Author Response of CollaborLog (Submissions #850)

We would like to sincerely thank the reviewers and the meta-reviewer for their constructive and insightful feedback. We have carefully revised the manuscript to address all the comments.

In the revised version, newly added or substantially rewritten content is highlighted in blue using add content, while deleted content is shown using ~~deleted content~~. For figures and tables that were modified or newly introduced, we highlighted them with a blue background for clarity, for example: The caption or revised table cell .

In the following, we provide point-by-point responses to all comments. Reviewer (and meta-review) comments are shown in **bold**, followed by our detailed replies. Specifically, each reviewer is assigned a dedicated section in this response letter, where we address all their comments, including those raised during the rebuttal stage as well as any additional remarks in the full reviews. For each point, we not only provide direct answers but also indicate where in the manuscript the corresponding revision has been made.

Finally, we provide a dedicated section for the meta-review, where we summarize how we have addressed the broader concerns. For clarity, we also include cross-references to the relevant RQs and revised sections of the paper.

I. Review #850A

A. *Questions for authors' response*

Q1: How is a variable-length log sequence represented as a fixed-dimension input for the autoencoder?

Re: We use BERT to encode each log entry into an embedding, then average all embeddings in a log sequence to get its feature vector. This allows the model to handle variable-length logs (e.g., Spark logs). In the revised manuscript (Section III.A), we have clarified this point: "For a given log sequence s, we first use BERT to encode each log entry $l_i \in s$ into an embedding, then average all embeddings in a log sequence to get its feature vector. This allows the model to handle variable-length logs. "

Q2: Why is $y_{ij} = y_i \cdot y_j$ in contrastive learning, given that (0,0) pairs are same-class and should be considered similar?

Re: In the revision, we corrected the definition: $y_{ij} = 1$ if $y_i = y_j$ (same class) and $y_{ij} = 0$ otherwise. This ensures that $(0, 0)$ pairs are treated as similar, consistent with standard contrastive learning. The correction is now reflected in Equation (4).

Q3: How is $sim(s, s_{pre})$ defined and computed?

Re: We have clarified this point in the revision. Specifically, $sim(s, s_{pre})$ is defined as the cosine similarity between the sequence-level feature vectors extracted by the BERT-RNN, as given in Equation (5). This definition has been added after Equation (7) in Section III.C for completeness.

Q4: Why are efficiency metrics not compared to traditional baselines (LogAnomaly, LogRobust, etc.)?

Re: Our main focus is on reducing LLM calls and token consumption, which are not issues for traditional baselines. Therefore, we primarily compared CollaborLog against LLM-only settings (i.e., vanilla). Nevertheless, in the revision we added efficiency comparisons with traditional baselines (LogAnomaly, LogRobust, etc.) in Table V of Section IV.E to highlight trade-offs. For example, on the Spark 2→ Spark 3 dataset, LogRobust and LogAnomaly process each sample in 0.96ms and 1.69ms, respectively, while on Hadoop 2 → Hadoop 3, they take 0.43ms and 0.83ms. In contrast, LLMs suffer from higher latency due to their large model size and network overhead. However, our coordinator design makes LLM-based methods more practical for industrial-speed requirements.

Q5: Can CollaborLog provide explainability or rationale for why a log sequence is abnormal?

Re: Yes. By leveraging the emergent reasoning capabilities of LLMs, CollaborLog integrates a chain-of-thought (CoT) strategy into every LLM invocation, ensuring that each decision is accompanied by an explicit reasoning trace. To make this more concrete, we added a new case study in Section IV.F (Fig. 7) of the revised manuscript.

In the case study, we selected two key scenarios within Evol-CoT: (a) Fig.7 (a) demonstrates how CollaborLog explains the identification of an evolution relation, and (b) Fig.7 (b) illustrates how CollaborLog performs anomaly detection when no evolution relation is matched. Together, these cases demonstrate that CollaborLog can provide clear and interpretable rationales

across both critical settings.

Q6: Can you clarify the difference between "OurLLM" and "Our (w/o Coord)"?

Re: We thank the reviewer for this comment. In the revised manuscript, Section IV.B has been thoroughly rewritten. We now present the ablation settings in a concise bullet-style description, avoiding duplication.

The repeated description of "Our_{LLM}" has been removed; instead, we use the term **Evol-CoT** to denote the LLM-based component of our method (③ in Figure 5). The variant **(w/o Coord)** refers to the setting where the coordinator is removed, and all logs are routed directly to the LLM. We clarified the distinction between "Evol-CoT" and "Our (w/o Coord)."

Q7: How is k chosen? What is its impact on performance?

Re: We select k based on the small model's F1-score on the validation set. In the revised manuscript (**Section IV.D and Fig. 6(b)**), we provide a detailed sensitivity analysis of how different k values affect performance. Similar trends are observed across both datasets: when $k = 1$, performance is poor due to insufficient reference samples; as k increases, performance improves, but overly large values introduce dissimilar samples, which reduces the effectiveness of majority voting. The optimal k is therefore determined by the highest F1-score on the validation set for both anomaly detection and RAG retrieval. We set the hyperparameter k to 4 on the Spark dataset and to 5 on the Hadoop dataset(**Section IV.A**).

B. For other questions in Detail comment

Q8: Missing/ambiguous technical details.

Re: The issues regarding variable-length log sequences, the definition of y_{ij} in Equation (4), and the computation of $sim(s, s_{pre})$ in Equation (7) have been clarified in detail (see Q1–Q3 above).

Q9: Datasets like LogHub:

Re: We first validated our model on LogHub before conducting full experiments on the LOGEVOL dataset for software evolution (as mentioned in section IV-B). Due to space limits, the LogHub results were not included in the main text but are available in the anonymous link [1] (now fixed). On LogHub, our method achieved higher F1-scores than using either the small model or LLM alone. The detailed results are shown in Table I-B.

Q10 Other issues of Experiment Settings.

What criteria guided the selection of baselines? We chose representative baselines for statistical, unsupervised, supervised, semisupervised, and LLM-based methods. Criteria and rationale have been explicitly added in **Section IV.B** of the revision.

TABLE I
RESULTS ON BGL AND ZOOKEEPER OF LOGHUB.

Method	BGL			Zookeeper		
	F1	Pr	Re	F1	Pr	Re
BERT	73.93%	97.74%	59.45%	53.33%	100%	36.96%
Trained SM	83.40%	98.14%	72.51%	72.17%	98.22%	57.04%
LLM	76.55%	62.10%	100%	84.69%	73.60%	100%
Our	97.10%	99.83%	94.67%	99.31%	99.99%	98.63%

Choice of k: The choice of k has been explained in detail in Q7 above. Specifically, we set $k = 4$ for the Spark dataset and $k = 5$ for the Hadoop dataset.

Ablation settings redundancy: The redundancy in the ablation settings has been eliminated (see Q6 above).

Q11: More efficiency comparisons. Re: In the revised manuscript, we have clarified that the reported small model cost already includes the overhead of BERT-based embedding generation, as the small model is built upon BERT. This implicitly reflects the relative inference cost of BERT within the overall efficiency comparison. In addition, we extended the comparisons to include traditional baselines such as LogAnomaly and LogRobust. These results are now reported in Section IV.D and Table V, helping readers better understand the trade-offs between LLM-based and traditional methods.

Q12: For explainability and artifact.

For explainability, we have added a case study (Section IV.F, Fig. 7) demonstrating how CollaborLog provides interpretable rationales for both evolution-relation recognition and anomaly detection (see Q5 above). Evol-CoT improves interpretability by breaking the task into three sub-tasks with Chain-of-Thought prompting, guiding the LLM to reason step by step. Prompts and examples are in [1].

Regarding the artifact, we have updated the anonymous link to include detailed implementation instructions and additional results (Section IV.A, link updated).

C. Writing and Presentation

We have carefully revised the manuscript to address all the noted writing, formatting, and presentation issues:

- Replaced "evolutionary logs" with "evolved logs."
- Reduced redundant acronym usage of LLM.
- Corrected the usage of "i.e." to "i.e.," throughout.
- We have ensured that the precision of F1-score, precision, and recall is uniformly reported to three decimal places (Section IV).
- Fixed mismatched table values (Table I, II).
- Rephrased the description of "Our_{SM} (w/o AEM)" to avoid misleading wording.

- Corrected page-level typos, broken references, spacing, and inconsistent notation as suggested.
- Fixed typographical errors (e.g., "los routed to" → "logs are routed to," "Fasle" → "False").
- Removed redundant words (e.g., "logs logs").
- Fixed broken references (e.g., Evol-CoT on page 2, Table III on page 9).
- Ensured consistent mathematical notation (e.g., indexing conventions, corrected Equation (1), and proper set notation in S_c).
- Improved formatting issues (e.g., spacing before equations, punctuation around "CollaborLog," and appropriate symbol usage).

II. Review #850B

A. *Questions for authors' response*

Q1: How many evolutionary logs are accumulated before an update is triggered, and how does this affect performance?

Re: We have addressed this question in the revision. As shown in the newly added Table IV, AEM leverages the evolutionary relations identified by Evol Detect, where 288 and 541 relations were collected on the Spark and Hadoop datasets, respectively. These relations are then used to update both the coordinator and the small model (SM): (1) the coordinator is updated so that logs with identified evolution relations are directly routed to the SM, and (2) the SM is further optimized using contrastive loss on these relations.

The effectiveness of this update process can be observed in the experimental results. First, Table III shows that after applying AEM, the proportion of logs routed to the LLM decreases, reducing resource consumption. Second, by refining the SM with contrastive loss on the evolution pairs, its capability on evolved logs improves significantly. As a result, the overall detection performance also increases, as reported in Table II, where all F1-scores with "+AEM" are higher than without AEM.

In summary, AEM enables the system to continuously adapt to new data, improving efficiency while maintaining or enhancing anomaly detection accuracy. These details are explicitly discussed in Section IV.D (4) "Effect of AEM" with updated results included in Tables II, III, and IV.

Q2: What is the time cost and frequency of performing an update?

Re: The update process of AEM is highly efficient. Since only a small number of high-confidence evolved logs are reused and the process is accelerated by GPUs, the update is highly efficient. As shown in **added Table IV of section IV.D**, the average per-sample update time is less than 0.2ms on both datasets, demonstrating that updates introduce negligible computational overhead. Regarding update frequency, in our experimental setup we performed one update per software version, i.e., using the evolution relations collected from the validation set after each version release. This ensures fairness, as no labels from the validation set are leaked into training.

In practical deployment, CollaborLog is applied in a large-scale microservice change system. We found that triggering an update when the number of new logs reaches a certain threshold (around 10% of total logs) yields the best balance: updates are frequent enough to capture distribution shifts while remaining stable and cost-effective.

Q3: Do experiments wait for updates to finish before continuing detection? How is the update adopted in practice?

Re: In experiments, updates are performed before detection due to the low update cost. In real-world deployment, detection must remain online. We therefore fine-tune a replica model, validate it, and then replace the online model. This strategy avoids service interruption and is much faster than full retraining.

Q4: How do you ensure the reliability of reusing detection results?

Re: Reliability is ensured in several ways. First, the evolution relationship detection task in Evol-CoT is formulated as a text association judgment, which is relatively simple for LLMs, and we strictly constrain the decision rules to guarantee consistency. Only high-confidence evolution pairs are reused. Second, the coordinator and small model are updated synchronously to ensure consistent handling of reused results. Third, even if an LLM judgment were imperfect, resubmitting the same case to LLM would produce the same output, meaning that AEM avoids redundant LLM usage without degrading detection capability. This clarification has been added in Section III.D.

Concerns about the rationality of utilizing evolutionary relationships: When guiding the LLM to identify evolutionary logs, the key criterion is that the underlying semantic pattern of the log sequence must remain unchanged. Even if additional events or parameters appear, once they alter the semantic pattern, the logs will not be judged as evolutionary. We have strictly constrained the identification rules to ensure that only highly reliable evolutionary relationships are recognized.

In the revised version, we also added experimental results to demonstrate the reliability of reusing detection results and the effectiveness of the update process. Specifically, Table IV reports the accuracy of reused detection results, computed by comparing the labels of pre- and post-evolution logs. The accuracy reaches 0.993 and 0.986 on the two datasets, confirming the high reliability of LLM-identified evolution relations. In addition, Tables II and III present the effect of AEM

updates on routing decisions and detection performance. The results show that AEM reduces the proportion of logs routed to the LLM while improving detection on evolved logs, leading to an overall increase in F1-score (Table II). Update costs are minimal (Table IV), ensuring practical applicability.

B. Other questions under Presentation

Q5: Which part of Evol-CoT corresponds to the CoT design?

Re: The core idea of Chain-of-Thought (CoT) is to solve tasks through explicit intermediate reasoning steps. In *Evol-CoT*, this is reflected in two ways: (1) we explicitly decompose anomaly detection into multiple reasoning steps, including *Evol Detect* (3.1) and *AD Agent* (3.2), with the AD Agent further containing two complementary reasoning paths; (2) all prompts adopt CoT-style step-by-step instructions, which guide the LLM to reason progressively and better leverage its reasoning ability. These designs are clarified in **Section III.C** of the revised manuscript, and the detailed prompt formulations are provided in [1].

Q6: Why is Equation (9) treated as a tool? What is the role of p_i in relation to uncertainty?

Re: Equation (9) is used in the AD Agent for log classification, estimating the uncertainty of a log s. For each similar historical log s_i, p_i denotes the proportion of logs sharing its anomaly label. If p_i is close to 0 or 1, the log is more certain and contributes less to overall uncertainty; if p_i is near 0.5, it contributes more. The final uncertainty of s is computed by aggregating $U(s_i)$, weighted by similarity to s. Since computing similarity directly with the LLM is complex and often inaccurate, we encapsulate Equation (9) as a tool for efficient uncertainty estimation and provide its results to the LLM as auxiliary input. We added this explanation in Section III.C of the revision.

Q7: Is it meaningful to consider evolution from a new version to an old version (e.g., Hadoop 3 → Hadoop 2)?

Re: Our setting follows prior work on software evolution [2], where both forward and backward version transitions provide valid scenarios to evaluate the robustness of log-based anomaly detection. We clarified this point in Section IV.A.

Q8: Several minor issues were noted, including missing outputs in Fig. 5, a typo in "Evol Relations," broken references, and an empty link.

Re: We carefully revised the manuscript to correct these issues: added the missing output in Fig. 5, fixed the typo, repaired broken references, and fixed the link [1]. These changes improve readability and completeness.

III. Review #850C

A. Questions for authors' response

Q1: What is the reasoning behind introducing an extra stage of Evolution Detection instead of feeding suspicious evolutionary logs directly into the AD Agent?

Re: The evolution detection stage enables in-model task decomposition, allowing the LLM to focus solely on identifying log evolution through CoT reasoning. This improves the reliability of the LLM and follows common practice in recent work (e.g., [3]). In the revised manuscript (**Section III.C**), we clarified this motivation and positioned evolution detection as a critical intermediate step before anomaly detection.

Q2: How does the model perform over τ less than 40?

Re: In the revised manuscript (**Section IV.D, Fig. 6**), we updated the figure to report the small model's F1-score on evolved logs across the full τ range (0–100 percentiles). We also added a detailed discussion in the subsection "Effect of Coordinator" of Section IV.D to explain the influence of this hyperparameter.

When $\tau < 40$, evolved and non-evolved logs are difficult to distinguish, resulting in only slight fluctuations in F1-scores. As τ increases, samples with greater distributional drift are gradually selected as evolved logs, and the small model's performance on these samples decreases accordingly. Moreover, τ controls how many samples are routed to the LLM: very small τ values increase cost, while very large values degrade routing and overall performance. Based on this analysis, we select τ at the inflection point(i.e., Q85 for Spark and Q90 for Hadoop), where the small model's F1-score on evolved logs begins to decline.

Q3: What proportion of logs are routed into Evol-CoT, and among those, how many proceed to the AD Agent?

Re: For the Hadoop dataset, 8.50% of logs are routed to Evol-CoT, with 5.16% subsequently proceeding to the AD Agent. For the Spark dataset, 4.82% are routed to Evol-CoT, with 4.05% further processed by the AD Agent. In the revised manuscript (**Section IV.D, Table III**), we explicitly report these statistics. We also added a new subsection **"3) Effect of Evol-CoT" in Section IV.D**, where we provide a detailed discussion of the routing behavior within Evol-CoT, including proportions, counts, and performance analysis.

B. Other Question in Soundness

Q4: What happens when $\tau < 40$? Is there an optimal value?

Re: More Detail Please see RQ2 above, where we revised Figure 6 (Section IV.D) to include the full τ range and added a detailed discussion of how τ affects

both routing cost and performance. We clarified that the optimal τ is chosen at the inflection point where the small model's F1-score on evolved logs begins to sharply decline.

Q5: In Equation (3), should "MSE$(s) \geq \tau$" be labeled as True?.

Re: We thank the reviewer for pointing this out. This was a typographical error. In the revision, we corrected Equation (3) so that "MSE$(s) \geq \tau$" is labeled as *True*.

Q6: Why use a Siamese network, and why contrastive loss?

Re: We adopt a Siamese network because anomaly detection in logs can be naturally formulated as a similarity problem: normal logs and their evolved variants should lie close in the embedding space, while abnormal logs should be distant. The Siamese structure, with parameter sharing, provides a stable mechanism for modeling semantic similarity and is robust to distribution shifts. Moreover, its pair-based design enables continual updates, as newly identified evolution pairs can be directly incorporated into training without retraining from scratch.

We selected contrastive loss because it is the standard training paradigm for Siamese networks, explicitly enforcing intra-class compactness and inter-class separation. This aligns well with the binary nature of anomaly detection (normal vs. abnormal).

In the revised manuscript (Section III.B), we have added a detailed explanation of the Siamese architecture, its suitability for log anomaly detection, and the motivation for adopting contrastive loss. These revisions clarify the design insights behind our choice.

Q7: Why does Equation (9) combine cross-entropy with inter-sample similarity? What do you mean by "fuzzy" samples?

Re: In the revision, we expanded **Section III.C** to explain the design. The uncertainty of a sample s is estimated by aggregating the uncertainties of similar historical logs S_c, weighted by semantic similarity to s. For each $s_i \in S_c$, we compute the class ratio p_i (the proportion of logs sharing the same anomaly label) and derive its uncertainty $U(s_i)$. Then, we weight $U(s_i)$ by its similarity to s. If most similar logs belong to the same class, s is more certain (low uncertainty). If the set is evenly split, s is more uncertain (high uncertainty). Intuitively, if most similar logs in S_c belong to the same class, s is more certain, so uncertainty $U(s)$ is low. If S_c is evenly split between normal and abnormal logs, s is more uncertain, and $U(s)$ is high.

As noted in the manuscript, "fuzzy" samples correspond to boundary cases where similar logs include both positives and negatives. We model this with cross-entropy, a common choice for quantifying uncertainty [4].

Q8: Why introduce Evolution Detection instead of sending logs directly to the AD Agent?

Re: Please see RQ1 above, where we explained that the evolution detection stage enables in-model task decomposition, allowing the LLM to focus exclusively on identifying log evolution through CoT reasoning, thereby improving reliability.

Q9: What proportion of logs are routed into Evol-CoT, and how effective are its components?

Re: Please see RQ3 above, where we reported detailed routing statistics (Section IV.D, Table III) and added a new subsection "Effect of Evol-CoT" to analyze its contribution. The ablation study has also been expanded to illustrate the impact of individual components within Evol-CoT.

C. Presentation

Q10: Most content in Section 2 is similar to the preliminary study in [2].

Re: We only referred to [2] when introducing examples of log evolution representation. Section 2 was intended to provide readers with the necessary background, ensuring clarity and readability of the paper. The content is not a direct reuse of [2]; instead, we reorganized and summarized the discussion to offer a more detailed explanation and formal definition of log evolution in the context of this work. This section thus serves as background rather than novel contribution, aligning the paper with the focus of our study.

Q11: P3: "Hallucination occurs when LLMs generate plausible but incorrect information": Can you provide an example on this?

Re: We thank the reviewer for the suggestion. After careful consideration, we decided not to include a concrete example in the main text, as hallucination is not the primary problem addressed in this paper and adding such an example would affect the conciseness and focus of the presentation. Instead, we clarified the concept in the revised manuscript and added citations [5], [6] to well-known studies on hallucination, so that readers unfamiliar with the term can refer to established examples in the literature.

Q12: Other presentation issues were noted, including missing or broken references, unclear wording, and minor formatting problems.

Re: We carefully proofread and revised the manuscript to address all presentation issues raised by the reviewer. Specifically, we added missing references (e.g., for Evol-CoT), and provided an explicit example. We also corrected typographical and formatting issues (e.g., "los routed" $\rightarrow$ "logs are routed," extra spaces in Fig. 1, missing table numbers), and fixed broken or unclear references (e.g., Evol-CoT citation, LogHub usage on p.7).

Together, these revisions improve the clarity, readability, and completeness of the presentation.

IV. Meta-Review Comments and Responses

Q1: Writing and Presentation.

Re: We have carefully proofread the entire manuscript to address all writing and presentation issues. The reviewer's specific comments and our corresponding revisions are detailed in **Section I-C of Review#850A, Section II-B of Review#850B, and Section III-C of Review#850C** above in this response letter. In addition to these explicit changes, we also refined technical clarifications, unified formal definitions and notation, corrected typos and grammatical errors, and ensured consistent formatting across the paper. We believe the revised version now meets the expected standard of clarity and presentation.

Q2: Sensitivity and Ablation Studies: Figure 6 and parameter sensitivity analyses remain unclear. Please analyze the effects of different τ values (including those below 40) and k values on anomaly detection. Additionally, provide a detailed investigation of Evol-CoT consistent with your earlier response to support your design.

Re: We extended our sensitivity analysis by reporting the full range of τ values (0–100) in Fig. 6(a) and by adding a detailed discussion in Section IV.D. We further analyzed the effect of k in Section IV.D with supporting results in Fig. 6(b). To clarify the design and effectiveness of Evol-CoT, we revised the ablation study in Section IV.D and added results in Table III. These revisions provide a more complete investigation of parameter sensitivity and component effectiveness.

For further details, please refer to Reviewer #850A RQ7, Reviewer #850B RQ5, and Reviewer #850C RQ1–RQ4 above.

Q3: Log Evolution Scenario: Clarify how the log evolution scenario integrates with your approach, specifically, how many evolutionary logs are accumulated for triggering, how their quantity impacts performance, and what proportion of logs is routed to various components.

Re: We have clarified how log evolution integrates with our approach. In the revision, we explicitly report: (i) how many evolutionary relations are accumulated before triggering updates (Table IV and Section IV.D), (ii) how their accumulation affects routing and performance (Tables II–III and Section IV.D), and (iii) what proportion of logs is routed to Evol-CoT and subsequently to the AD Agent (Table III and Section IV.D). These clarifications make the role and impact of log evolution explicit in our framework.

For further details, please refer to Reviewer #850B RQ1–RQ2 and Reviewer #850C RQ3 above.

Q4: Reliability of Reuse and Update Effectiveness: Please conduct an experimental analysis to evaluate how to ensure the reliability of reusing detection results and to assess the effectiveness of the update process.

Re: We added both methodological clarifications and experimental results to demonstrate the reliability of reusing detection results and the effectiveness of the update process. Specifically, Table IV reports the accuracy of reused detection results, computed by comparing the labels of pre- and post-evolution logs. The accuracy reaches 0.993 and 0.986 on the two datasets, confirming the high reliability of LLM-identified evolution relations.

In addition, Tables II and III present the effect of AEM updates on routing decisions and detection performance. The results show that AEM reduces the proportion of logs routed to the LLM while improving detection on evolved logs, leading to an overall increase in F1-score (Table II). Update costs are minimal (Table IV), ensuring practical applicability.

These results and discussions have been added in Section IV.D. For more details, please also refer to Reviewer #850B RQ1 and RQ2.

V. Conclusion

We sincerely appreciate the reviewer's feedback. The revision addresses all technical, experimental, and presentation issues, and we believe the paper is significantly improved.

References

[1] Anonymous. (2025) Code repository. [Online]. Available: https://anonymous.4open.science/r/CLSLog-0007/README.md

[2] Y. Huo, C. Lee, Y. Su, S. Shan, J. Liu, and M. R. Lyu, "Evlog: Identifying anomalous logs over software evolution," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 391–402.

[3] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal, "Decomposed prompting: A modular approach for solving complex tasks," *arXiv preprint arXiv:2210.02406*, 2022.

[4] X. Chen, S. Kar, and D. A. Ralescu, "Cross-entropy measure of uncertain variables," *Information Sciences*, vol. 201, pp. 53–60, 2012.

[5] Y. Zhang, P. Xiao, L. Wang, C. Zhang, M. Fang, Y. Du, Y. Puzyrev, R. Yao, S. Qin, Q. Lin *et al.*, "Ruag: Learned-rule-augmented generation for large language models," *arXiv preprint arXiv:2411.03349*, 2024.

[6] Z. Ji, T. Yu, Y. Xu, N. Lee, E. Ishii, and P. Fung, "Towards mitigating llm hallucination via self reflection," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 1827–1843.