Beyond LLM-Based Reasoning: Lightweight GNNs for Agent Failure Attribution

Ting-Wei Li
University of Illinois
Urbana-Champaign, IL USA
twli@illinois.edu

Yuanchen Bei
University of Illinois
Urbana-Champaign, IL USA
bei4@illinois.edu

Xiao Lin
University of Illinois
Urbana-Champaign, IL USA
xiaol13@illinois.edu

Hanghang Tong
University of Illinois
Urbana-Champaign, IL USA
htong@illinois.edu

Abstract

Large language model (LLM)-based multi-agent systems (MAS) often exhibit complex failure modes, which frequently cause agents to produce incorrect outcomes. This motivates the task of *Agent Failure Attribution (AFA)*: given a failed multi-agent trajectory, identify the faulty agents and their corresponding error types. Existing approaches predominantly rely on LLMs to perform failure attribution, either through direct prompting, fine-tuning on synthetic data or complex agentic pipelines. While effective, these methods incur substantial computational overhead due to long-context processing, expensive post-training and handcrafted workflows. Moreover, empirical evidence shows that even state-of-the-art models achieve limited accuracy on existing benchmarks, suggesting that scaling model size alone is insufficient. In this work, we revisit this task and question the necessity of such expensive generative solutions. We introduce `AFANet`, a lightweight graph-based framework that models interaction trajectories through step-level semantic signals and agent-level relationships. We show that with significantly fewer parameters and near-zero inference cost, `AFANet` (i) matches or outperforms LLM-based baselines, including fine-tuned models on in-domain benchmarks, (ii) maintains robust performance across different GNN architectures and (iii) can be further improved with inexpensive test-time adaptation on the OOD benchmark. Our results suggest that a lightweight, structured modeling offers a highly scalable and competitive alternative to heavy LLM reasoning for agent failure attribution.

1 Introduction

LLM-based multi-agent systems [13, 7, 27] have emerged as a powerful paradigm for solving complex tasks through coordinated reasoning, tool use, and interaction across multiple agents. Despite their strong capabilities, these systems often exhibit *complex failure modes* [3, 35, 25, 14], where errors introduced by one agent propagate through subsequent interactions and ultimately lead to incorrect outcomes. This motivates the task of *Agent Failure Attribution (AFA)* [3, 35, 5, 10]: given a failed multi-agent interaction trajectory, identify the faulty agents and their corresponding error types responsible for the failure. However, this task is inherently challenging due to long interaction horizons, complex dependencies across agents and the ambiguous nature of error propagation. Even state-of-the-art reasoning models or powerful proprietary models achieve limited accuracy on this task, highlighting the difficulty of identifying root causes within long trajectories [10].

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

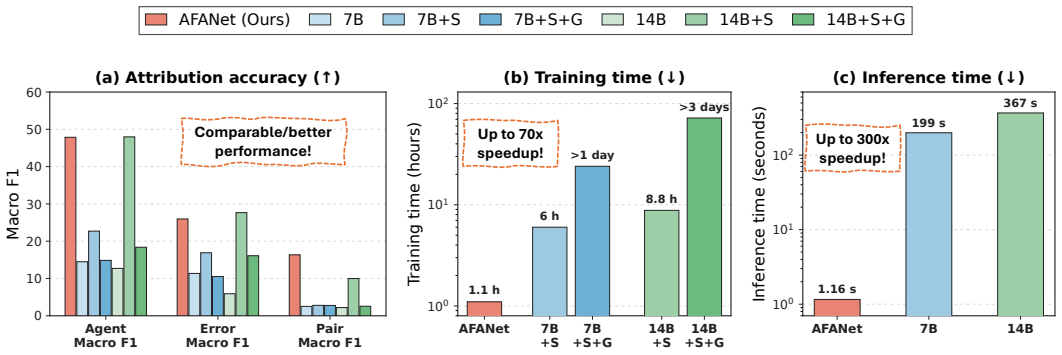


Figure 1: Attribution accuracy and efficiency comparison between our proposed `AFANet` and LLMs. `AFANet` can achieve similar or better performance against LLMs while incurring significantly lower training and inference cost. We consider the base models: `Qwen-2.5-7B/14B-Instruct`, denoted as **7B/14B**, respectively. We also include their post-trained variants obtained through SFT (**S**) and subsequent GRPO training (**G**), denoted by **7B+S/14B+S** and **7B+S+G/14B+S+G**.

Existing approaches predominantly treat failure attribution as a *generative reasoning problem*, relying on LLMs to analyze interaction trajectories and directly generate the answers. These methods typically involve direct prompting [35], fine-tuning on synthetic failure data [10, 34, 33] or complex multi-stage agentic pipelines [32, 2, 26, 20]. While these approaches have improved performance, they suffer from several fundamental limitations. Firstly **(Efficiency Bottleneck)**, they incur substantial computational overhead due to long-context processing, repeated inference and expensive training procedures. Secondly **(Architectural Complexity)**, they introduce significant system complexity through multi-stage workflows and handcrafted pipelines. Lastly **(Systematical Ineffectiveness)**, despite these efforts, their performance remains limited, sometimes defeated by random baselines, suggesting that scaling model size alone is insufficient for solving this task. These observations raise a fundamental question:

Is heavy LLM reasoning necessary for agent failure attribution?

In this work, we challenge this prevailing paradigm and argue that agent failure attribution can be solved much efficiently with lightweight models. Instead of relying on expensive LLM-based reasoning, we propose `AFANet`, a lightweight graph neural network (GNN) [30] that models interaction trajectories through step-level semantic and agent-level structural signals. By explicitly encoding temporal dependencies and inter-agent interactions, `AFANet` captures the underlying structure of failure propagation. Specifically, `AFANet` addresses the core challenges of agentic failure attribution in a principled manner. First, graph-based representations naturally model error propagation and inter-agent dependencies, enabling accurate identification of root causes. Second, the model operates with significantly reduced computational cost, avoiding both long-context inference and expensive post-training. Through extensive experiments, we demonstrate that `AFANet` matches or outperforms strong LLM-based baselines, including supervised fine-tuned and RL-enhanced models, while using significantly fewer parameters and achieving near-zero inference cost (as shown in Figure 1). We further show that `AFANet` is robust to architectural choices and can be improved through inexpensive test-time adaptation on the OOD dataset. We summarize our contributions are as follows:

- We introduce a new perspective on agent failure attribution: graph-based modeling of failure trajectories provides an efficient and competitive alternative against LLM-based reasoning.
- We propose `AFANet`, a lightweight framework that models multi-agent failure propagation through a turn-level conversation graph, integrating turn-level semantics and temporal/intra-agent dependencies via graph message passing.
- We demonstrate that with substantially lower training and inference cost, `AFANet` achieves competitive performance compared to LLMs.
- We conduct comprehensive ablation study and show that `AFANet` remains strong under different architectural choices and can be improved via inexpensive test-time adaptation.

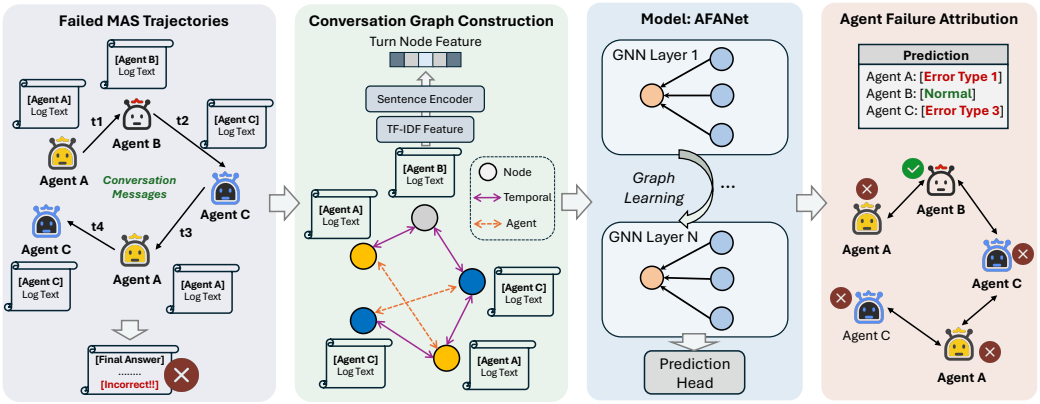


Figure 2: Overall pipeline. Given a failed multi-agent system trajectory, we first transform it into a conversation graph with turn-level textual features and temporal/agent-level connections. The graph will then be passed through our proposed `AFANet` for agent failure prediction.

2 Preliminaries

2.1 Multi-Agent System Trajectories

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$ denote a set of LLM-based agents. Given a user query $q \in \mathcal{Q}$, a multi-agent system attempts to solve the task and produces an interaction trajectory $\mathcal{T} = \{(a_{i_t}, x_t)\}_{t=1}^{T}$, where T is the trajectory length, $a_{i_t} \in \mathcal{A}$ is the agent selected at step t, and $x_t \in \mathcal{X}$ is the content generated by that agent. Each trajectory $\mathcal{T}$ is associated with an outcome variable $o = g(\mathcal{T}, q) \in \{0, 1\}$, where g is the outcome verifier and $o = 0/1$ indicates task failure/success, respectively.

2.2 Agent Failure Attribution (AFA)

Let $\mathcal{E} = \{e_1, e_2, \ldots, e_K\}$ denote a predefined set of error types and K is the number of error types. We define the label space as the Cartesian product over agent space and error space: $\mathcal{L} = \mathcal{A} \times \mathcal{E}$. Given a failed trajectory $\mathcal{T}$, the task of **Agent Failure Attribution** (AFA) is to predict a subset of labels $\mathcal{Y} \subseteq \mathcal{L}$, where each $(a_j, e_k) \in \mathcal{Y}$ indicates that agent a_j commits an error of type e_k that contributes to the failure. We thus formulate AFA as a multi-way classification problem for each agent and our goal is to learn a mapping function $h : \mathcal{T} \to \{0, 1\}^{M \times (K+1)}$ where the first class corresponds to the *clean* state and the remaining K classes correspond to each error type.

3 Methodology

In this section, we propose a lightweight graph-based model, `AFANet`, to tackle agent failure attribution. Given a conversation, we first construct a turn-level graph with temporal and intra-agent connections (Sec. 3.1). We then introduce our proposed model, `AFANet`, a simple GNN that combines deviation/statistical signals and contextual semantics to produce agent-level representations (Sec. 3.2). Then, we detail the learning objective (Sec. 3.3) and finally describe the inference procedure of `AFANet` (Sec. 3.4). The overall pipeline is detailed in Figure 2.

3.1 Conversation Graph Construction

We first transform each multi-agent conversation into a heterogeneous graph with turn-level nodes. The key intuition is that failures in multi-agent systems are not only reflected in the semantic content of individual turns, but also in how an agent's contributions evolve over time and interact with other agents. Therefore, we represent it as a heterogeneous graph whose nodes are conversation turns and whose edges encode temporal and intra-agent dependencies. We detail the construction of node features and edges in the following paragraphs.

Turn node representation. Given a failed trajectory $\mathcal{T} = \{(a_{i_t}, x_t)\}_{t=1}^{T}$, we construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v_t \in \mathcal{V}$ corresponds to one turn (a_{i_t}, x_t) in the conversation and the number of nodes is $|\mathcal{V}| = T$. Inspired by recent anomaly detection literature [21, 19], we aim to capture abnormal deviation and statistical patterns to derive useful turn-level features for solving AFA. The key intuition is that faulty agent behavior is often reflected through inconsistencies in interaction dynamics rather than only the semantic content of individual turns.

To obtain such signals efficiently, we first construct per-conversation TF-IDF representations by treating each turn as a document and the entire trajectory as the corpus (more details in Appendix A). We then apply truncated SVD to obtain low-dimensional dense representations: $\mathbf{X}_{\text{svd}} \in \mathbb{R}^{T \times r}$, where r is the rank size. Based on $\mathbf{X}_{\text{svd}}$, we compute a set of deviation features that captures how each turn diverges from the contexts in the conversation. We concatenate these deviation features to obtain $\mathbf{x}_t^{\text{dev}}$ for each turn, and stack them across all turns: $\mathbf{X}_{\text{dev}} = [\mathbf{x}_{\text{dev}}^1 | \cdots | \mathbf{x}_{\text{dev}}^T] \in \mathbb{R}^{T \times d_{\text{dev}}}$, where d_{dev} is the number of deviation features.

In addition, we consider statistical features $\mathbf{X}_{\text{stat}} \in \mathbb{R}^{T \times d_{\text{stat}}}$ (d_{stat} denotes the number of statistical features) that encode non-semantic properties such as positional encoding. Finally, we consider dense semantic features obtained from pretrained sentence encoders (e.g., sentence-BERT [18]) to capture richer contextual meaning, leading to $\mathbf{X}_{\text{dense}} \in \mathbb{R}^{T \times d_{\text{dense}}}$, where d_{dense} is the embedding dimension. Stacking these components together, we obtain the initial feature matrix for turn nodes: $\mathbf{X} = \left[\mathbf{X}_{\text{dev}} \,\middle|\, \mathbf{X}_{\text{stat}} \,\middle|\, \mathbf{X}_{\text{dense}}\right] \in \mathbb{R}^{T \times d}$, where $d = d_{\text{dev}} + d_{\text{stat}} + d_{\text{dense}}$ is the number of feature dimension. Each turn node is therefore associated with a feature vector $\mathbf{x}_t = \mathbf{X}[t, :] \in \mathbb{R}^d$. The formal definitions of deviation and statistical features are deferred to Appendix A.

Edge construction. We construct two types of connections: *temporal progression* and *intra-agent dependency*. Firstly, to preserve the sequential nature of the conversation, we add bidirectional temporal edges between adjacent turns: $(v_t, v_{t+1}) \in \mathcal{E}_{\text{seq}}^{\rightarrow}, (v_{t+1}, v_t) \in \mathcal{E}_{\text{seq}}^{\leftarrow}$. Secondly, to capture long-range consistency across non-adjacent turns induced by the same agent, we connect turns produced by the same agent. For two turns $u < v$ such that $a_{i_u} = a_{i_v}$, we add $(v_u, v_v) \in \mathcal{E}_{\text{agent}}^{\rightarrow}, (v_v, v_u) \in \mathcal{E}_{\text{agent}}^{\leftarrow}$. Therefore, the full edge set is $\mathcal{E} = \mathcal{E}_{\text{seq}}^{\rightarrow} \cup \mathcal{E}_{\text{seq}}^{\leftarrow} \cup \mathcal{E}_{\text{agent}}^{\rightarrow} \cup \mathcal{E}_{\text{agent}}^{\leftarrow}$.

3.2 `AFANet`: A Lightweight GNN for Agent Failure Attribution

After constructing the turn-level graph $\mathcal{G}$, we introduce `AFANet`, a model that captures both structural and semantic signals to solve AFA. Specifically, turn features are first projected into a hidden space , followed by graph neural network layers that propagate neighboring turn information along the conversation structure. Finally, agent-level representations are obtained and mapped to prediction scores that quantify the likelihood of each agent being associated with certain error types. We detail the procedure of `AFANet` as follows.

Input projection. We represent each turn node t in the conversation graph with a concatenated feature vector $\mathbf{x}_t \in \mathbb{R}^d$. Before graph message passing, we project these initial node features into a hidden dimension d_{hidden}. This is achieved via a projection module consisting of a linear transformation, a ReLU activation, and layer normalization:

$$\mathbf{h}_t^{(0)} = \text{LN}(\text{ReLU}(\mathbf{W}_{\text{in}}\mathbf{x}_t + \mathbf{b}_{\text{in}})),$$

where $\mathbf{W}_{\text{in}}$ is learnable weight matrix, $\mathbf{b}_{\text{in}}$ is learnable bias and LN denotes layer normalization.

Graph Neural Network blocks. To capture contextual dependencies and the structural flow of the conversation, we apply L layers of message passing over the conversation graph. We also incorporate residual skip connections at each layer. The node representation at layer $\ell + 1$ is computed as:

$$\mathbf{h}_t^{(\ell+1)} = \mathbf{h}_t^{(\ell)} + \text{GNN}^{(\ell)}\left(\mathbf{h}_t^{(\ell)}, \{\mathbf{h}_u^{(\ell)} : (u, t) \in \mathcal{E}\}\right),$$

where $\mathcal{E}$ represents the set of edges in the graph. After L layers, we obtain the per-turn node representation $\mathbf{h}_t^{\text{gnn}} = \mathbf{h}_t^{(L)} \in \mathbb{R}^{d_{\text{hidden}}}$. Note that the GNN layer can be instantiated with any architecture (e.g. GCN [9], GAT [23] and GraphSAGE [8]).

Agent-level pooling. Since the final classification objective is performed at the agent level, we aggregate the turn-level representations belonging to each respective agent. For a given agent a_j

with the corresponding turn index set $\mathcal{I}_j$, we apply a dual pooling strategy that concatenates both the mean-pooled and max-pooled representations for better expressiveness:

$$\mathbf{z}_j = \left[\frac{1}{|\mathcal{I}_j|} \sum_{t \in \mathcal{I}_j} \mathbf{h}_t^{\text{gnn}} \parallel \max_{t \in \mathcal{I}_j} \mathbf{h}_t^{\text{gnn}} \right].$$

Prediction head. Given the pooled agent representation $\mathbf{z}_j \in \mathbb{R}^{2d_{\text{hidden}}}$, we map it to the prediction space through a multi-layer projection head. We first obtain an intermediate hidden representation for each agent:

$$\mathbf{h}_j = \text{LN}(\text{ReLU}(\mathbf{W}_1 \mathbf{z}_j + \mathbf{b}_1)),$$

where $\mathbf{h}_j \in \mathbb{R}^{d_{\text{hidden}}}$. We then apply a bottleneck projection followed by a non-linear activation and dropout:

$$\tilde{\mathbf{h}}_j = \text{Dropout}(\text{ReLU}(\mathbf{W}_{\text{mid}} \mathbf{h}_j + \mathbf{b}_{\text{mid}})),$$

where $\tilde{\mathbf{h}}_j \in \mathbb{R}^{d_{\text{mid}}}$ and d_{mid} is the bottleneck dimension size. Finally, we map to the prediction logits:

$$\mathbf{s}_j = \mathbf{W}_{\text{out}} \tilde{\mathbf{h}}_j + \mathbf{b}_{\text{out}},$$

where $\mathbf{s}_j \in \mathbb{R}^{K+1}$ represents the un-normalized scores over K classes. The $K+1$ classes correspond to class 0 for a clean type (i.e. no error occurs) and classes $1, \ldots, K-1$ for different error types.

3.3 Training Objectives

Given the agent-level representations $\{\mathbf{h}_j\}_{j=1}^M$, we optimize `AFANet` using a two-term objective that combines (i) agent-level fault detection and (ii) error-type classification.

Agent-level objective. We first derive a binary fault logit a_j from the prediction scores $\mathbf{s}_j \in \mathbb{R}^{K+1}$ for each agent j:

$$a_j = \log \sum_{k=1}^{K} \exp(s_{j,k}) - \log \exp(s_{j,0}).$$

We then apply a weighted binary cross-entropy loss with logits:

$$\mathcal{L}_{\text{agent}} = \frac{1}{M} \sum_{j=1}^{M} \text{BCEWithLogits}(a_j, y_j^{\text{binary}};\, w^+),$$

where $y_j^{\text{binary}} = \mathbb{I}[y_j \neq 0]$ denotes whether agent j is faulty and the positive-class weight is defined as $w^+ = \frac{n_{\text{clean}}}{n_{\text{faulty}}}$, where $n_{\text{clean}}, n_{\text{faulty}}$ is the number of non-faulty and faulty agents summing over the conversations in the batch.

Error-level objective. For faulty agents $\mathcal{F} = \{j \mid y_j \neq 0\}$, we apply a multi-class cross-entropy loss over the error-type subspace:

$$\mathcal{L}_{\text{fm}} = \frac{1}{|\mathcal{F}|} \sum_{j \in \mathcal{F}} \text{CE}([s_{j,1}, \ldots, s_{j,K}], y_j - 1).$$

Overall objective. The final loss is simply the weighted sum over the above two terms: $\mathcal{L} = \lambda_{\text{agent}} \mathcal{L}_{\text{agent}} + \lambda_{\text{fm}} \mathcal{L}_{\text{fm}}$, where $\lambda_{\text{agent}}, \lambda_{\text{fm}}$ are hyper-parameters.

3.4 Inference with `AFANet`

We first compute the score vector $\mathbf{s}_j \in \mathbb{R}^{K+1}$ for each agent, where $s_{j,0}$ denotes the *clean* class. We derive an agent-level fault score: for each agent j, $\text{score}_j^{\text{agent}} = \mathbb{P}[\text{agent}_\text{j} \text{ is faulty}] = \frac{\sum_{k=1}^{K} \exp(s_{j,k})}{\sum_{k=0}^{K} \exp(s_{j,k})}$. During inference, we consider both threshold-based and ranking-based decoding. For *threshold-based decoding*, we utilize the validation set to sweep the best threshold τ that maximizes agent-level metric, and then fix τ to select τ_{fm} that maximizes error-level metric. We predict faulty agents as $\hat{\mathcal{F}} = \{j \mid \text{score}_j^{\text{agent}} \geq \tau\}$. For each $j \in \hat{\mathcal{F}}$, we assign $\hat{e}_j = \arg\max_{k \geq 1} s_{j,k}$ and retain $(a_j, \hat{e}_j)$ only if $s_{j,\hat{e}_j} \geq \tau_{\text{fm}}$. For *ranking-based decoding*, we rank agents by $\text{score}_j^{\text{agent}}$ and directly select the top-ranked agent(s), assigning each selected agent the error type with the largest score.

Table 1: Performance comparison on AEGIS-Bench and Who&When. Best and second-best results are in **bold** and underline. All scores are percentages (%). For the Qwen-based models, `Instruct` are abbreviated as **It**, while `Non-Thinking` and `Thinking` modes are abbreviated as **NT** and **T**. Their SFT and subsequent GRPO-trained variants are denoted by **+S** and **+S+G**. Rows marked with ** indicate models re-implemented and evaluated by us under the same setting, while the remaining rows are adapted from [10].

Category	Method	AEGIS-Bench						Who&When						Avg.
		Agent		Error		Pair		Agent		Error		Pair		
		μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	
Baseline	Random	4.54	3.56	11.23	11.15	0.33	0.21	1.06	0.83	8.74	7.14	0.11	0.05	4.08
Pre-trained LLMs	Qwen2.5-7B-It	27.55	14.49	14.96	11.36	5.02	2.52	40.92	23.50	3.64	1.77	2.31	1.14	12.43
	Qwen2.5-7B-It**	26.38	16.15	15.14	11.54	3.67	1.76	46.25	37.85	3.26	1.94	2.06	2.44	14.04
	Qwen2.5-14B-It	35.78	12.71	20.24	5.91	5.47	2.20	49.88	33.19	1.56	1.35	0.00	0.00	13.99
	Qwen2.5-14B-It**	36.72	19.62	18.59	16.49	5.86	2.61	45.47	44.83	0.00	0.00	0.00	0.00	15.85
	Qwen3-8B-NT	21.34	8.16	15.81	13.89	3.96	1.40	27.78	17.64	3.88	1.91	3.88	1.81	10.12
	Qwen3-8B-T	34.63	9.01	17.48	14.31	4.42	1.52	37.91	27.58	4.65	2.21	1.95	1.10	13.06
Fine-tuned LLMs	Qwen2.5-7B-It+S	60.03	22.70	19.61	16.90	5.05	2.80	43.51	32.51	6.77	4.20	1.26	0.52	17.99
	Qwen2.5-7B-It+S**	37.93	22.14	17.69	13.85	4.48	2.02	**56.77**	**65.81**	5.42	4.75	3.94	**4.58**	19.97
	Qwen2.5-7B-It+S+G	35.43	14.86	17.21	10.54	7.11	2.77	50.77	30.14	3.86	2.30	2.31	1.19	14.87
	Qwen2.5-14B-It+S	**76.53**	**47.97**	**27.53**	**27.66**	16.62	9.99	51.14	36.94	9.87	7.77	4.03	2.08	**26.51**
	Qwen2.5-14B-It+S**	41.04	21.80	17.16	13.83	4.38	2.02	49.37	64.90	1.77	1.73	0.41	0.25	18.22
	Qwen2.5-14B-It+S+G	49.74	18.38	21.19	16.10	6.84	2.55	54.43	40.88	4.15	2.67	2.45	1.49	18.41
	Qwen3-8B-NT+S	64.79	38.96	20.37	20.36	9.68	5.73	45.48	30.77	8.00	5.29	5.17	2.33	21.41
	Qwen3-8B-NT+S+G	45.91	17.39	20.89	15.15	6.94	2.82	50.94	38.26	2.21	1.68	2.21	1.45	17.15
	Qwen3-8B-T+G	36.11	15.73	17.94	12.03	4.41	1.66	53.12	40.52	11.25	6.91	**8.10**	3.19	17.58
Proprietary LLMs	GPT-4.1	37.48	11.12	20.65	15.75	7.44	2.27	42.29	28.93	7.00	5.84	3.36	1.16	15.27
	GPT-4o-mini	38.54	14.72	19.95	16.02	5.76	1.63	47.42	34.21	5.26	3.33	2.11	0.98	15.83
	o3	40.31	23.27	22.37	16.76	7.86	2.27	53.10	42.55	**14.88**	8.63	7.41	3.98	20.24
	Gemini-2.5-Flash	42.02	16.45	23.47	19.85	6.99	2.76	55.56	36.98	11.94	7.96	7.32	3.33	19.55
	Gemini-2.5-Pro	41.32	16.15	19.93	16.29	6.96	2.88	53.11	34.92	11.07	8.11	6.81	2.69	18.35
	Claude-Sonnet-4	40.73	15.51	21.21	16.55	7.68	2.34	44.76	37.23	13.33	**9.23**	6.77	2.66	18.16
Ours	`AFANet`	74.16	47.86	27.01	25.96	**17.42**	**16.35**	37.93	20.26	13.79	6.05	6.90	4.16	24.82

4 Experiment

In this section, we conduct extensive experiments to evaluate the effectiveness of `AFANet` against LLM-based baselines. Our experiments are designed to answer the following research questions:

- (**RQ1**): How does `AFANet` compare to LLM-based methods w.r.t. efficacy and efficiency?
- (**RQ2**): How does `AFANet` perform under different GNN backbones and hyper-parameters?
- (**RQ3**): How does `AFANet` perform across domains and how to improve it?

4.1 Experimental Setup

Datasets. We conduct experiments on the AEGIS-Bench [10][1] and the Who&When [35] datasets[2]. AEGIS-Bench serves as the in-domain dataset and has pre-defined train/val/test splits. On the other hand, Who&When is utilized as the out-of-distribution datasets and has exactly one faulty agent per conversation. During training phase, the training and validation splits of AEGIS-Bench are used to train and validate `AFANet`; the test split of AEGIS-Bench and the entire Who&When are used to report the evaluation performances. The data statistics is in Appendix B.

Baselines. We primarily compare `AFANet` against a broad range of LLM-based methods, including pre-trained, fine-tuned and proprietary LLMs. Following Kong et al. [10], we adopt the *All-at-Once* prompting strategy for all LLM baselines, where the model is provided with the user query together with the complete failure trajectory, and is directly asked to identify the faulty agent as well as the corresponding error type *all at once*. The evaluated LLM backbones include `Qwen2.5` [22], `Qwen3` [31], `GPT-4.1` [16], `GPT-4o-mini` [15], `o3` [17], `Gemini-2.5-Flash/Pro` [4] and

[1] `https://huggingface.co/datasets/Fancylalala/AEGIS`
[2] `https://huggingface.co/datasets/Kevin355/Who_and_When`

Table 2: Efficiency comparison between AFANet and LLM-based baselines. For AFANet, preprocessing time summed over train/validation/test graph construction time. Inference time is reported as in-domain / OOD, respectively.

Method	Training Time	Inference Time	Preprocessing Time	Trainable Params
7B SFT	6 hrs	199s / 108s	–	7B
7B SFT + GRPO	> 26 hrs	199s / 108s	–	7B
14B SFT	8.8 hrs	367s / 231s	–	14B
14B SFT + GRPO	> 74 hrs	367s / 231s	–	14B
AFANet	1.1 hrs	1.16s / 0.37s	80.8s	65K

Claude-Sonnet-4 [1]. We also consider the supervised fine-tuning (**+S**) and subsequent GRPO-trained (**+S+G**) variants on small-scale Qwen2.5 and Qwen3 models.

Evaluation Metrics. Following Kong et al. [10], we evaluate the attribution accuracy at three levels of granularity: **Pair-level** (correct agent-error pairs), **Agent-level** (correct faulty agents, ignoring the error types) and **Error-level** (correct error types, ignoring faulty agents). For each level, **Micro-F1** (μF1) and **Macro-F1** (MF1) are reported. Micro-F1 pools predictions over all samples to compute a single global score. In contrast, Macro-F1 calculates the F1 score separately for each class (e.g., each of the error types) and then averages them uniformly. The details of the error type taxonomy is in Appendix C.

Implementation Details. For AFANet, we default its backbone GNN to a 2-layer GCN [9] with a hidden dimension of 64 and a bottleneck dimension of 32. The model is trained using the Adam optimizer with a learning rate of 1×10^{-2} for up to 100 epochs, with early stopping based on validation performance using a patience of 10 epochs. We set the batch size to 2000 and apply dropout rate 0.1. We employ all-MiniLM-L6-v2 [18] as the embedding model to encode turn node contexts. The best model is selected based on the validation pair-level Micro-F1 score. For AFANet inference, we utilize Micro-F1 to select agent threshold τ and error threshold τ_{fm} on AEGIS-Bench; we utilize ranking-based decoding on Who&When to capture the single error nature. All experiments for AFANet are conducted on an NVIDIA V100 GPU. For the LLM-based methods, we fine-tune Qwen2.5-7B-Instruct & Qwen2.5-14B-Instruct ourselves and evaluate them under the same setting. The versions we re-implement are marked with ** in Table 1. The prompts and the evaluation scripts for the re-implementation are from the AEGIS [10] codebase[3].

4.2 Main Results

AFANet is comparable to or outperforms LLMs. We present our main results in Table 1. AFANet achieves competitive or superior average performance compared to pretrained, post-trained and proprietary LLM baselines over two in-domain and OOD datasets. In particular, despite not relying on LLM-based reasoning, AFANet is highly effective w.r.t. the pair-level metrics.

We note that pair-level attribution is the most challenging and practically important, since it requires simultaneously identifying both the faulty agent and the correct error type. On AEGIS-Bench, AFANet achieves top-1 pair-level performances. More importantly, under the OOD setting on Who&When, AFANet remains highly-ranked on pair-level attribution, demonstrating strong robustness under distribution shift. We hypothesize that this advantage mainly comes from modeling structural interaction dynamics and consistency patterns, which remain relatively stable across trajectories; however, post-trained approaches may overfit to surface-level semantic patterns or dataset-specific failure distributions.

AFANet is much more efficient than LLMs. As shown in Figure 1 and Table 2, AFANet achieves significantly lower training time, inference time and trainable parameters compared to LLM counterpart. The preprocessing time of AFANet is also neglectable (i.e. the time for conversation graph construction over all splits).

[3] https://github.com/kfq20/AEGIS

Table 3: Backbone sensitivity study of AFANet. Best and second-best results are in **bold** and underline. We show that AFANet remains robust across different GNN architectures and model depths. All scores are percentages (%). The reference model uses a 2-layer GCN with both edge types.

Backbone	AEGIS-Bench						Who&When						Avg.
	Agent		Error		Pair		Agent		Error		Pair		
	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	
Main Model													
GCN (2-layer, all edges)	74.16	47.86	**27.01**	**25.96**	17.42	**16.35**	**37.93**	20.26	**13.79**	**6.05**	**6.90**	**4.16**	**24.82**
GNN Backbone													
GAT (2-layer, all edges)	71.92	43.31	24.23	23.13	16.57	13.38	34.48	**25.57**	7.47	3.67	0.46	0.25	22.04
GraphSAGE (2-layer, all edges)	**74.34**	**50.76**	24.94	24.68	16.42	14.30	30.46	23.69	8.05	4.53	3.45	1.55	23.10
GNN Depth													
GCN (1-layer, all edges)	71.17	42.17	23.23	21.88	16.46	14.39	34.48	24.78	5.17	3.17	1.15	0.85	21.58
GCN (3-layer, all edges)	72.84	44.25	25.05	24.61	**17.96**	14.22	31.61	20.45	9.77	5.62	3.45	1.85	22.64

Table 4: Ablation study of AFANet. Best and second-best results are in **bold** and underline. We show that graph structure, edge design and each feature component all contribute to AFANet's performance. All scores are percentages (%). The reference model uses a 2-layer GCN with both edge types.

Setting	AEGIS-Bench						Who&When						Avg.
	Agent		Error		Pair		Agent		Error		Pair		
	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	μF1	MF1	
Main Model													
GCN (all edges)	74.16	47.86	**27.01**	**25.96**	17.42	16.35	**37.93**	20.26	**13.79**	**6.05**	**6.90**	**4.16**	**24.82**
Edge Type Ablation													
GCN (no edges)	71.26	43.21	24.65	23.54	**18.49**	15.03	32.18	25.10	8.62	5.55	3.45	2.39	22.78
GCN (only same-agent edges)	70.53	42.74	22.32	21.58	16.31	12.23	36.78	26.21	6.32	2.95	4.60	3.16	22.14
GCN (only temporal edges)	**74.82**	**53.60**	24.82	24.39	17.54	**16.52**	35.06	20.41	5.75	3.41	4.60	1.99	23.58
Model Design Ablation													
W/o dev/stat features	65.90	35.47	23.59	22.25	14.00	9.80	**37.93**	**32.00**	5.75	2.89	1.72	1.14	21.04
W/o sentence embedding	73.58	48.18	23.62	22.14	14.78	10.16	**37.93**	21.76	8.05	5.44	2.30	1.33	22.44
W/o GNN	71.11	43.80	26.34	25.61	17.81	14.93	26.44	18.86	11.49	5.68	2.87	1.89	22.24

4.3 Discussions

Backbone sensitivity. We present backbone sensitivity in Table 4 to evaluate the robustness of AFANet under different architectural choices. We observe that the overall performance remains relatively stable across different GNN architectures and layer configurations. These observations suggest that the effectiveness of AFANet is not tied to a specific GNN implementation or carefully tuned depth, indicating good architectural robustness.

Ablation. In Table 4, we further study the impact of connection types and some key model designs. Firstly, removing all edges consistently degrades performance, demonstrating that relational structure is important for agent failure attribution. Secondly, using only same-agent edges or only temporal edges also leads to noticeable performance drops compared to the full graph, suggesting that both temporal interaction patterns and intra-agent behavioral consistency provide complementary signals.

For model design ablation, we observe that removing deviation/statistical features causes large overall performance degradation, particularly on pair-level metrics, supporting our hypothesis that faulty agents are often characterized by abnormal interaction dynamics and consistency violations. Removing the GNN module also harms performance, showing the importance of explicit structured modeling.

Generalization study. While AFANet demonstrates strong performance on the in-domain benchmark, generalizing agent failure attribution under severe distribution shift remains an inherently

Table 5: Test-time adaptation results on Who&When. Best and second-best results are in **bold** and underline. We observe that pair-level adaptation achieves the strongest overall gains.

Category	Method	Agent		Error		Pair		Avg.
		μF1	MF1	μF1	MF1	μF1	MF1	
Original	Base	38.51	22.37	9.77	5.05	5.17	2.92	13.96
TTA	Agent	35.06	21.42	**14.37**	5.56	6.32	4.40	14.52
	Pair	35.06	20.15	**14.37**	**5.77**	**7.47**	**4.96**	14.63
	Pair-Faulty	**39.08**	**22.47**	11.49	5.06	6.32	4.06	**14.75**

challenging problem. As a case study, we investigate whether lightweight test-time adaptation (TTA) can improve `AFANet` under OOD settings without requiring retraining or additional supervision.

Motivated by entropy-minimization (EM) based test-time adaptation such as TENT [24], we explore whether `AFANet` can adapt to OOD trajectories by encouraging more confident predictions at inference time. We consider three adaptation objectives at different prediction granularities as follows: (i) **Agent-level EM** minimizes the binary entropy of agent faulty probability; (ii) **Pair-level EM** encourages confident predictions over pairwise label space; and (iii) **Faulty-agent pair-level EM** applies pair-level entropy minimization only to agents that are likely to be faulty. The TTA procedure and the detailed formulas of these strategies are in Appendix F. Table 5 shows that lightweight test-time adaptation can consistently improve OOD performance.

5 Related Work

Direct prompting. Early approaches to agent failure attribution primarily rely on prompting pretrained LLMs to directly analyze interaction trajectories [35, 5]. These methods treat attribution as a reasoning task over long execution logs. However, these approaches rely on costly LLM inference and often require carefully designed prompting strategies.

Finetuning-based methods. Another line of work focuses on training specialized attribution models using synthetic data with labeled faulty trajectories [34, 33, 10]. These methods generate large-scale annotated failure trajectories through techniques such as error injection, counterfactual replay, or graph-guided synthesis, and subsequently post-train models using supervised fine-tuning and reinforcement learning. However, these methods introduce significant overhead in training cost and they still inherit the limitations of direct LLM inference at test time.

Agentic systems. Some works explores more complex failure attribution frameworks. In this category, some approaches employ more sophisticate prompting techniques [37], some construct structured representations such as causal graphs [26], hierarchical context models [2] and complex patterns [6, 20] to better capture inter-agent dependencies, while others incorporate memory mechanisms to reuse previously observed failure patterns [32].

6 Conclusion

We revisit agent failure attribution in LLM-based multi-agent systems and show that heavy generative approaches are not necessary for strong performance. We propose `AFANet`, a lightweight graph-based framework that models interaction trajectories through simple step-level signals and agent-level structure, achieving competitive or superior results against LLM-based methods with minimal computational cost. These findings suggest that structured modeling of agent interactions is sufficient for effective attribution. Future work includes designing more tailored aggregation operators and conducting in-depth analysis of generalization under distribution shifts.

Acknowledgments and Disclosure of Funding

This work is supported by NSF (2433308). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] AI Anthropic. System card: Claude opus 4 & claude sonnet 4. *Claude-4 Model Card*, 2025.

[2] Adi Banerjee, Anirudh Nair, and Tarik Borogovac. Where did it all go wrong? a hierarchical look into multi-agent error attribution. *arXiv preprint arXiv:2510.04886*, 2025.

[3] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *Advances in Neural Information Processing Systems*, 38, 2026.

[4] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[5] Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

[6] Yu Ge, Linna Xie, Zhong Li, Yu Pei, and Tian Zhang. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. *arXiv preprint arXiv:2509.13782*, 2025.

[7] 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. *arXiv preprint arXiv:2402.01680*, 2024.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[9] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[10] 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. *arXiv preprint arXiv:2509.14295*, 2025.

[11] Ting-Wei Li, Sirui Chen, Jiaru Zou, Yingbing Huang, Tianxin Wei, Jingrui He, and Hanghang Tong. Evoselect: Data-efficient llm evolution for targeted task adaptation. *arXiv preprint arXiv:2604.26170*, 2026.

[12] Ting-Wei Li, Ruizhong Qiu, and Hanghang Tong. Graph data selection for domain adaptation: A model-free approach. *Advances in Neural Information Processing Systems*, 38:85341–85377, 2026.

[13] Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

[14] Xuyan Ma, Xiaofei Xie, Yawen Wang, Junjie Wang, Boyu Wu, Mingyang Li, and Qing Wang. Diagnosing failure root causes in platform-orchestrated agentic systems: Dataset, taxonomy, and benchmark. *arXiv preprint arXiv:2509.23735*, 2025.

[15] OpenAI. Gpt-4o-mini. `https://openai.com/index/gpt-4o-system-card/`, 2024.

[16] OpenAI. Gpt-4.1. `https://openai.com/index/gpt-4-1/`, 2025.

[17] OpenAI. o3. `https://openai.com/index/introducing-o3-and-o4-mini/`, 2025.

[18] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `https://arxiv.org/abs/1908.10084`.

[19] Amit Roy, Juan Shu, Jia Li, Carl Yang, Olivier Elshocht, Jeroen Smeets, and Pan Li. Gad-nr: Graph anomaly detection via neighborhood reconstruction. In *Proceedings of the 17th ACM international conference on web search and data mining*, pages 576–585, 2024.

[20] Kai Sun, Wenqiang Li, Bo Dong, Yuxin Lin, Jingyao Zhang, and Bin Shi. Scope delineation before localization: A two-stage framework for enhancing failure attribution in multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33108–33116, 2026.

[21] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. In *International conference on machine learning*, pages 21076–21089. PMLR, 2022.

[22] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[24] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[25] Junjie Wang, Yawen Wang, Mengzhuo Chen, Xiaofei Xie, Chunyang Chen, Fangwen Mu, Zhe Liu, and Qing Wang. A survey for llm agent trajectory analysis: From failure attribution to enhancement. 2026.

[26] Yawen Wang, Wenjie Wu, Junjie Wang, and Qing Wang. From flat logs to causal graphs: Hierarchical failure attribution for llm-based multi-agent systems. *arXiv preprint arXiv:2602.23701*, 2026.

[27] Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, et al. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538*, 2026.

[28] Jun Wu, Jingrui He, and Elizabeth Ainsworth. Non-iid transfer learning on graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 10342–10350, 2023.

[29] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of the web conference 2020*, pages 1457–1467, 2020.

[30] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[31] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[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 preprint arXiv:2509.24088*, 2025.

[33] Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems? *arXiv preprint arXiv:2509.03312*, 2025.

[34] Heng Zhang, Yuling Shi, Xiaodong Gu, Haochen You, Zijian Zhang, Lubin Gan, Yilei Yuan, and Jin Huang. Graphtracer: Graph-guided failure tracing in llm agents for robust multi-turn deep search. *arXiv preprint arXiv:2510.10581*, 2025.

[35] 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. URL `https://openreview.net/forum?id=GazlTYxZss`.

[36] Xiaowen Zhang, Yuntao Du, Rongbiao Xie, and Chongjun Wang. Adversarial separation network for cross-network node classification. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 2618–2626, 2021.

[37] 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 *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7659–7688, 2026.

A Graph construction

TF-IDF computation. For each conversation, we first construct a local vocabulary from all turns after lowercasing and tokenization. Based on this vocabulary, we compute turn-level TF-IDF representations using standard term frequency and smoothed inverse document frequency, followed by row-wise ℓ_2 normalization. Let $\mathbf{X}_{\text{TF-IDF}} \in \mathbb{R}^{T\times|\mathcal{V}|}$ denote the resulting TF-IDF matrix, where T is the number of turns and $|\mathcal{V}|$ is the vocabulary size.

To obtain compact dense representations that capture the dominant vocabulary structure within the conversation, we apply truncated SVD:

$$\mathbf{Z}_{\text{svd}} = \text{SVD}_r(\mathbf{X}_{\text{TF-IDF}}) \in \mathbb{R}^{T\times r},$$

where r denotes the retained rank size. Each row

$$\mathbf{z}^t_{\text{svd}} = \mathbf{Z}_{\text{svd}}[t,:] \in \mathbb{R}^r$$

provides an r-dimensional representation for turn t. In our implementation, we use $r = 16$.

Deviation features. Let $\phi_t = \mathbf{z}^t_{\text{svd}}$ denote the TF-IDF SVD feature of turn t, and let $\mathcal{I}_a = \{t \mid a_{i_t} = a\}$ denote the set of turns generated by agent a.

- **Sequential deviation.** We measure how abruptly a turn deviates from the immediately preceding turn:
$$f_{\text{seq}}(t) = 1 - \cos(\phi_t, \phi_{t-1}).$$
- **Self inconsistency.** We measure deviation from the historical behavior of the same agent:
$$\mu_{a_{i_t}} = \frac{1}{|\mathcal{I}_{a_{i_t}}|} \sum_{s\in\mathcal{I}_{a_{i_t}}} \phi_s,$$
$$f_{\text{self}}(t) = 1 - \cos(\phi_t, \mu_{a_{i_t}}).$$
- **Conversation consensus deviation.** We compute deviation from the global conversation centroid:
$$\mu_{\mathcal{T}} = \frac{1}{T}\sum_{s=1}^{T} \phi_s,$$
$$f_{\text{cons}}(t) = 1 - \cos(\phi_t, \mu_{\mathcal{T}}).$$
- **Cross-agent deviation.** We measure deviation from the average representation of all other agents:
$$\mu_{\neg a_{i_t}} = \frac{1}{T - |\mathcal{I}_{a_{i_t}}|} \sum_{s\notin\mathcal{I}_{a_{i_t}}} \phi_s,$$
$$f_{\text{other}}(t) = 1 - \cos(\phi_t, \mu_{\neg a_{i_t}}).$$
- **Agent internal consistency.** We compute the average pairwise similarity among all turns generated by the same agent:
$$f_{\text{within}}(a) = \frac{1}{|\mathcal{I}_a|(|\mathcal{I}_a|-1)} \sum_{\substack{s,u\in\mathcal{I}_a \\ s\neq u}} \cos(\phi_s, \phi_u).$$
- **Same-agent temporal consistency.** We measure similarity to the previous turn generated by the same agent:
$$p(t) = \max\{s < t \mid a_{i_s} = a_{i_t}\},$$
$$f_{\text{prev}}(t) = \cos(\phi_t, \phi_{p(t)}).$$
- **Vocabulary stability.** Let $\mathcal{V}_t$ denote the token set of turn t, and let
$$\mathcal{V}^{(-t)}_{a_{i_t}} = \bigcup_{s\in\mathcal{I}_{a_{i_t}}, s\neq t} \mathcal{V}_s.$$
We measure the overlap ratio between the current turn and the remaining turns from the same agent:
$$f_{\text{vocab}}(t) = \frac{|\mathcal{V}_t \cap \mathcal{V}^{(-t)}_{a_{i_t}}|}{|\mathcal{V}_t|}.$$

- **Problem alignment.** We measure similarity between the current turn and the initial problem statement:

$$f_{\text{prob}}(t) = \cos(\phi_t, \phi_1).$$

Statistical features.

- **Numerical statistics.** We include lightweight numerical indicators, including whether the turn contains numeric tokens:

$$f_{\text{num}}(t) = \mathbb{I}[\exists \text{ digit token in } x_t],$$

as well as the normalized mean and standard deviation of word lengths within the conversation:

$$f_{\text{wmean}}(t) = \frac{\text{mean_len}(x_t) - \mu_{\text{len}}}{\sigma_{\text{len}}}, \qquad f_{\text{wstd}}(t) = \frac{\text{std_len}(x_t) - \mu_{\text{std}}}{\sigma_{\text{std}}}.$$

- **Position statistics.** We include normalized turn position

$$f_{\text{pos}}(t) = \frac{t-1}{T-1},$$

binary indicators for whether the turn is the first or last turn:

$$f_{\text{first}}(t) = \mathbb{I}[t = 1], \qquad f_{\text{last}}(t) = \mathbb{I}[t = T],$$

and the cumulative fraction of turns authored by the same agent up to step t:

$$f_{\text{agent_frac}}(t) = \frac{|\{s \le t \mid a_{i_s} = a_{i_t}\}|}{t}.$$

- **Length statistics.** Let ℓ_t denote the token length of turn t. We include normalized turn length:

$$f_{\text{len}}(t) = \frac{\ell_t}{\max_s \ell_s},$$

together with the conversation-level z-normalized length:

$$f_{\text{zlen}}(t) = \frac{\ell_t - \mu_\ell}{\sigma_\ell}.$$

- **Structural statistics.** We additionally include the total participation ratio of the corresponding agent:

$$f_{\text{part}}(t) = \frac{|\mathcal{I}_{a_{i_t}}|}{T},$$

and the spread of the agent's participation positions:

$$f_{\text{spread}}(a) = \text{std}\left(\left\{\frac{s-1}{T-1} \mid s \in \mathcal{I}_a\right\}\right).$$

B Dataset Details

Both datasets ae under MIT licenses.

AEGIS-Bench [10]			**Who&When** [35]
Train	**Validation**	**Test**	**Test**
7146	1787	600	184

Table 6: Dataset size statistics used in our experiments.

Table 7: Sensitivity to sentence encoder choice. The default encoder is in bold.

Dataset	Encoder	aμF1	aMF1	eμF1	eMF1	pμF1	pMF1
AEGIS-Bench	Small	73.09	47.18	23.03	22.29	16.83	15.33
	Default	**74.16**	**47.86**	27.01	25.96	17.42	**16.35**
	Large	72.15	46.41	**27.98**	**27.47**	**17.48**	14.22
Who&When	Small	33.09	**24.33**	11.44	**8.79**	2.87	2.10
	Default	**37.93**	20.26	**13.79**	6.05	**6.90**	**4.16**
	Large	35.00	23.63	10.01	7.25	3.00	1.72

C Error Types

Adopted from [10], we consider the following 14 error types in agent failure attribution:

- **Specification Issues**
  - FM-1.1: Task specification deviation
  - FM-1.2: Role specification deviation
  - FM-1.3: Add redundant steps
  - FM-1.4: Remove conversation history
  - FM-1.5: Remove termination conditions
- **Inter-Agent Misalignment**
  - FM-2.1: Repeat handled tasks
  - FM-2.2: Make request ambiguous
  - FM-2.3: Deviate from main goal
  - FM-2.4: Hide important information
  - FM-2.5: Ignore other agents
  - FM-2.6: Inconsistent reasoning
- **Task Verification Failures**
  - FM-3.1: Premature termination
  - FM-3.2: Remove verification steps
  - FM-3.3: Incorrect verification

D Encoders Sensitivity

We investigate the sensitivity of `AFANet` to the choice of sentence encoder. We use `paraphrase-MiniLM-L3-v2` (17M parameters), `all-MiniLM-L6-v2` (22M parameters, default), and `all-mpnet-base-v2` (109M parameters). We report results on both the in-domain AEGIS-Bench and out-of-domain Who&When benchmarks in Table 7. Overall, AFANet maintains competitive performance across all three encoders.

E Training-Free and Lightweight Baselines

We also compare `AFANet` against lightweight alternatives: a training-free z-score method, Logistic Regression (LR), and Random Forest (RF). For each agent, we apply mean and max pooling to its deviation and statistical features and concatenate the pooled representations, yielding a 38-dimensional feature vector.

Baseline configurations. The training-free z-score baseline computes the within-conversation z-score L2 norm over deviation and statistical features. For LR and RF, we train binary classifiers using agent-level and error-level labels. LR minimizes cross-entropy, while RF fits decision trees on bootstrap samples. For LR, we search over $C \in \{0.01, 0.1, 1, 10\}$ and $\{\ell_1$/SAGA, ℓ_2/LBFGS$\}$, resulting in eight configurations. For RF, we search over $n_{\text{estimators}} \in \{100, 200\}$ and $\text{max_depth} \in \{5, 8, 12, \text{None}\}$, also resulting in eight configurations. We select the best configuration using

Table 8: Comparison with training-free and lightweight baselines on AEGIS-Bench and Who&When. All results are percentages. Avg. is the arithmetic mean of the six F1 metrics.

Dataset	Method	aμF1	aMF1	eμF1	eMF1	pμF1	pMF1	Avg.
AEGIS-Bench	z-score	21.59	6.75	7.96	3.76	2.90	0.74	7.28
	LR	44.06	24.00	9.44	8.78	5.94	3.40	15.94
	RF	52.32	34.89	11.50	10.73	8.26	4.42	20.35
	`AFANet`	**74.16**	**47.86**	**27.01**	**25.96**	**17.42**	**16.35**	**34.79**
Who&When	z-score	28.74	20.82	6.32	4.04	0.57	0.46	10.16
	LR	24.14	15.53	9.20	4.29	1.72	0.47	9.22
	RF	35.06	**21.22**	7.47	3.59	1.72	0.42	11.58
	`AFANet`	**37.93**	20.26	**13.79**	**6.05**	**6.90**	**4.16**	**14.85**

validation pair-level micro-F1. The selected configurations are $C = 10$ with ℓ_1 regularization for LR and 200 trees with unrestricted depth for RF.

Results. As shown in Table 8, `AFANet` substantially outperforms the lightweight baselines on both datasets.

F Test-Time Adaptation for `AFANet`

Below we detail the loss objective for each TTA strategy.

Agent-level entropy minimization. Let a_i denote the scalar faulty-agent logit for agent i, and let $p_i = \sigma(a_i)$. We define

$$\mathcal{L}_{\text{agent}} = \frac{1}{N_A} \sum_{i=1}^{N_A} H_{\text{bin}}(p_i) = -\frac{1}{N_A} \sum_{i=1}^{N_A} \left[p_i \log p_i + (1 - p_i) \log(1 - p_i) \right].$$

Pair-level entropy minimization. Let $\mathbf{s}_i \in \mathbb{R}^{K+1}$ denote the pair logits for agent i, where the first class corresponds to the clean class and the remaining K classes correspond to error types. With $\hat{p}_{ic} = \text{softmax}(\mathbf{s}_i)_c$, we minimize

$$\mathcal{L}_{\text{pair}} = \frac{1}{N_A} \sum_{i=1}^{N_A} H_{\text{cat}}(\mathbf{s}_i) = -\frac{1}{N_A} \sum_{i=1}^{N_A} \sum_{c=0}^{K} \hat{p}_{ic} \log \hat{p}_{ic}.$$

Faulty-agent pair-level minimization. We first estimate the number of faulty agents as $k = \max\left(1, \sum_{i=1}^{N_A} \mathbb{I}[a_i > 0]\right)$ and select the top-k agents according to the faulty-agent logits a_i. The adaptation objective is then

$$\mathcal{L}_{\text{pair-faulty}} = \frac{1}{k} \sum_{i \in \text{Top-}k(a)} H_{\text{cat}}(\mathbf{s}_i) = -\frac{1}{k} \sum_{i \in \text{Top-}k(a)} \sum_{c=0}^{K} \hat{p}_{ic} \log \hat{p}_{ic}.$$

For the TTA procedure, we freeze all parameters except the LayerNorm affine parameters across all layers in `AFANet` and minimize the above three losses. We use Adam optimizer with learning rate $1e-3$ for 30 gradient steps over the entire OOD batch. No OOD labels are used. The final checkpoint is taken as the final model for evaluation.

G Extended Related Work

Domain Adaptation. Attributing agent failures across diverse multi-agent systems requires robustness against distribution shifts. Prior studies explore domain adaptation by aligning representations across domains [29, 36, 28] or adapting models toward specific target tasks [12, 11].

H Limitation

- **OOD generalization.** Although `AFANet` demonstrates certain robustness under distribution shift, generalization across substantially different multi-agent settings remains challenging. Future work may include designing customized message passing operators and adaptive graph propagation mechanisms that better capture failure propagation patterns.
- **Limited reasoning capability.** While lightweight structural modeling is effective for our studied attribution scenarios, certain failures may still require deeper semantic understanding and long-horizon reasoning. An important future direction is to combine graph-based modeling with LLM reasoning to jointly leverage structural consistency and semantic inference.

I Impact Statement

This paper discusses the advancement of the field of Large Language Model and Graph Machine Learning. While there are potential societal consequence of our work, none of which we feel must be hightlighted.

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: Our main contributions are summarized as four bullet points in Section 1 and each of them are supported by claims in introduction or experiment section.
   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: We discuss potential limitations of our presented work in Appendix H.
   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: [N/A]
   Justification: We don't include theoretical results.
   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: We provide all experimental details needed in Section 4.1.
   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: All the benchmarks used in this work is public. We will provide the code package during submission and make the code available upon acceptance.
   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: We provide all experimental details needed in Section 4.1.
   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: [No]
   Justification: Our conclusions are well supported by consistent trends across multiple datasets, evaluation metrics, model backbones and ablation settings.
   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: We provide all experimental details needed in Section 4.1.
   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 confirm that this work is conducted with the NeurIPS Code of Ethics.
   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: We provide related discussion in Appendix I.
    Guidelines:
    - The answer [Yes] 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 confirm that this work does not pose safety risks.
    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 detail them in Appendix B.
    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: No new assets introduced.
    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: This paper does not involve 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: This paper does not involve 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: [N/A]
Justification: The proposed method is entirely LLM-free and does not use LLMs as part of its core methodology. Although some comparison baselines are LLM-based methods, they are only included for evaluation purposes and are not key components of the proposed framework.
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.