal Algorithms for Agent Error Attribution

3.1.1 Vanilla Conformal Prediction

The simplest approach is to ignore the sequential nature of x and treat all steps as unordered classes in an ℓ-way classification task. We will write $S_{\text{VCP}} = 1 - g$ for the conformal score function, and $C_{\text{VCP}}(x_{n+1}; \hat{q}) = \{c_i \in x_{n+1} \mid S_{\text{VCP}}(x_{n+1}, c_i) \leq \hat{q}\}$ for prediction sets generated by Vanilla CP (VCP). Prediction requires iterating over every step in the trajectory using ℓ evaluations of g, and does not produce contiguous sets.

3.1.2 Leaf-to-Root Tree Traversal

To produce contiguous sets, we can adapt algorithms for hierarchical classification by mapping agent trajectories x onto a binary tree $\mathcal{T}$ as depicted in Figure 2, with root node $v_1 = [\ell]$, and leaf nodes $v_\ell, \ldots, v_{2\ell-1}$ as individual steps $c_1, \ldots, c_\ell$. CP is conducted by traversing the tree from leaf to root following the CRSVP algorithm [22] described in full detail in Section B. For each test datapoint, CRSVP outputs one node of the tree as the prediction set which is always a contiguous set, and guarantees the lower bound on coverage in Equation (1).

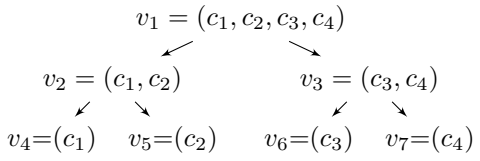

Figure 2: An example binary tree $\mathcal{T}$ representing an agent trajectory x consisting of four steps $c_1, ..., c_4$. Contiguous prediction sets $C(x_{n+1})$ will consist of a single node v_i.

CRSVP lacks an upper bound on coverage, uses ℓ evaluations of g for prediction, and produces inflexible sets following the tree's splits. For example, in Figure 2 the middle steps c_2 and c_3 can only be predicted together in the trivial case where all steps are predicted (v_1). VCP and CRSVP serve as baselines in our experiments. The following novel algorithms improve on their limitations.

3.1.3 Left (Right) Filtration

Viewing a trajectory $x = (c_1, \ldots, c_\ell)$ as a sequence, Left Filtration (LF) progressively removes steps from x starting on the left with c_1 until the remaining subsequence scores below a calibrated threshold, returning a ***suffix*** of the full sequence (see Figure 3).

We assume access to a scoring function g_{LF} which scores subintervals $c_{j:k} := (c_j, \ldots, c_k) \subseteq x$ for their likelihood to contain y^*, imposing the boundary conditions $g_{\text{LF}}(\varnothing) = 0$, since the empty interval cannot contain y^*, and $g_{\text{LF}}(x) = \infty$ since $y^* \in x$. We will use j^* as the index where y^* appears, so $c_{j^*} = y^*$. Finally, it is desirable, but not mandatory, to have g_{LF} obey a monotonicity condition:

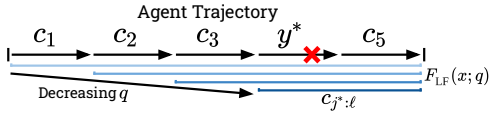

Figure 3: Left filtering with $F_{\text{LF}}(x;q)$ progressively removes steps from the left as q decreases. The smallest q which retains the decisive error y^* is used as the conformal score $S_{\text{LF}}(x, y^*)$.

$$c_{b:c} \subseteq c_{a:d} \implies g_{\text{LF}}(c_{b:c}) \leq g_{\text{LF}}(c_{a:d}). \tag{2}$$

Next, we define a filtering function which returns the longest suffix that has a low enough score g_{LF}. Formally, we write the set of suffixes as $\mathcal{I}_{\text{LF}} := \{c_{j:\ell}\}_{j=1}^{\ell+1}$, with the conventions for subintervals that $c_{j:j} = c_j$, and $c_{j:k} = \varnothing$ when $j > k$. With these conventions, the left-filtering function is

$$F_{\text{LF}}(x;q) := \underset{c_{j:\ell} \in \mathcal{I}_{\text{LF}}}{\arg\max} \left(|c_{j:\ell}| \mid g_{\text{LF}}(c_{j:\ell}) \leq q \right), \tag{3}$$

where $q \in \mathbb{R}^+$, and we note that $F_{\text{LF}}(x;\infty) = x$. Since the suffixes in $\mathcal{I}_{\text{LF}}$ are nested, $F_{\text{LF}}(x;q)$ also satisfies a nesting property demonstrating that more steps are filtered out as q decreases. (All formal proofs are given in Section A.2.)

Lemma 3.1. *For any x and thresholds $0 \leq q_1 \leq q_2$, $F_{LF}(x;q_1) \subseteq F_{LF}(x;q_2)$.*

Proof sketch. Suffixes themselves are nested. When q_1 increases to q_2, the set of valid suffixes with $g_{\text{LF}}(c_{j:\ell}) \leq q$ can only grow, and F_{LF} returns the largest valid suffix. □

For a trajectory x, the conformal score is effectively the smallest threshold q where y^* is not filtered out (see Figure 3). More formally we define

$$S_{\text{LF}}(x, y^*) := \inf \left(q \in \mathbb{R}^+ \mid y^* \in F_{\text{LF}}(x;q) \right). \tag{4}$$

Although this definition involves two optimizations, S_{LF} is designed so that its computation greatly simplifies in practice. When g_{LF} obeys monotonicity (Equation (2)), S_{LF} is simply the value of g_{LF} on the suffix that starts at y^*.

Proposition 3.2. *When g_{LF} obeys monotonicity such that $c_{b:c} \subseteq c_{a:d} \implies g_{LF}(c_{b:c}) \leq g_{LF}(c_{a:d})$, then $S_{LF}(x, y^*) = g_{LF}(c_{j^*:\ell})$.*

Proof sketch. $S_{\text{LF}}(x, y^*)$ optimizes over q where $F_{\text{LF}}(x;q)$ returns a suffix at least as large as $c_{j^*:\ell}$. Due to monotonicity, suffixes larger than $c_{j^*:\ell}$ cannot have scores less than $g_{\text{LF}}(c_{j^*:\ell})$, so $g_{\text{LF}}(c_{j^*:\ell})$ is the smallest valid threshold. □

The LF conformal algorithm computes S_{LF} for each calibration datapoint, and sets the conformal threshold $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile. For a test datapoint we predict $C_{\text{LF}}(x_{n+1};\hat{q}) = F_{\text{LF}}(x_{n+1};\hat{q})$, which is the longest suffix $c_{j:\ell}$ such that $g_{\text{LF}}(c_{j:\ell}) \leq \hat{q}$. A shorter suffix is preferable since it better isolates the decisive error. This algorithm satisfies Equation (1) for any scoring function g_{LF} as defined above. Before proving this coverage guarantee, we present a lemma to assist:

Lemma 3.3. *For a fixed $\hat{q}$, we have $S_{LF}(x, y^*) \leq \hat{q} \iff y^* \in F_{LF}(x;\hat{q})$.*

Proof sketch. The infimum in $S_{\text{LF}}(x, y^*)$ is always achieved at $q_{\text{min}} = \min\{g_{\text{LF}}(c_{j:\ell}) \mid j \leq j^*\}$, and $y^* \in F_{\text{LF}}(x;q_{\text{min}})$. Because F_{LF} is nested in q (Lemma 3.1), increasing q_{min} to $\hat{q}$ maintains $y^* \in F_{\text{LF}}(x;\hat{q})$. □

With these facts established, we can prove the coverage guarantee of Equation (1) using a standard conformal argument.

Theorem 3.4. *Suppose $\{(x_i, y_i^*)\}_{i=1}^n$ and (x_{n+1}, y_{n+1}^*) are exchangeable. Given a scoring function g_{LF} acting on subintervals of the x_i, define the conformal score $S_{LF}(x_i, y_i^*)$ as in Equation* (4)*. Let $\hat{q}$ be the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of conformal scores $\{S_i\}_{i=1}^n = \{S_{LF}(x_i, y_i^*)\}_{i=1}^n$. Then prediction sets constructed as $C_{LF}(x_{n+1};\hat{q}) = F_{LF}(x_{n+1};\hat{q})$ satisfy $1 - \alpha \leq \mathbb{P}[y_{n+1}^* \in C_{LF}(x_{n+1};\hat{q})] < 1 - \alpha + \frac{1}{n+1}$.*

Proof. Since the fixed function S_{LF} is applied to each element of the exchangeable sequence $(x_1, y_1^*), \ldots, (x_{n+1}, y_{n+1}^*)$, the resulting sequence $S_1, \ldots, S_{n+1}$ is also exchangeable. We assume for simplicity that the scores are non-degenerate, because noise can easily be added to break ties. Let $S_{(1)} \leq S_{(2)} \leq \cdots \leq S_{(n)}$ be the order statistics of the calibration scores so that the empirical quantile $\hat{q}$ can be defined as $S_{(k)}$, where $k = \lceil (n+1)(1-\alpha) \rceil$ (assuming $n \geq \frac{1}{\alpha} - 1$ to ensure $k \leq n$). The event $S_{n+1} \leq \hat{q}$ occurs if and only if S_{n+1} is among the k smallest scores overall. From exchangeability, all orderings are equally likely, so $\mathbb{P}[S_{n+1} \leq \hat{q}] = \frac{k}{n+1}$. By Lemma 3.3 this event is equivalent to $y^* \in F_{\text{LF}}(x_{n+1}; \hat{q})$. Hence, $\mathbb{P}[y^* \in F_{\text{LF}}(x_{n+1}; \hat{q})] = \frac{\lceil (n+1)(1-\alpha) \rceil}{n+1} \geq 1 - \alpha$. For the upper bound, we simply note that $\frac{\lceil (n+1)(1-\alpha) \rceil}{n+1} < \frac{1+(n+1)(1-\alpha)}{n+1} = 1 - \alpha + \frac{1}{n+1}$. □

Beyond generating contiguous prediction sets, LF uses fewer scoring function evaluations on average for inference when g_{LF} is monotonic. LF starts with the first suffix $c_{\ell:\ell}$ and evaluates each longer suffix $c_{j:\ell}$ until one with $g_{\text{LF}}(c_{j:\ell}) > q$ is found, then returns $c_{j+1:\ell}$. When errors are evenly distributed over the length ℓ, the average number of evaluations with a strong g_{LF} is only $\frac{\ell+1}{2}$.

Of course, there is nothing special about filtering from the left; Right Filtration (RF) can easily be defined which filters from the right, returning a *prefix* of x. Using a similarly defined scoring function g_{RF} we write the relevant prefixes as $\mathcal{I}_{\text{RF}} := \{c_{1:k}\}_{k=0}^{\ell}$ and define the right-filtering function $F_{\text{RF}}(x; q) := \arg\max_{c_{1:k} \in \mathcal{I}_{\text{RF}}} \left(|c_{1:k}| \mid g_{\text{RF}}(c_{1:k}) \leq q \right)$. Based on these definitions, the conformal score is $S_{\text{RF}}(x, y^*) := \inf \left(q \in \mathbb{R}^+ \mid y^* \in F_{\text{RF}}(x; q) \right)$, and prediction sets are generated as $C_{\text{RF}}(x_{n+1}; \hat{q}) = F_{\text{RF}}(x_{n+1}; \hat{q})$. RF gives coverage as a straightforward extension of Theorem 3.4 and is advantageous over LF if agents tend to fail earlier in their trajectory rather than later.

3.1.4 Two-Way Filtration

Filtering from one direction has downsides; when the decisive error is at the start (end), the suffix (prefix) which covers the ground truth will contain the entire trajectory. Moreover, when the decisive error is near the middle, neither LF nor RF can isolate it. However, these limitations can be avoided by building on LF and RF with Two-Way Filtration (TWF). Bidirectional filtering allows more precise isolation of the decisive error in principle, regardless of where in the trajectory it occurs.

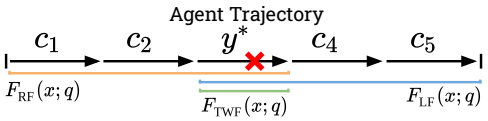

Figure 4: Two-way filtering with $F_{\text{TWF}}(x; q)$ uses the intersection of filtering from the right and left. The smallest q which retains the decisive error y^* is the conformal score $S_{\text{TWF}}(x, y^*)$.

There are many possible ways to define a bidirectional filtration. We present a version that uses the *intersection* of left and right filtered subintervals. Using g_{LF} and g_{RF} as above, we define the two-way filtering function as

$$F_{\text{TWF}}(x; q) := F_{\text{LF}}(x; q) \cap F_{\text{RF}}(x; q). \tag{5}$$

This function finds the longest suffix and prefix that each score at most q, and takes their intersection. Note that this function still satisfies $F_{\text{TWF}}(x; \infty) = x$, as well as nesting: (All formal proofs are given in Section A.3)

Lemma 3.5. *For any x and thresholds $0 \leq q_1 \leq q_2$, $F_{TWF}(x; q_1) \subseteq F_{TWF}(x; q_2)$.*

Proof sketch. The result follows from nesting for L/RF, and set inclusion under intersection: for any sets A, B, C, D, if $A \subseteq B$ and $C \subseteq D$, then $A \cap C \subseteq B \cap D$. □

On top of this we can define the conformal score function

$$S_{\text{TWF}}(x, y^*) := \inf \left(q \in \mathbb{R}^+ \mid y^* \in F_{\text{TWF}}(x; q) \right), \tag{6}$$

which operates the same way as before, effectively finding the smallest threshold q under which two-way filtering retains the decisive error. Although S_{TWF} appears to involve three optimizations, its computation can be simplified to only two scoring function evaluations:

Proposition 3.6. *$S_{TWF}(x, y^*)$ as defined in Equation (6) can be expressed as*

$$S_{TWF}(x, y^*) = \max \left(S_{LF}(x, y^*), S_{RF}(x, y^*) \right). \tag{7}$$

When g_{LF} and g_{RF} obey the monotonicity condition in Equation (2), S_{TWF} further simplifies to

$$S_{TWF}(x, y^*) = \max \left(g_{LF}(c_{j^*:\ell}), g_{RF}(c_{1:j^*}) \right). \tag{8}$$

Proof sketch. q is a valid threshold ($y^* \in F_{\text{TWF}}(x; q)$) only when y^* is included by both F_{LF} and F_{RF}, which means q is sufficiently high for both S_{LF} and S_{RF}. We must use the higher of these two thresholds, giving Equation (7). Equation (8) follows from substituting in Proposition 3.2. □

Like for LF, the TWF conformal algorithm computes S_{TWF} for each calibration datapoint, sets the conformal threshold $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile, and predicts $C_{\text{TWF}}(x_{n+1}; \hat{q}) = F_{\text{TWF}}(x_{n+1}; \hat{q})$ for any test datapoint x_{n+1}. $F_{\text{TWF}}(x_{n+1}; \hat{q})$ will tend to be a short subinterval when g_{LF} and g_{RF} both narrow in on the same steps. This algorithm satisfies Equation (1) with a theorem and proof similar to Theorem 3.4 (see Theorem A.2). The core prerequisite is the analogue of Lemma 3.3:

Lemma 3.7. *For a fixed $\hat{q}$, we have $S_{TWF}(x, y^*) \leq \hat{q} \iff y^* \in F_{TWF}(x; \hat{q})$.*

Proof sketch. From the use of intersection in $F_{\text{TWF}}(x; \hat{q})$ (Equation (5)), and the result of Lemma 3.3, $y^* \in F_{\text{TWF}}(x, \hat{q}) \iff S_{\text{LF}}(x, y^*) \leq \hat{q} \wedge S_{\text{RF}}(x, y^*) \leq \hat{q}$. Equivalently we write $\max\left(S_{\text{LF}}(x, y^*), S_{\text{RF}}(x, y^*)\right) \leq \hat{q}$, which is the same as $S_{\text{TWF}}(x, y^*) \leq \hat{q}$ (Proposition 3.6). □

While we described these algorithms using the language of agent trajectories, they can equally be applied to any type of sequential data x with ground truth $y^* \in x$, for example electronic health records where x is a sequence of vitals measurements, and y^* is the first warning sign that should trigger medical intervention [24].

The five conformal algorithms we compare are summarized in Table 1, showing which generate contiguous prediction sets, the number of function evaluations (NFE) of g needed for each test datapoint in the case where g is strong and errors are uniformly distributed, and whether an upper bound on coverage is guaranteed.

Table 1: Conformal Algorithm Properties

Method	Contiguous	Inference NFE	Cov. Upper Bound
VCP	×	ℓ	✓
CRSVP	✓	ℓ	×
L/RF	✓	$\approx \frac{1}{2}(\ell + 1)$	✓
TWF	✓	ℓ	✓

3.2 Scoring Functions for Agent Error Attribution

Each of the preceding conformal algorithms requires a scoring function g which estimates how likely a step, or set of steps, is to contain the decisive error. Since MAS traces are primarily composed of textual agent interactions, our scoring functions rely on LLM-based components to map agent inputs and outputs to numerical scores. We compare three LLM regimes as outlined in Section 2.1:

1. *Naive LLM-as-a-judge* is implemented by prompting gpt-4o-mini to estimate error likelihoods with information about the task and step, as in [33];
2. *Prompt and context engineering* produces step-level likelihood estimates using multiple LLMs with role-specific prompts, inspired by ECHO [3]. Scores from LLMs with different roles are averaged to get a final likelihood.
3. A *fine-tuned specialized LLM* is implemented by following the data generation and training paradigm of AEGIS [18] to fine-tune a Qwen3-1.7B model [30].

Additional details on prompts, training, and data are given in Section C. For VCP, step-level scores as described above are used directly. However, L/R/TWF and CRSVP require set-level scores. To enable efficient computation in L/R/TWF we design g to obey monotonicity (Equation (2)) by aggregating step-level scores. Specifically, we use summation and normalize by the trajectory length ℓ to make conformal scores for different datapoints more comparable: $g(c_{j:k}) = \frac{1}{\ell} \sum_{i=j}^{k} \text{LLM}(c_i)$. We experiment with alternative monotonic aggregations, namely Max and LogSumExp, in Section C.2, and discuss circumstances when they may be preferred. Below, we evaluate the discriminatory power of g independently from CP algorithms by treating it as a multiclass classifier.

3.3 Conformal Agent Rollbacks

CP sets for agent error attribution have multiple uses. They can be used by humans for manual debugging; a person can focus only on the steps within the set and have good coverage of true errors. Here, contiguity is a great benefit—it is much easier to debug several consecutive steps than to parse through a scattered set.

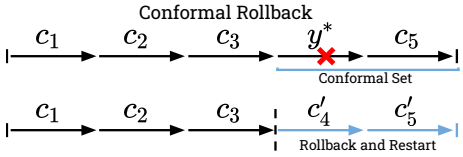

Figure 5: After generating a conformal set for a failed task, we roll back the state of the MAS to the first step in the set, and restart the agent with information on the failed trace.

However, our main downstream application is automated agent recovery. Knowing that an agent has failed, we wish for it to learn from its mistakes and retry parts of the task. Optimally, the agent's state must be rolled back in the trajectory only far enough to cover the decisive error. Rolling back further is inefficient. However, error attribution accuracy is low for point prediction [33, 3, 34].

Prediction sets with coverage guarantees provide the solution for automated rollbacks of failed trajectories. Given a conformal set, we roll back the state of the MAS to the first step in the set, and restart the trajectory, as depicted in Figure 5. The coverage guarantee ensures with high confidence that we roll back far enough to correct the decisive error, while contiguity ensures that we don't roll back excessively far. Upon restarting, we add information to the prompt about the steps taken previously, and instruct the MAS to replan its trajectory to avoid making the same mistakes.

4 Experiments

Our implementation of scoring functions and error attribution algorithms is available at `github.com/layer6ai-labs/conformal-agent-error-attribution`.

4.1 Datasets

We evaluate CP algorithms, scoring functions, and downstream task performance on both a real-world benchmark and synthetic MAS traces. The **Who&When** dataset [33] (MIT license) is a benchmark for step-level error attribution in MAS. Each of the 184 real-world examples contains a full agent execution trace annotated with the decisive error step.[2] Who&When is small-scale and provides limited variety over agents and tasks. It is the largest existing academic dataset with step-level error annotations by humans, allowing us to assess whether our methods apply to real failures beyond the controlled synthetic setting. We also evaluate on the smaller AgentRx dataset [4], with broadly consistent findings on scorer sensitivity reported in Appendix C.2.2.

In the absence of other real-world data, and to provide greater variety, we follow existing practices [18] to synthetically generate failed agent trajectories through error injection. To induce errors at controlled steps, we inject instructions to create errors into the agent's prompt, using the error mode taxonomy of Cemri et al. [5]. For a given trajectory, one step is selected uniformly at random, and the agent is restarted in that state with a modified prompt describing the desired error mode. Given that the overall trajectory fails, the injected step is labeled as the decisive error. We use two mathematical reasoning datasets, MATH [13] and GSM8k [6] (MIT licenses), paired with two representative multi-agent architectures, MACNET [23] and DyLAN [21]. For each dataset–architecture combination, we generate 1200 failed trajectories.

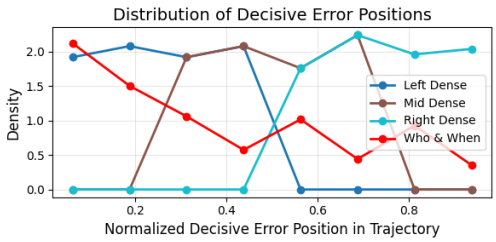

Figure 6: Normalized decisive error position distributions for the real-world benchmark (Who&When), and controlled error injection datasets (Left/Mid/Right Dense) on GSM8k.

Complex tasks generally have some steps that are harder to complete than others, leading to non-uniform decisive error distributions. This distribution is shown in Figure 6 for the Who&When dataset, which tends to have decisive errors early on. To mimic this property, and explore the impact of distribution on error attribution performance, we construct variants of the synthetic data by dividing the normalized trajectory length into thirds, and subsampling conditional on decisive error location. These are the **Left**, **Mid**, and **Right Dense** variants of GSM8k in Figure 6. Our coverage guarantees in Theorems 3.4 and A.2 require exchangeability between calibration and test samples. Hence, calibrated thresholds are not transferable between datasets with different decisive error distributions. Additional details and analysis are provided in Appendix C.

4.2 Evaluation Metrics

Scoring functions g can be viewed as classifiers on an ℓ-way task—predicting the decisive error location. Hence, we evaluate them using AUROC, AUPRC, and accuracy. Since ℓ can be large, baseline levels of these metrics are very low.

[2] We remove 3 inconsistent datapoints where the error index is greater than the trajectory length.

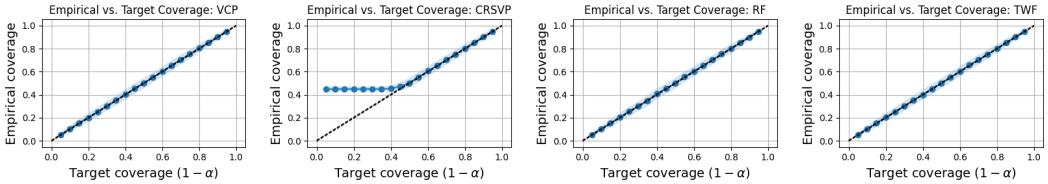

Figure 7: Empirical Coverage vs. Target Coverage for CP methods with the fine-tuned LLM scoring function on MATH.

Given a test dataset $\{(x_i, y_i^*)\}_{i=n+1}^{n+m}$ and predicted sets $\{C(x_i)\}_{i=n+1}^{n+m}$, we evaluate conformal agent error attribution with the following metrics:

$$\text{EC} = \frac{1}{m}\sum_{i=n+1}^{n+m} \mathbb{I}[y_i^* \in C(x_i)]\,; \qquad \text{RR} = \frac{1}{m}\sum_{i=n+1}^{n+m}\left(1 - \frac{|C(x_i)|}{\ell_i}\right). \tag{9}$$

Empirical Coverage (EC) measures how often the predicted set contains the decisive error. EC should be at least $1-\alpha$, the target coverage, and respect the corresponding upper bound where applicable. **Removal Rate** (RR) measures how much of the trajectory is filtered out, essentially measuring prediction set size, where ℓ_i denotes the number of steps in x_i. A higher removal rate indicates smaller conformal sets, and more precise localization of the error step, given equal EC.

Finally, rollback performance is measured across three axes: Success Rate - the fraction of tasks successfully completed after the rollback; Coverage - the fraction of tasks where the state is rolled back at least to the decisive error; and Cost - the fraction of steps rolled back, needing to be redone. As a baseline, we test the agent restarting from the single most likely predicted step (Top-1).

5 Results

5.1 Empirical Verification of Coverage Guarantee

First, we empirically verify that CP algorithms considered in this work satisfy their respective coverage guarantees from Section 3. Figure 7 shows empirical coverage as a function of the target coverage $1-\alpha$, where we average EC over 1000 random, even splits of the calibration and test sets of MATH. The shaded regions show one standard deviation. EC is above the lower bound for all methods, consistent with the theoretical guarantees. VCP, RF, and TWF all obey their respective upper bounds, leading to exact linear behaviour. A deviation is observed in the low $1-\alpha$ regime for CRSVP, but this is not a violation as CRSVP does not provide an upper bound on coverage.

5.2 Scoring Function Evaluation

Next, we compare the three regimes of scoring functions for discriminatory power as binary classifiers. As a baseline, we also consider a random scoring function that assigns each step a score drawn uniformly from $[0, 1]$ and normalizes the scores to sum to one within each trajectory. Table 2 shows results on a combination of GSM8k and MATH test sets, matching the training distribution for the fine-tuned LLM. Increasing the complexity of g does pay off; fine-tuning on the error classification task considerably increases its power. The naive prompted LLM is hardly better than random guessing, while prompt engineering provides small improvements. These results are consistent with performance levels observed in prior studies [33, 3, 18].

Table 2: Step-level discrimination metrics across scoring functions on combined MATH and GSM8k test set.

Scoring function	AUROC	AUPRC	Accuracy
Random Guessing (baseline)	0.500	0.118	0.118
Naive LLM (gpt-4o-mini)	0.519	0.128	0.110
Role Prompted (gpt-4o-mini)	0.554	0.161	0.164
Fine-tuned (Qwen3-1.7B)	**0.762**	**0.382**	**0.731**

5.3 Conformal Agent Error Attribution Evaluation

Our main comparison of error attribution examines the removal rate (Equation (9)) for conformal algorithms and scoring functions across real and synthetic datasets, including trajectories created

Table 3: Removal rate ($\pm$ std over 1000 calibration/test splits) for conformal algorithms and scoring functions at $1 - \alpha = 0.8$. Higher removal rates indicate more precise error attribution, with the best performing conformal algorithm in each column bolded among algorithms which produce contiguous sets. Averages are taken over conformal algorithms to evaluate scoring functions. $\times$ indicates that fine-tuning is not possible due to insufficient data.

	Conformal Method	Who&When	GSM8k Left Dense		GSM8k Mid Dense		GSM8k Right Dense		MATH	
			DyLAN	MACNET	DyLAN	MACNET	DyLAN	MACNET	DyLAN	MACNET
Random Guessing	Vanilla	0.20±0.06	0.20±0.06	0.20±0.06	0.21±0.05	0.19±0.05	0.19±0.06	0.19±0.05	0.20±0.04	0.19±0.04
	CRSVP	0.23±0.05	0.27±0.04	0.19±0.04	0.27±0.04	0.19±0.04	0.28±0.04	0.21±0.05	0.28±0.03	0.19±0.03
	Left Filtering	0.12±0.04	0.02±0.03	0.07±0.05	0.30±0.01	0.35±0.02	0.53±0.01	**0.61±0.02**	0.19±0.02	0.21±0.02
	Right Filtering	**0.30±0.05**	**0.64±0.01**	0.60±0.02	0.43±0.02	0.34±0.01	0.20±0.01	0.04±0.05	**0.28±0.02**	0.20±0.02
	Two-Way	0.19±0.06	0.03±0.05	0.14±0.10	**0.61±0.03**	**0.60±0.02**	0.40±0.03	0.07±0.09	**0.29±0.02**	0.02±0.05
	Average	0.21	0.23	0.24	0.36	0.33	0.32	0.22	0.25	0.16
Naive LLM gpt-4o-mini	Vanilla	0.20±0.06	0.21±0.02	0.24±0.03	0.18±0.03	0.15±0.03	0.16±0.03	0.17±0.03	0.19±0.02	0.20±0.03
	CRSVP	0.21±0.04	0.58±0.03	0.54±0.02	0.29±0.04	0.21±0.04	0.21±0.03	0.10±0.03	0.30±0.03	0.20±0.03
	Left Filtering	0.12±0.04	0.10±0.01	0.24±0.01	0.31±0.01	0.47±0.02	0.53±0.01	**0.60±0.02**	0.18±0.02	0.19±0.02
	Right Filtering	**0.31±0.04**	**0.64±0.02**	0.71±0.01	0.43±0.02	0.45 ±0.01	0.20±0.02	0.20±0.01	**0.27±0.02**	0.20±0.02
	Two-Way	0.19±0.04	0.17±0.03	0.22 ±0.03	**0.59±0.03**	**0.59±0.01**	0.40±0.03	0.21±0.03	**0.27±0.02**	0.15±0.03
	Average	0.21	0.34	0.39	0.36	0.37	0.30	0.26	0.24	0.19
Role-prompted gpt-4o-mini	Vanilla	0.19±0.06	0.30±0.04	0.24±0.04	0.27±0.04	0.27±0.03	0.22±0.04	0.25±0.04	0.19±0.03	0.24±0.04
	CRSVP	0.23±0.04	0.27±0.04	0.21±0.05	0.27±0.04	0.26±0.05	0.27±0.04	0.22±0.05	**0.27±0.03**	0.22±0.03
	Left Filtering	0.12±0.04	0.10±0.02	0.31±0.01	0.36±0.01	0.53±0.01	**0.64±0.02**	**0.60±0.02**	0.19±0.02	0.19±0.02
	Right Filtering	**0.30±0.05**	**0.63±0.02**	0.59±0.02	0.43±0.01	0.33±0.01	0.21±0.01	0.10±0.04	**0.28±0.02**	0.19±0.02
	Two-Way	0.22±0.02	0.18±0.02	0.21±0.02	**0.62±0.02**	**0.59±0.02**	0.42±0.02	0.20±0.02	**0.29±0.02**	0.18±0.02
	Average	0.21	0.30	0.31	0.39	0.40	0.35	0.27	0.24	0.20
Fine-tuned Qwen3-1.7B	Vanilla	$\times$	0.78±0.03	0.82±0.02	0.66±0.03	0.75±0.02	0.34±0.04	0.63±0.03	0.33±0.03	0.62±0.03
	CRSVP	$\times$	**0.69±0.06**	**0.80±0.02**	0.36±0.05	0.38±0.07	0.24±0.03	0.29±0.06	**0.32±0.03**	**0.30±0.04**
	Left Filtering	$\times$	0.09±0.01	0.10±0.02	0.31±0.02	0.35±0.01	0.52±0.01	**0.60±0.01**	0.18±0.02	0.19±0.02
	Right Filtering	$\times$	**0.65±0.02**	0.63±0.02	0.44±0.01	0.33±0.01	0.20±0.01	0.10±0.02	0.27±0.01	0.20±0.02
	Two-Way	$\times$	0.18±0.02	0.20±0.05	**0.63±0.02**	**0.60±0.02**	0.40±0.02	0.19±0.05	**0.29±0.02**	0.19±0.02
	Average	$\times$	0.48	0.51	0.48	0.48	0.33	0.36	0.28	0.30

through two agentic frameworks. Table 3 displays the results using a target coverage of 80%. Results are shown as the mean and standard deviation over 1000 random splits of the calibration and test data.

Our first observation is that the efficacy of conformal algorithms depends on the distribution of errors in the dataset. The real-world Who&When dataset has decisive errors heavily skewed toward early steps in the trajectory (see Figure 6). This distribution naturally aligns with RF which can return a short prefix, and accordingly shows the strongest performance. The synthetic datasets demonstrate this behaviour even more clearly. LF is the strongest algorithm on right-dense data, and TWF successfully isolates errors in the middle of trajectories. Conformal error attribution is most successful when the CP algorithm is matched to the data's error distribution. Luckily, this is simple to achieve in practice. Applying CP already requires a calibration dataset from a distribution similar to the test set. Practitioners can visualize the error distribution of their calibration data like in Figure 6, and choose the appropriate filtering algorithm, or automate the selection based on the mean error location of trajectories, for example.

Notably, filtering algorithms are less dependent on the discriminatory power of the scoring function than VCP and CRSVP. While the baseline approaches tend to require the fine-tuned LLM scoring function to achieve high RR, filtering methods are largely insensitive. This means that L/R/TWF can be used with simple, easier to design scoring functions as long as they are properly matched to the data's error distribution. This is especially important for real-world data where labeling is expensive; Who&When only supplies 184 labeled datapoints—enough to calibrate and test, but not nearly enough to also fine-tune an LLM. Our filtering algorithms remain more practical in this setting.

Conversely, stronger step-level discrimination does not necessarily translate into higher removal rates for filtering methods. One explanation is that, when the filtering direction matches the error distribution, much of the reduction in set size can already be achieved by removing the long portion of the trajectory beyond the error region. For example, if decisive errors are confined to the first quarter of trajectories, retaining this quarter as a prefix removes the remaining 75% without distinguishing individual steps within it. A stronger scoring function may further refine the filtering boundary, but the additional improvement is limited when this boundary already lies close to the decisive error.

5.4 Computational Efficiency

As noted throughout Section 3 and Table 1, at inference existing conformal methods including VCP and CRSVP require evaluating the scoring function g on each step in the trajectory. LF and RF need only evaluate steps from one direction until their threshold is crossed, which can be much more efficient. To demonstrate this quantitatively, we count the average NFE of g used by each algorithm when predicting on the GSM8k test set variants. The result in Table 4 confirms that VCP, CRSVP, and TWF use exactly ℓ function calls per trajectory. In contrast, LF shows a clear advantage on right-dense error distributions, while RF uses a mere 36% of the calls on left-dense data where most errors occur early. This result emphasizes the cost advantages of selecting an algorithm that has good implicit bias for the distribution observed in calibration data.

Table 4: Inference cost comparison for conformal methods in terms of average NFE of g (Fine-tuned LLM) per trajectory with 80% target coverage on GSM8k variants.

Conformal Method	Left Dense	Mid Dense	Right Dense
Vanilla	8.50	8.50	8.50
CRSVP	8.50	8.50	8.50
Left Filtering	7.50	5.64	**3.70**
Right Filtering	**3.08**	**5.16**	7.09
Two-Way	8.50	8.50	8.50

5.5 Conformal Agent Rollbacks

As noted in Section 3.3, CP sets can enable automated rollbacks for failed tasks with high confidence that decisive errors are captured. We evaluate automated rollbacks on the GSM8K test set variants. For each dataset, we randomly sample 100 MACNET trajectories and score them with the fine-tuned LLM scoring function. As a baseline method, we use the single step in the trajectory with the highest likelihood (Top-1) as the restart point. For conformal rollbacks we predict a conformal set using either the VCP or LF algorithm with a target coverage of 80%, and then restart at the first step in the set. In all cases we restart the MACNET agent with a modified prompt containing the previously failed trace to help the system avoid repeating the same mistakes. As shown in Table 5, LF achieves the highest success rate by a small margin, within one standard deviation of baselines. However, LF is the only method which gives control over coverage: Top-1 gives no guarantee, while VCP overcovers in this setting because its sets are not contiguous.

Table 5: Automated rollback results using CP sets from VCP and LF compared to Top-1 predictions. Datasets are variants of GSM8k, and the scoring function uses the fine-tuned LLM.

Dataset	Method	Success Rate ↑	Coverage	Cost ↓
Left Dense	Top-1	0.76±0.05	0.92±0.03	0.83±0.02
	VCP	0.73±0.05	0.85±0.03	0.79±0.02
	LF	0.77±0.04	0.81±0.05	0.90±0.00
Mid Dense	Top-1	0.67±0.04	0.86±0.04	0.56±0.02
	VCP	0.59±0.05	0.95±0.02	0.65±0.01
	LF	0.68±0.05	0.81±0.03	0.65±0.01
Right Dense	Top-1	0.71±0.05	0.86±0.04	0.50±0.02
	VCP	0.70±0.05	1.00±0.00	0.66±0.02
	LF	0.75±0.04	0.82±0.04	0.40±0.01

6 Conclusion

In this work we designed novel set-prediction algorithms for the agent error attribution task. By taking account of the data's sequential nature, our filtering-based conformal algorithms showed strong ability to precisely isolate decisive errors in agent trajectories within contiguous sets, and with much less compute needed at inference time. Our conformal algorithms can be used as one part of an agent improvement pipeline, allowing humans to more efficiently understand and debug errors in their agents, or enabling fully automated rollbacks wherein agents correct their own mistakes. The limitations of our methods include: the requirement of exchangeable data, which could be loosened with standard CP methods to handle distribution shifts [10]; the need to match algorithm choice to data distribution, which can easily be done by checking the calibration dataset; and the assumption that failed trajectories have a single decisive error, which can be naturally expanded to the setting of an arbitrary number of agent errors by using a CP guarantee on the recall level over *all* errors [19].

Impact Statement

We design systems using conformal prediction to systematically identify and isolate errors in multi-agent systems, providing a principled, model-agnostic method to improve reliability of error attribution. This bridges uncertainty quantification with practical debugging in complex agent interactions, enabling more robust MAS development. We do not anticipate significant negative societal impacts from this work.

References

[1] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv:2107.07511*, 2021.

[2] Anastasios N. Angelopoulos, Stephen Bates, Michael Jordan, and Jitendra Malik. Uncertainty sets for image classifiers using conformal prediction. In *International Conference on Learning Representations*, 2021.

[3] Adi Banerjee, Anirudh Nair, and Tarik Borogovac. Where did it all go wrong? a hierarchical look into multi-agent error attribution. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025.

[4] Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI agent failures from execution trajectories. In *Findings of the Association for Computational Linguistics: EMNLP 2026*. Association for Computational Linguistics, 2026. ISBN 979-8-89176-462-0.

[5] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why Do Multi-Agent LLM Systems Fail? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

[7] Jesse C. Cresswell, Yi Sui, Bhargava Kumar, and Noël Vouitsis. Conformal Prediction Sets Improve Human Decision Making. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 9439–9457, 2024.

[8] Jesse C. Cresswell, Bhargava Kumar, Yi Sui, and Mouloud Belbahri. Conformal Prediction Sets Can Cause Disparate Impact. In *The Thirteenth International Conference on Learning Representations*, 2025.

[9] Alireza Ghafarollahi and Markus J. Buehler. SciAgents: Automating Scientific Discovery Through Bioinspired Multi-Agent Intelligent Graph Reasoning. *Advanced Materials*, 37(22), 2024. doi: 10.1002/adma.202413523.

[10] Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. In *Advances in Neural Information Processing Systems*, volume 34, pages 1660–1672, 2021.

[11] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/890.

[12] Chirag Gupta, Arun K. Kuchibhotla, and Aaditya Ramdas. Nested conformal prediction and quantile out-of-bag ensemble methods. *Pattern Recognition*, 127:108496, 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108496.

[13] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

[14] Floris den Hengst, Inès Blin, Majid Mohammadi, Syed Ihtesham Hussain Shah, and Taraneh Younesian. Hierarchical conformal classification. *arXiv:2508.13288*, 2025.

[15] 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. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Dong Huang, Jie M Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. AgentCoder: Multi-Agent-based Code Generation with Iterative Testing and Optimisation . *arXiv:2312.13010*, 2023.

[17] Jianguo Huang, Huajun Xi, Linjun Zhang, Huaxiu Yao, Yue Qiu, and Hongxin Wei. Conformal prediction for deep classifier via label ranking. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[18] Fanqi Kong, Ruijie Zhang, Huaxiao Yin, Guibin Zhang, Xiaofei Zhang, Ziang Chen, Zhaowei Zhang, Xiaoyuan Zhang, Song-Chun Zhu, and Xue Feng. Aegis: Automated error generation and attribution for multi-agent systems. In *The Fourteenth International Conference on Learning Representations*, 2026.

[19] Bruce Kuwahara, Chen-Yuan Lin, Xiao Shi Huang, Kin Kwan Leung, Jullian Arta Yapeter, Ilya Stanevich, Felipe Perez, and Jesse C Cresswell. Document summarization with conformal importance guarantees. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[20] Kin Kwan Leung, Mouloud Belbahri, Yi Sui, Alex Labach, Xueying Zhang, Stephen Anthony Rose, and Jesse C. Cresswell. Classifying and addressing the diversity of errors in retrieval-augmented generation systems. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3185–3207, 2026. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.147.

[21] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A Dynamic LLM-Powered Agent Network for Task-Oriented Agent Collaboration. In *First Conference on Language Modeling*, 2024.

[22] Thomas Mortier, Alireza Javanmardi, Yusuf Sale, Eyke Hüllermeier, and Willem Waegeman. Conformal prediction in hierarchical classification with constrained representation complexity. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.

[23] Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling Large Language Model-based Multi-Agent Collaboration. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Narges Razavian and David Sontag. Temporal convolutional neural networks for diagnosis from lab tests. *arXiv:1511.07938*, 2015.

[25] Yaniv Romano, Matteo Sesia, and Emmanuel Candès. Classification with valid and adaptive coverage. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[26] Brendan Leigh Ross, Noël Vouitsis, Atiyeh Ashari Ghomi, Rasa Hosseinzadeh, Ji Xin, Zhaoyan Liu, Yi Sui, Shiyi Hou, Kin Kwan Leung, Gabriel Loaiza-Ganem, and Jesse C. Cresswell. Textual bayes: Quantifying prompt uncertainty in LLM-based systems. In *The Fourteenth International Conference on Learning Representations*, 2026.

[27] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[28] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

[29] Ning Wu, Ming Gong, Linjun Shou, Shining Liang, and Daxin Jiang. Large language models are diverse role-players for summarization evaluation. In *Natural Language Processing and Chinese Computing*, pages 695–707. Springer Nature Switzerland, 2023. ISBN 978-3-031-44693-1.

[30] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin

Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

[31] Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan W. Suchow, Zhenyu Cui, Rong Liu, Zhaozhuo Xu, Denghui Zhang, Koduvayur Subbalakshmi, Guojun Xiong, Yueru He, Jimin Huang, Dong Li, and Qianqian Xie. FinCon: A Synthesized LLM Multi-Agent System with Conceptual Verbal Reinforcement for Enhanced Financial Decision Making. In *Advances in Neural Information Processing Systems*, volume 37, pages 137010–137045, 2024. doi: 10.52202/079017-4354.

[32] Yifan Yu, Moyan Li, Shaoyuan Xu, Jinmiao Fu, Xinhai Hou, Fan Lai, and Bryan Wang. CORRECT: COndensed eRror RECognition via knowledge Transfer in multi-agent systems. *arXiv:2509.24088*, 2025.

[33] Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. In *Forty-second International Conference on Machine Learning*, 2025.

[34] Chenyang Zhu, Spencer Hong, Jingyu Wu, Kushal Chawla, Yuhui Tang, Youbing Yin, Nathan Wolfe, Erin Babinsky, and Daben Liu. RAFFLES: Reasoning-based attribution of faults for LLM systems. In *First Workshop on Multi-Turn Interactions in Large Language Models*, 2025.

A Theorems and Proofs

In this appendix we provide complete proofs of the theorems stated in the main text.

A.1 Vanilla Conformal Prediction

The theorem showing valid coverage for VCP with variable number of classes is essentially the same as the standard result for conformal prediction, for instance as presented by [1], since the number of classes per datapoint does not affect exchangeability of the conformal scores. Although not novel, we present the theorem below for completeness. Note that following tradition in CP, the scores in this algorithm reflect *non-conformity* in that they take higher values when y disagrees with x.

Theorem A.1. *Suppose* $\{(x_i, y_i^*)\}_{i=1}^n$ *and* (x_{n+1}, y_{n+1}^*) *are exchangeable, where the number of possible classes* ℓ_i *varies per datapoint. Define* $\hat{q}$ *as the* $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ *quantile of the scores* $\{S_i\}_{i=1}^n = \{S(x_i, y_i^*)\}_{i=1}^n$. *Then prediction sets constructed as* $C(x_{n+1}; \hat{q}) = \{y \in \mathcal{Y}(x_{n+1}) \mid S(x_{n+1}, y) \leq \hat{q}\}$ *satisfy* $\mathbb{P}[y_{n+1}^* \in C(x_{n+1}; \hat{q})] \geq 1 - \alpha$.

Proof. Since the fixed scoring function S is applied to each element of the exchangeable sequence $(x_1, y_1^*), \ldots, (x_{n+1}, y_{n+1}^*)$, the resulting sequence of scalar scores $S_1, \ldots, S_{n+1}$ is also exchangeable. We assume for simplicity that the scores are non-degenerate, but ties can be handled by defining S to add a small amount of noise to its output. Let $S_{(1)} \leq S_{(2)} \leq \cdots \leq S_{(n)}$ be the order statistics of the calibration scores so that the empirical quantile $\hat{q}$ can be defined as $S_{(k)}$, where $k = \lceil (n+1)(1-\alpha) \rceil$ (assuming $n \geq \frac{1}{\alpha} - 1$ to ensure $k \leq n$). Given the definition of $C(x_{n+1}; \hat{q})$, the event $y_{n+1}^* \in C(x_{n+1}; \hat{q})$ is equivalent to the event $S_{n+1} \leq \hat{q}$, which occurs if and only if S_{n+1} is among the k smallest scores overall. From exchangeability, all orderings are equally likely, so $\mathbb{P}[S_{n+1} \leq \hat{q}] = \frac{k}{n+1}$. Altogether we have

$$\mathbb{P}[y_{n+1}^* \in C(x_{n+1}; \hat{q})] = \frac{k}{n+1} = \frac{\lceil (n+1)(1-\alpha) \rceil}{n+1} \geq \frac{(n+1)(1-\alpha)}{n+1} = 1 - \alpha. \tag{10}$$

□

We note that the variable number of classes ℓ_i per datapoint does not come up in the proof, and is only present in the assumption that such datapoints can still be exchangeable. This is certainly possible when we treat ℓ_i as a (random) property of x. If desired, one could treat ℓ_i as an explicit random variable and draw points as $(x_i, \ell_i, y_i^*) \sim \mathbb{P}$. Another way of viewing this extension would be to set a fixed ℓ as some upper bound of possible sizes, such as $\ell = \max\{\ell_i\}$. Then, ordinary conformal prediction can be performed with a model f defined over ℓ classes but giving $f_j = 0$ when $j > \ell_i$, as long as we assume $\ell_{n+1} \leq \ell$.

A.2 Left (Right) Filtration

Here we provide full formal proofs for each of the steps in Section 3.1.3. For ease of reference, we repeat key definitions and assumptions. A trajectory x is a sequence of a variable number ℓ steps $(c_1, \ldots, c_\ell)$, and the decisive error $y^* = c_{j^*}$ is the step at index j^*. The scoring function g_{LF} scores subintervals $c_{j:k} \subseteq x$ with the properties $g_{\text{LF}}(\varnothing) = 0$ and $g_{\text{LF}}(x) = \infty$. The filtering function is defined as $F_{\text{LF}}(x; q) := \arg\max_{c_{j:\ell} \in \mathcal{I}_{\text{LF}}} \left(|c_{j:\ell}| \mid g_{\text{LF}}(c_{j:\ell}) \leq q \right)$ which optimizes over suffixes $\mathcal{I}_{\text{LF}} := \{c_{j:\ell}\}_{j=1}^{\ell+1}$ and satisfies $F_{\text{LF}}(x; \infty) = x$. Meanwhile, the conformal score function is defined as $S_{\text{LF}}(x, y^*) := \inf \left(q \in \mathbb{R}^+ \mid y^* \in F_{\text{LF}}(x; q) \right)$. Both of these functions optimize over a set of steps, and it will be convenient to refer to those sets as follows:

$$\mathcal{V}_q := \{c_{j:\ell} \in \mathcal{I}_{\text{LF}} \mid g_{\text{LF}}(c_{j:\ell}) \leq q\}, \tag{11}$$

$$\mathcal{Q} := \{q \in \mathbb{R}^+ \mid y^* \in F_{\text{LF}}(x; q)\}. \tag{12}$$

Lemma 3.1. *For any* x *and thresholds* $0 \leq q_1 \leq q_2$, $F_{LF}(x; q_1) \subseteq F_{LF}(x; q_2)$.

Proof. $\mathcal{V}_q$ is the set of valid suffixes for $F_{\text{LF}}(x; q)$. For every $c_{j:\ell} \in \mathcal{V}_{q_1}$ we have $g_{\text{LF}}(c_{j:\ell}) \leq q_1 \leq q_2$, and so $c_{j:\ell} \in \mathcal{V}_{q_2}$. Hence, $\mathcal{V}_{q_1} \subseteq \mathcal{V}_{q_2}$. $F_{\text{LF}}(x; q)$ returns the longest suffix in $\mathcal{V}_q$, so we also have $|F_{\text{LF}}(x; q_1)| \leq |F_{\text{LF}}(x; q_2)|$. Since the suffixes $c_{j:\ell} \in \mathcal{I}_{\text{LF}}$ are nested (i.e. $j_2 < j_1 \iff c_{j_1:\ell} \subset$

$c_{j_2:\ell}$), we also have $|c_{j_1:\ell}| \le |c_{j_2:\ell}| \implies c_{j_1:\ell} \subseteq c_{j_2:\ell}$, which gives the desired nesting property of $F_{\text{LF}}(x; q)$. □

Proposition 3.2. *When g_{LF} obeys monotonicity such that $c_{b:c} \subseteq c_{a:d} \implies g_{LF}(c_{b:c}) \le g_{LF}(c_{a:d})$, then $S_{LF}(x, y^*) = g_{LF}(c_{j^*:\ell})$.*

Proof. $\mathcal{Q}$ is the set of valid thresholds for $S_{\text{LF}}(x, y^*)$. Let $q^* = g_{\text{LF}}(c_{j^*:\ell})$, and note that $c_{j^*:\ell} \in \mathcal{V}_{q^*}$. Since $|c_{j^*:\ell}| < |c_{j:\ell}|$ only when $j < j^*$, and $y^* \in c_{j:\ell}$ for these cases, the longest suffix in $\mathcal{V}_{q^*}$ contains y^*. Hence, we have that $y^* \in F_{\text{LF}}(x; q^*)$, and $q^* \in \mathcal{Q}$.

Now consider any $q' < q^*$. Notice that $c_{j^*:\ell} \notin \mathcal{V}_{q'}$. By monotonicity of g_{LF}, for all $j < j^*$, $g_{\text{LF}}(c_{j:\ell}) \ge q^* > q'$. Therefore, the longest suffix in $\mathcal{V}_{q'}$ is shorter than $c_{j^*:\ell}$, and does not contain y^*. Hence, $q' \notin \mathcal{Q}$.

Since $q^* \in \mathcal{Q}$, but all $q' < q^*$ are not, q^* is the minimum of $\mathcal{Q}$, and $S_{\text{LF}}(x, y^*) = q^* = g_{\text{LF}}(c_{j^*:\ell})$. □

Lemma 3.3. *For a fixed $\hat{q}$, we have $S_{LF}(x, y^*) \le \hat{q} \iff y^* \in F_{LF}(x; \hat{q})$.*

Proof. To begin, we show that the infimum in $S_{\text{LF}}(x, y^*)$ is always achieved. Note that the set of suffix scores $g_{\text{LF}}(c_{j:\ell})$ is finite. Considering only the valid suffixes that contain y^*, let $q_{\min} = \min\{g_{\text{LF}}(c_{j:\ell}) \mid j \le j^*\}$. Due to the nesting of suffixes, the condition $y^* \in F_{\text{LF}}(x; q)$ is satisfied if and only if there exists a valid suffix with score $\le q$, which occurs precisely when $q_{\min} \le q$. Consequently, $\mathcal{Q}$ is the closed interval $[q_{\min}, \infty)$, and the infimum in $S_{\text{LF}}(x, y^*)$ is achieved at $q_{\min}$.

Hence we have a chain of equivalences:

$$S_{\text{LF}}(x, y^*) \le \hat{q} \iff q_{\min} \le \hat{q} \iff \hat{q} \in \mathcal{Q} \iff y^* \in F_{\text{LF}}(x; \hat{q}). \tag{13}$$

□

A.3 Two-Way Filtration

Next we cover the proofs of statements in Section 3.1.4. Continuing from the definitions and results above, including their counterparts for RF, for TWF we had $F_{\text{TWF}}(x; q) := F_{\text{LF}}(x; q) \cap F_{\text{RF}}(x; q)$ (where $F_{\text{TWF}}(x; 0) = \varnothing$ and $F_{\text{TWF}}(x; \infty) = x$), and $S_{\text{TWF}}(x, y^*) := \inf\big(q \in \mathbb{R}^+ \mid y^* \in F_{\text{TWF}}(x; q)\big)$.

Lemma 3.5. *For any x and thresholds $0 \le q_1 \le q_2$, $F_{TWF}(x; q_1) \subseteq F_{TWF}(x; q_2)$.*

Proof. From Lemma 3.1 we have $F_{\text{LF}}(x; q_1) \subseteq F_{\text{LF}}(x; q_2)$, and the equivalent statement for F_{RF} holds. The intersection operation preserves set inclusion: for any sets A, B, C, D, if $A \subseteq B$ and $C \subseteq D$, then $A \cap C \subseteq B \cap D$. Hence,

$$F_{\text{TWF}}(x; q_1) = F_{\text{LF}}(x; q_1) \cap F_{\text{RF}}(x; q_1) \subseteq F_{\text{LF}}(x; q_2) \cap F_{\text{RF}}(x; q_2) = F_{\text{TWF}}(x; q_2). \tag{14}$$

□

Proposition 3.6. *$S_{TWF}(x, y^*)$ as defined in Equation* (6) *can be expressed as*

$$S_{TWF}(x, y^*) = \max\big(S_{LF}(x, y^*), S_{RF}(x, y^*)\big). \tag{7}$$

When g_{LF} and g_{RF} obey the monotonicity condition in Equation (2)*, S_{TWF} further simplifies to*

$$S_{TWF}(x, y^*) = \max\big(g_{LF}(c_{j^*:\ell}), g_{RF}(c_{1:j^*})\big). \tag{8}$$

Proof. By the definition of F_{TWF}, $y^* \in F_{\text{TWF}}(x; q)$ if and only if $y^* \in F_{\text{LF}}(x; q)$ and $y^* \in F_{\text{RF}}(x; q)$. The condition $y^* \in F_{\text{LF}}(x; q)$ holds if and only if $q \ge S_{\text{LF}}(x, y^*)$ by the definition of $F_{\text{LF}}(x; q)$ and its nesting property from Lemma 3.1. Similarly, $y^* \in F_{\text{RF}}(x; q)$ if and only if $q \ge S_{\text{RF}}(x, y^*)$. Hence, $y^* \in F_{\text{TWF}}(x; q)$ if and only if $q \ge \max(S_{\text{LF}}(x; y^*), S_{\text{RF}}(x; y^*))$. By taking the infimum for $S_{\text{TWF}}(x; y^*)$, we reach Equation (7).

From Proposition 3.2, when g_{LF} obeys its monotonicity condition, we have $S_{\text{LF}}(x, y^*) = g_{\text{LF}}(c_{j^*:\ell})$, and by a similar argument $S_{\text{RF}}(x; y^*) = g_{\text{RF}}(c_{1:j^*})$. Substituting these results into Equation (7) gives Equation (8). □

Lemma 3.7. *For a fixed $\hat{q}$, we have $S_{TWF}(x, y^*) \leq \hat{q} \iff y^* \in F_{TWF}(x; \hat{q})$.*

Proof. From the definition of $F_{\text{TWF}}(x, q)$ as an intersection, the event $y^* \in F_{\text{TWF}}(x, \hat{q})$ occurs if and only if both $y^* \in F_{\text{LF}}(x, \hat{q})$ and $y^* \in F_{\text{RF}}(x, \hat{q})$ occur. We have already shown that $y^* \in F_{\text{LF}}(x, \hat{q}) \iff S_{\text{LF}}(x, y^*) \leq \hat{q}$ (Lemma 3.3), and similarly for RF. Hence,

$$y^* \in F_{\text{TWF}}(x, \hat{q}) \iff S_{\text{LF}}(x, y^*) \leq \hat{q} \wedge S_{\text{RF}}(x, y^*) \leq \hat{q}. \tag{15}$$

Equivalently we write $\max\left(S_{\text{LF}}(x, y^*), S_{\text{RF}}(x, y^*)\right) \leq \hat{q}$, which is the same as $S_{\text{TWF}}(x, y^*) \leq \hat{q}$ (Proposition 3.6). □

Theorem A.2. *Suppose $\{(x_i, y_i^*)\}_{i=1}^n$ and (x_{n+1}, y_{n+1}^*) are exchangeable. Given scoring functions g_{LF} and g_{RF} as defined above, define the conformal score $S_{TWF}(x_i, y_i^*)$ as in Equation* (6)*. Let $\hat{q}$ be the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of conformal scores $\{S_i\}_{i=1}^n = \{S_{TWF}(x_i, y_i^*)\}_{i=1}^n$. Then prediction sets constructed as $C_{TWF}(x_{n+1}; \hat{q}) = F_{TWF}(x_{n+1}; \hat{q})$ satisfy $1 - \alpha \leq \mathbb{P}[y_{n+1}^* \in C_{TWF}(x_{n+1}; \hat{q})] < 1 - \alpha + \frac{1}{n+1}$.*

Proof. The proof proceeds in the same way as Theorem 3.4, using Lemma 3.7 instead of Lemma 3.3. □

As a final point, we note that prediction sets constructed as $C_{\text{TWF}}(x_{n+1}; \hat{q}) = F_{\text{TWF}}(x_{n+1}; \hat{q})$ may be empty due to the use of intersection. While this is not strictly a concern as coverage is valid marginally, it can be undesirable to predict empty sets. As a fallback, TWF can predict the single most likely step under g_{LF} or g_{RF} (which are likely to be the same function in practice).

B Additional Details of Conformal Algorithms

In Section 3.1.2 we described how to apply the CRSVP algorithm by Mortier et al. [22], originally designed for hierarchical classification, to sequential data like agent trajectories. Here we provide a more detailed description of our implementation.

To recap, a binary tree $\mathcal{T}$ is constructed from an agent trajectory $x = (c_1, \ldots, c_\ell)$ to have root node $v_1 = [\ell]$, and leaf nodes $v_\ell, \ldots, v_{2\ell-1}$ as individual steps $c_1, \ldots, c_\ell$. For each calibration datapoint (x, y^*), a scoring function g_{CRSVP} is used to find the single leaf $\bar{v}$ most likely to contain the error y^*. If $\bar{v}$ is itself the decisive error y^*, then the conformal score is returned as $S_{\text{CRSVP}}(x, y^*) := g_{\text{CRSVP}}(\bar{v})$. Otherwise, the tree is traversed upwards until the first node v^* is reached which does contain y^*. Since the number of steps in each node grows exponentially as the tree is traversed, simply using the score $g_{\text{CRSVP}}(v^*)$ would greatly overcover the true labels. Hence, CRSVP interpolates between steps a random amount, similar to other conformal scoring functions [2]. Let v^{*-1} be the node before v^* on the traversal path. Then the conformal score is set as

$$S_{\text{CRSVP}}(x, y^*) := g_{\text{CRSVP}}(v^*) - u \cdot (g_{\text{CRSVP}}(v^*) - g_{\text{CRSVP}}(v^{*-1})), \tag{16}$$

where $u \sim \mathcal{U}(0, 1)$ is noise drawn uniformly from $[0, 1]$. Note that parent nodes are strict supersets of their children, which means prediction sets are nested when traversing from leaf to root [12].

After conformal calibration to get the $\frac{\lfloor (n+1)(\alpha) \rfloor}{n}$ quantile of the scores[3], which we denote $\hat{q}$, for a test datapoint x_{n+1}, CRSVP again finds the most likely leaf node $\bar{v}$ according to g_{CRSVP}. If that node's score is below the threshold, $g_{\text{CRSVP}}(\bar{v}) < \hat{q}$, CRSVP recursively traverses up the tree until the threshold is surpassed at $\hat{v}$. Drawing noise u, if $g_{\text{CRSVP}}(\hat{v}) - u \cdot (g_{\text{CRSVP}}(\hat{v}) - g_{\text{CRSVP}}(\hat{v}^{-1})) \geq \hat{q}$, then $\hat{v}^{-1}$ is returned, otherwise $\hat{v}$ itself is. By construction all prediction sets generated this way are contiguous subintervals of x_{n+1} and the lower bound on coverage is guaranteed (Equation (1)) [22].

[3] Note that the quantile is different than VCP because we use conformity scores in CRSVP, and $1 - g$ as a nonconformity score for VCP. These are merely conventions and can easily be modified.

Table 6: Fine-tuning configuration

Model	Dataset	Epochs	Batch Size	Precision	LoRA (r, α)	LR
Qwen3-1.7B	GSM8k + MATH	18	32	8-bit	(2, 8)	2×10^{-6}

C Implementation Details and Additional Results

C.1 Additional Experiment Details

In this section we provide more detail about datasets, models, and experimental results.

C.1.1 Synthetic Dataset Generation

As discussed in Section 4.1 we construct three variants of the original GSM8k dataset that explicitly control the location of the decisive error step along the execution trajectory. Errors are first generated uniformly by selecting one step in a trajectory, and injecting instructions to the agent to fail on that step in various ways [18, 5]. We generate 1200 failed trajectories for GSM8k for each of the DyLAN and MACNET agent frameworks (and similarly for the MATH dataset). Then the trajectories are split by error position, in the first third, middle third, or final third to form the Right-, Mid-, and Left-Dense GSM8k datasets.

C.1.2 Context-Engineered LLM Implementation

Prior work has shown that engineering the context of LLMs and prompting them with diverse role-specific instructions can improve evaluation robustness [29]. Following this general idea, we employ multiple role prompts, inspired by ECHO [3], to obtain step-level scores from different perspectives. We first use the LLM to summarize why the agent failed the task using the overall trace, then pass this information to LLMs with four different roles to generate scores. The average of these scores is the final step score. The exact prompts used for the context-engineered LLM can be found in Section C.3.

C.1.3 Fine-tuned Scoring Function

As described in Section 3.2, we fine-tuned an LLM to estimate step-level likelihoods of being the decisive error step.

Training data Due to limited data in Who&When [33], we only perform fine-tuning on the MATH and GSM8k datasets with synthetically injected errors. As mentioned in Section C.1.1, we generated 4,800 trajectories in total across datasets and MAS architectures. Of these, 4,000 were used for model fine-tuning, equally split across the datasets, architectures, and error locations. We then used an 85%/15% split for training and validation to support hyperparameter selection. The 800 trajectories not used for training were held out entirely from fine-tuning and instead were used for conformal calibration and testing with an even split. Throughout this work, we report model performance on this unseen testing split.

Model Training We fine-tuned a Qwen3-1.7B [30] LLM for 20 epochs to estimate step-wise error likelihoods in failed multi-agent trajectories. Since each trajectory had one step labeled as the decisive error (determined through error injection), we fine-tuned the model f_θ on the ℓ-way classification task to produce a scalar score $f_\theta(c_{i,j})$ for each step j in datapoint i, interpreted as the likelihood that step $c_{i,j}$ contains the decisive error. We write $e_{i,j}$ as the binary indicator of whether step j in datapoint i is the decisive error. The model parameters θ were optimized via binary cross-entropy:

$$\mathcal{L}(\theta) = -\sum_{i}\sum_{j=1}^{\ell_i}\Big[e_{i,j}\log\sigma(f_\theta(c_{i,j})) + (1-e_{i,j})\log(1-\sigma(f_\theta(c_{i,j})))\Big], \tag{17}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid.

We provide more training set up details in Table 6, and report the performance before and fine-tuning the models in Table 7.

Table 7: Performance of Qwen3-1.7B models before and after fine-tuning

Model	AUROC	AUPRC	Accuracy
Pretrained	0.505	0.369	0.509
Fine-tuned	0.762	0.382	0.731

Compute Resources For fine-tuning the 1.7B parameter LLM we used an Nvidia GeForce RTX 5090. 20 epochs of training took roughly one day, and consumed less than 16 GB of GPU memory with batch size 32. All other calls to LLMs used commercial APIs, and we discussed the number of calls needed in Section 5.4. Once scoring calls are made, performing conformal calibration is a trivial computational cost.

C.1.4 Rollback Experiment Implementation

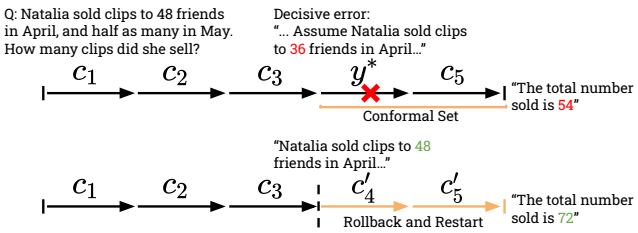

Figure 8: Example rollback scenario with text for the task, decisive error, final answer, and corrected versions after the rollback. We first generate a conformal set for the failed task, roll back the state of the MAS to the first step in the prediction set, and restart the agent with instructions to avoid the same mistakes.

Figure 8 provides a concrete example of the rollback procedure. After identifying a conformal set for a failed trajectory using the LF algorithm, the system rolls back to the earliest step in the set and re-executes the task from that point on with additional context from the failed trace, allowing the MAS to correct the final outcome. Prompt details used during the restart are provided in Section C.3.

C.2 Additional Experimental Results

C.2.1 Scoring Aggregation Methods

In Section 3.2 we discussed ways to tailor LLMs to the error attribution task and generate a step-wise error likelihood score. Step-wise scores then need to be aggregated into a set-wise score, ideally one that obeys monotonicity (Equation (2)). In that section we demonstrated the normalized sum function for aggregation, $g(c_{j:k}) = \frac{1}{\ell}\sum_{i=j}^{k} \text{LLM}(c_i)$, which was used for experiments in the main text. Here we consider alternatives and show empirical results across them.

Max Aggregation Another way to monotonically aggregate step-level scores is by taking the maximum score over a set:

$$g_{\max}(c_{j:k}) = \max_{i\in\{j,\dots,k\}} \text{LLM}(c_i) + \lambda\frac{k-j}{\ell}. \tag{18}$$

Note that as opposed to summation where length normalization is included on the sum to account for trajectories of different lengths ℓ, we do not normalize the max, since the single-step max tends to be similar across trajectories of different lengths. Instead we have included an optional length penalty term, weighted by the hyperparameter λ, to penalize sets that are longer than necessary to contain the highest scoring single step. Below, we also test adding a similar length penalty on the sum aggregation. Max aggregation should be useful when the LLM produces very peaked step-wise scores, as opposed to sum aggregation which can benefit when step-wise scores are more uniform.

Table 8: Removal Rate with the Right Filtering conformal algorithm across various aggregation functions on GSM8K Left Dense dataset.

Aggregation	Hypers	DyLAN			MACNET		
		Naive LLM	Role-prompted	Qwen3-1.7B	Naive LLM	Role-prompted	Qwen3-1.7B
Sum + Length Penalty	$\lambda = 0.01$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.01	0.62±0.02
	$\lambda = 0.02$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.01	0.62±0.02
	$\lambda = 0.05$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.01	0.62±0.02
	$\lambda = 0.1$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.01	0.62±0.01
Max + Length Penalty	$\lambda = 0.01$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.02	0.62±0.02
	$\lambda = 0.02$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.01	0.62±0.02
	$\lambda = 0.05$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.02	0.62±0.01
	$\lambda = 0.1$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.02	0.62±0.01
Normalized LogSumExp	$\beta = 0.1$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.02	0.63±0.02
	$\beta = 1$	0.64±0.02	0.63±0.02	0.65±0.01	0.59±0.01	0.59±0.02	0.63±0.02
	$\beta = 10$	0.63±0.02	0.63±0.02	0.64±0.02	0.60±0.01	0.59±0.01	0.62±0.01

LogSumExp Aggregation As a third alternative, we consider a family of monotonic aggregations that interpolate between the extremes of sum and max by using the LogSumExp function:

$$g_{\text{LSE}}(c_{j:k}) = \frac{1}{\beta}\text{LogSumExp}(\beta\ell \cdot [\text{LLM}(c_j), ..., \text{LLM}(c_k)]) - \log \ell. \tag{19}$$

This aggregation includes the inverse temperature hyperparameter β, where higher values behave more like the max function, while β close to zero behaves like the mean function (but is monotonic, unlike the true mean function). We also add length normalization via ℓ to make scores more similar across trajectories with different lengths.

Experimental Results In Table 8, we compare Max, LogsumExp, and Sum aggregation with the optional length penalty using the RF conformal algorithms on the Left-Dense subset of GSM8k, across three evaluators(Naive LLM, Role-prompted, and the finetuned Qwen3-1.7B) settings and two agent architectures (DyLAN and MACNET). The primary result is that RF is largely insensitive to the details of the scoring function g, from the LLM design, to the aggregation function, or various hyperparameters. This means a simpler scoring function design is sufficient for use with our filtering algorithms.

C.2.2 AgentRx Results

We conduct a complementary evaluation on real-world agent errors using the AgentRx dataset [4] (MIT License). The publicly available, usable subset contains 73 trajectories: 44 from Magentic-One and 29 from τ-bench. We evaluate all five conformal algorithms using the same three scoring functions considered on Who&When: random, naive LLM, and role-prompted scoring. For each run, we randomly select 30 Magentic-One trajectories and 20 τ-bench trajectories for calibration, leaving 23 trajectories for testing. We repeat this stratified splitting procedure 1,000 times at $\alpha = 0.2$.

The results in Table 9 are broadly consistent with our Who&When findings regarding the limited sensitivity of filtering methods to the scoring function. Here, LF achieves the highest mean removal rate across all three scorers, ranging from 0.31 to 0.32. However, the small and heterogeneous evaluation set yields substantial variation across splits. This points to a lack of large-scale and real-world data on agent errors in practice.

Table 9: Removal rates on AgentRx at $\alpha = 0.2$ (target coverage 80%), reported as mean $\pm$ standard deviation over 1,000 random calibration–test splits.

Scoring function	Vanilla	CRSVP	LF	RF	TWF
Random	0.23±0.10	0.12±0.05	0.31±0.08	0.12±0.06	0.17±0.08
Naive LLM	0.18±0.08	0.13±0.05	0.32±0.07	0.12±0.07	0.16±0.08
Role-prompted	0.20±0.09	0.16±0.06	0.32±0.08	0.12±0.07	0.16±0.08

C.3 Prompts Used

Naive LLM Prompts We use following prompt for the naive LLM scoring function.

```
You are an AI assistant tasked with analyzing a segment of a multi-agent
conversation. Multiple agents are collaborating to address a user query, with
the goal of resolving the query through their collective dialogue.
Your primary task is to identify the location of the most critical mistake
within the provided segment.
The problem to address is as follows: {problem}
The final correct answer for the problem is: {answer}
The wrong answer given at the end of the conversation is: {wrong_answer}
Review the following conversation segment: {evaluate_content}
Cut from the entire history:
{chat_segment_content}
Based on your analysis, please respond with ONLY a single probability value
between 0 and 1, representing the estimated probability that the critical error
occurs within this conversation segment. Do not include any explanation or
additional text.
```

Context Engineered Prompts As mentioned earlier, along with the context engineered setting we use four role-based LLMs as part of the scoring function. Here, we provide the prompts used for each LLM.

```
Conservative:
- Attribute an error only when explicit contradiction or logical error is
visible.
- Assign high confidence only if the evidence is direct and quotes are exact.

Liberal:
- Consider potential upstream or downstream causes even if indirect.
- Assign moderate confidence (0.5-0.7) for plausible but not proven causes.

Skeptical:
- Always include at least one alternative explanation and note missing evidence.
- Lower confidence if evidence is incomplete.

Pattern:
- Focus on repeated reasoning or coordination issues.
- Identify patterns across multiple steps rather than isolated mistakes.
```

We force the output into the following schema:

Output Schema

```
{
  "investigation_summary": "short summary",
  "primary_conclusion": {
    "agent": "Agent-X",
    "mistake_step": "int",
    "confidence": "float (0--1)",
    "evidence": ["quoted spans or step references"],
    "reason": "brief reason",
    "alternative_explanations": ["..."]
  }
}
```

The final prompt for each role is as follows:

```
Problem to solve: {problem}
Expected correct answer: {answer}
Conversation segment under review: {evaluate_content}
Full chat context: {chat_segment_content}
ROLE: {role} Analyst
{role_instructions}
TASK: Identify whether this segment likely contains the key reasoning error
that led to the wrong final answer. Provide your analysis strictly in JSON
format matching the schema below. Quote evidence spans exactly from the text
and assign a numeric confidence between 0 and 1.
Confidence mapping: 0.0-0.2 = Very unlikely, 0.2-0.4 = Unlikely, 0.4-0.6 =
Possible, 0.6-0.8 = Likely, 0.8-1.0 = Very likely.
Return ONLY valid JSON (no extra text): {schema}
```

Conformal Rollback Prompts The conformal rollback instructions provided to the MAS are as follow:

```
NOTE: The following conversation history (Node 0 to Node {cut_index}) contains
wrong information. Please avoid making the same mistakes if you detect the
mistake:
{wrong_conversation_content}
With the following mostly correct conversation history: {prev_context}
Solve this problem step by step: {task_query}
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction summarize the claims of the paper, which are then followed up on in Section 3 and 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed throughout the text, and recounted in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: While the main text in Section 3 includes proof sketches, full formal proofs are included in Appendix A. Assumptions are always stated.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The most important experimental details are given in Section 4 and Section 5, while additional detail for complete reproducibility is given in Section C. Code is attached as supplemental material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is attached as supplemental material, and details to reproduce our results are continued in Section C.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The only model trained is a fine-tuned LLM, with details given in Section C. Data splits and hyperparameters are further discussed in Section C as well.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper generally reports 1 standard deviation on the results as error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computing resources used for this work are minimal. The only intensive component is fine-tuning an LLM, which was discussed in Section C. Other aspects of the paper used API-based models with minimal additional compute for conformal calibration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and followed the ethics guidelines.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: There are minimal societal impacts expected from this work. We presented statistical theory, and methods to improve the oversight and understanding of agentic systems.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release artifacts with a high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all relevant work that we built on. All datasets we used come under the MIT license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: Although code is provided as supplemental material, we have not released any artifacts.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not use human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not use human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We study LLM-based multi-agent systems, and described how LLMs are used.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.