Think Wider, Detect Sharper: Reinforced Reference Coverage for Document-Level Self-Contradiction Detection

Anonymous EMNLP submission

Abstract

Detecting self-contradictions within documents is a challenging task for ensuring textual coherence and reliability. While large language models (LLMs) have advanced in many natural language understanding tasks, document-level self-contradiction detection (DSCD) remains insufficiently studied. Recent approaches leveraging Chain-of-Thought (CoT) prompting aim to enhance reasoning and interpretability; however, they only gain marginal improvement and often introduce inconsistencies across repeated responses. We observe that such inconsistency arises from incomplete reasoning chains that fail to include all relevant contradictory sentences consistently. To address this, we propose a two-stage method that combines supervised fine-tuning (SFT) and reinforcement learning (RL) to enhance DSCD performance. In the SFT phase, a teacher model helps the model learn reasoning patterns, while RL further refines its reasoning ability. Our method incorporates a task-specific reward function to expand the model's reasoning scope, boosting both accuracy and consistency. On the ContraDoc benchmark, our approach significantly boosts Llama 3.1-8B-Instruct's accuracy from **38.5%** to **51.1%**, and consistency from **59.6%** to **76.2%**. [1]

1 Introduction

In the field of natural language understanding, contradiction detection has long served as a fundamental benchmark for evaluating a model's capacity for deep semantic comprehension (Su et al., 2024; Hsu et al., 2021; Li et al., 2024; Zheng et al., 2022). Traditionally, research has focused on identifying sentence-pair inconsistencies by natural language inference(NLI) methods (Lendvai et al., 2016; Badache et al., 2018). However, pairwise approaches are limited in detecting document-level self-contradictions, especially those spanning non-adjacent or multiple sentences. With a computational complexity of $n(n-1)/2$ conflict checks for n sentences, these methods are expensive and often fail to capture deeper semantic contradictions involving more than two sentences.

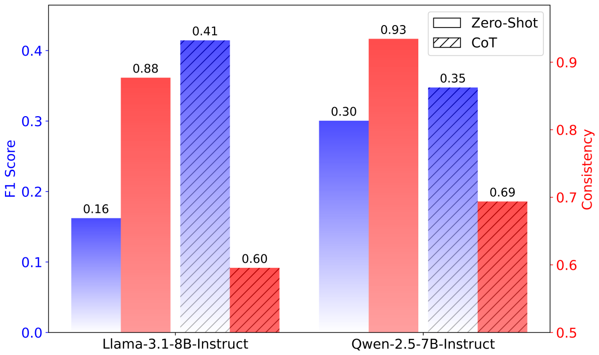


Figure 1: F1 Score and Consistency of LLaMA and Qwen Models Using Zero-Shot and CoT prompting strategy on ContraDoc. In DSCD tasks, reasoning improves LLM performance but reduces answer consistency, leading to greater variability in responses to the same question and introducing potential unreliability.

To address these challenges, Document-level Self-Contradiction Detection (DSCD) (Hsu et al., 2021) has gained increasing attention. DSCD takes a multi-sentence document as input and predicts a binary label indicating whether any self-contradictions exist, going beyond pairwise contradiction detection to assess document-level consistency. ContraDoc (Li et al., 2024) extends this concept by not only identifying the presence of contradictions in a document but also localizing the specific sentences in which they occur. Recently, Chain-of-Thought (CoT) (Wei et al., 2022) has been applied to the DSCD task. CoT enables the model to perform step-by-step reasoning to identify where the contradictions lie and why they occur, potentially improving interpretability and accuracy. However, CoT yields only marginal per-

---

[1] Data and Code: https://anonymous.4open.science/r/RRC-DSCD-555.

formance gains and introduces inconsistencies in the model's responses to the same input. As shown in Figure 1, applying the CoT strategy significantly reduces response consistency, thereby undermining the reliability of the model's predictions.

Why does the model produce different answers to the same question when using CoT? Through in-depth case analyses (e.g., Figure 13), we observe that failure cases often stem from incomplete or overly diffuse reasoning based on a limited subset of relevant sentences. In one reasoning instance, the model may focus on a subset of sentences while overlooking others; in another, it shifts attention to a different subset entirely. Consequently, shifts in focus across different attempts can result in inconsistent judgments. This observation raises an important question: *Can both accuracy and consistency be improved simultaneously if the reasoning chain takes a more comprehensive account of potentially contradictory sentences?*

In pursuit of this goal, we propose a method that explicitly trains the model to incorporate all potentially contradictory sentences into the reasoning process. Our approach consists of two key training processes: (1) supervised fine-tuning (SFT) using CoT data distilled from a strong teacher model to help the model learn basic reasoning patterns, and (2) reinforcement learning (RL) for iterative self-improvement, enhancing the model's overall reasoning ability in DSCD.

In the SFT stage, there is a lack of automatically generated data for the DSCD task. To address data scarcity and high annotation costs, we propose a fully automated DSCD sample synthesis pipeline based on the StorySumm (Subbiah et al., 2024) and REPLIQA (Monteiro et al., 2024) datasets. Then, we use Deepseek R1 to obtain distilled CoT data by running the pipeline. In the RL phase, we employ the GRPO algorithm (Shao et al., 2024), which omits the value function and facilitates self-iterative optimization via multi-output comparison. Our reward function aims to address the challenges in the reasoning process by optimizing multiple dimensions of reasoning. To enhance accuracy, we designed the *Accuracy Reward*, which focuses on contradiction detection and localization. By encouraging the reasoning chain to cover potentially contradictory sentences, the *Reference Coverage Reward* promotes the comprehensiveness of the reasoning process. Meanwhile, the *Format Reward* ensures consistency in the reasoning format, thereby guaranteeing the correctness of the reasoning structure. These designs synergistically guide the model to generate more comprehensive reasoning paths.

Experimental results show our method achieves around **10%** improvements over the baseline across tasks, demonstrating its effectiveness. Specifically, Llama-3.1-8b-Instruct improves accuracy by **10.6%** on *Binary Judgment*, and by up to **16.6%** in consistency metrics. In summary, this work makes the following three key contributions:

1) To the best of our knowledge, we are the first to consider the problem of consistency in CoT reasoning for the DSCD task and mitigate it using reference coverage-based reinforcement learning.

2) We propose a fully automated pipeline for generating DSC examples, which effectively addresses the bottleneck of costly human annotations.

3) We demonstrate the effectiveness and robustness of our method by training on our constructed out-of-domain dataset and achieving strong performance on the Contradoc benchmark.

2 Related Work

2.1 Contradiction Detection

Contradiction detection in text is a critical task in NLU, aimed at identifying inconsistencies or conflicting information within textual data. Most existing research has centered on the NLI framework, where contradictions are evaluated at the sentence-pair level (Lendvai et al., 2016; Badache et al., 2018). Recent efforts have extended contradiction detection to dialogue systems (Zheng et al., 2022; Wen et al., 2024) and question-answering tasks (Fortier-Dubois and Rosati, 2023).

However, identifying self-contradictions at the document level remains a significant challenge due to the increased contextual complexity and long-range dependencies. Hsu et al. (2021) framed DSCD as a binary classification problem. Li et al. (2024) extends this concept and introduces ContraDoc, a manually annotated dataset. Despite this, their work did not fully exploit the capabilities of these models in this field. In this paper, we propose a fully automated pipeline for generating DSC examples and significantly improving model reliability on this task through the RL method.

2.2 RL for LLMs Reasoning

RL has demonstrated considerable potential in enhancing the reasoning capabilities of LLMs across various domains, such as mathematics (Guo et al., 2025), code generation (OpenAI et al., 2025), and RAG (Jin et al., 2025). These tasks often require complex, multi-step decision-making, which is challenging for traditional SFT methods.

Initial alignment of model outputs with human preferences was achieved via RLHF (Ouyang et al., 2022; Christiano et al., 2017). To address the complexity of actor-critic methods like PPO (Schulman et al., 2017), more efficient approaches emerged. DPO (Rafailov et al., 2023) simplifies training by removing the learned critic, though its off-policy nature limits generalization (Pang et al., 2024). More recently, GRPO (Shao et al., 2024) enhances stability through improved advantage estimation. Despite these advancements, RL-based approaches for document-level reasoning, particularly for DSCD, remain underexplored. In this work, we extend the GRPO framework to fine-tune LLMs for DSCD, addressing a crucial yet under-investigated challenge in long-form reasoning.

3 Method

3.1 Preliminary

According to the definition in ContraDoc (Li et al., 2024), the DSCD task is divided into two parts: *Binary Judgment* and *Judge then Find*. The former requires a binary decision on whether a document d contains a contradiction. *Judge then Find* additionally requires locating supporting evidence sentences, enabling a more thorough evaluation of the model's reasoning.

As previously noted, CoT-based models often exhibit inconsistency by producing different answers to the same input. To quantitatively assess this, we represent i-th inference pass as a binary vector $\mathbf{v}^{(i)} = [v_1^{(i)}, \dots, v_N^{(i)}] \in \{0,1\}^N$, where $i \in \{1, \dots, T\}$ and $v_k^{(i)} = 1$ if the model's prediction on the k-th sample is correct, and 0 otherwise. Here, T denotes the total number of independent inference passes performed on the model, and N is the number of evaluation samples. Consistency between two passes $\mathbf{v}^{(i)}$ and $\mathbf{v}^{(j)}$ is measured by:

$$\text{Sim}(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}) = \frac{1}{N} \sum_{k=1}^{N} \mathbb{I}\left[v_k^{(i)} = v_k^{(j)}\right], \quad (1)$$

where $\mathbb{I}[\cdot]$ is the indicator function. This metric reflects the proportion of matching predictions across two inference passes. The overall consistency of a model on a given set of samples is then computed as the average pairwise similarity across all inference vectors:

$$\text{Consistency} = \frac{2}{T(T-1)} \sum_{1 \le i < j \le T} \text{Sim}(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}), \quad (2)$$

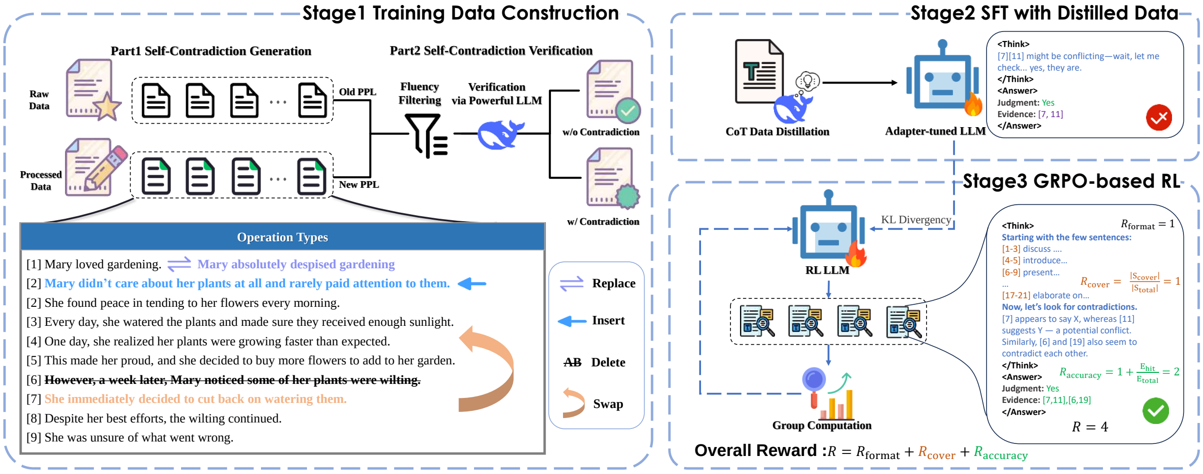
We evaluate performance under both Zero-Shot and CoT prompting strategies. As illustrated in Figure 1, CoT prompting leads to improvements in task performance. However, this enhancement is accompanied by a reduction in response consistency, indicating a trade-off between accuracy and stability in reasoning patterns.

Why does the model give different answers to the same question under CoT prompting? Case analyses (see Appendix C) reveal that inconsistencies often arise from incomplete or shifting reasoning chains that overlook relevant contradictory sentences. This raises a key question: Can accuracy and consistency be improved by ensuring all potentially contradictory sentences are included in the reasoning process? To address this, we propose a method that explicitly trains the model to incorporate such sentences. Our approach combines (1) SFT using CoT data distilled from a strong teacher model to teach core reasoning patterns, and (2) RL for iterative self-improvement in DSCD.

3.2 Construction of the Training Data

Training models require substantial data, but existing approaches rely heavily on manual annotation and lack large-scale, high-quality datasets. To support our two-stage training framework (SFT and RL), we introduce an automated pipeline for generating DSCD training data at scale, addressing the limitations of labor-intensive data construction methods. We define contradiction types and generation methods, enabling LLMs to autonomously select modification locations. This process also provides a rationale for each contradiction and uses LLMs for automatic verification.

We selected two datasets unlikely to be included in LLM training data as the original document sources: StorySumm (Subbiah et al., 2024), consisting of 32 short stories, and REPLIQA (Monteiro et al., 2024), which includes 17 thematic domains. From REPLIQA, we chose two subsets, `repliqa_1` and `repliqa_2`, as the basis for our dataset construction. To ensure factual consistency and data diversity, we applied a preprocessing step that separated positive and negative samples, ensuring no pair originated from the same source document. As shown in Figure 2, the pipeline consists



[1] Mary loved gardening. ⇄ Mary absolutely despised gardening

[2] Mary didn't care about her plants at all and rarely paid attention to them. ←

[2] She found peace in tending to her flowers every morning.

[3] Every day, she watered the plants and made sure they received enough sunlight.

[4] One day, she realized her plants were growing faster than expected.

[5] This made her proud, and she decided to buy more flowers to add to her garden.

[6] ~~However, a week later, Mary noticed some of her plants were wilting.~~

[7] She immediately decided to cut back on watering them.

[8] Despite her best efforts, the wilting continued.

[9] She was unsure of what went wrong.

```
<Think>
[7][11] might be conflicting—wait, let me check... yes, they are.
</Think>
<Answer>
Judgment: Yes
Evidence: [7, 11]
</Answer>
```

```
<Think>
Starting with the few sentences:
[1-3] discuss ....
[4-5] introduce...
[6-9] present...
...
[17-21] elaborate on...
Now, let's look for contradictions.
[7] appears to say X, whereas [11]
suggests Y — a potential conflict.
Similarly, [6] and [19] also seem to
contradict each other.
</Think>
<Answer>
Judgment: Yes
Evidence: [7,11],[6,19]
</Answer>
```

$R_{\text{format}} = 1$

$R_{\text{cover}} = \frac{|S_{\text{cover}}|}{|S_{\text{total}}|} = 1$

$R_{\text{accuracy}} = 1 + \frac{E_{\text{hit}}}{E_{\text{total}}} = 2$

$R = 4$

Overall Reward : $R = R_{\text{format}} + R_{\text{cover}} + R_{\text{accuracy}}$

Figure 2: Overview of our proposed framework. **Stage 1**: A self-contradictory document is generated by applying one of the operations—*insert*, *replace*, *delete*, or *swap*—and then verified using LLMs. See Appendix B for details on the rationale behind generating self-contradictions. **Stage 2**: A powerful model is used to distill CoT data, which is then used to fine-tune Our model via SFT. **Stage 3**: A fine-grained reward function is constructed and combined with a GRPO-based RL method to enhance the model's reasoning coverage.

of two parts: *contradiction generation* and *contradiction verification*. The resulting dataset includes 2,754 positive and 4,276 negative samples.

3.2.1 Self-Contradiction Generation

To maximize coverage of self-contradiction, we define six types following prior work (De Marneffe et al., 2008): *Attitudinal*, *Definition*, *Logical*, *Factual*, *Scope*, and *Temporal*. We employ LLMs to develop an automated strategy for generating contradictions via four primary operations: *Insert*, *Delete*, *Replace*, and *Swap*. By precisely controlling the model's behavior through tailored prompting techniques, we guide it to generate specific forms of self-contradictory statements. Examples of each operation type are provided in Figure 2.

During implementation, we select a subset of the dataset for contradiction generation. Using DeepSeek-V3 (DeepSeek-AI, 2024), we generate modified samples $\hat{d}_i$, along with a set of labeled contradictory statements $\mathcal{S} = \{s_1, s_2, \ldots\}$ and corresponding explanations r, based on carefully designed prompts tailored to each operation type. Detailed prompt templates and the format of the model are provided in the Appendix B.

3.2.2 Self-Contradiction Verification

Maintaining the quality and reliability of the generated self-contradiction data requires rigorous validation. We employ a two-stage verification framework consisting of: (1) Fluency Filtering and (2) Contradiction Verification, aimed at enhancing the quality of the data.

Fluency Filtering. Ensuring comparable fluency between the original and revised documents is essential. Therefore, we adopt the approach proposed in ContraDoc (Li et al., 2024) and refine it by introducing a relative threshold instead of an absolute one, mitigating the impact of document length on perplexity. Specifically, To quantify fluency shifts between the original document d and a candidate modification $\hat{d}_i$, we compare their perplexity scores (Jelinek et al., 1977). This ensures that each $\hat{d}_i$ maintains fluency within a permissible range relative to d. Formally, for each $\hat{d}_i$, we require:

$$\frac{ppl(\hat{d}_i)}{ppl(d)} \leq \theta, \tag{3}$$

where $ppl(\cdot)$ denotes the perplexity of a given document, which we compute using the Llama-3.1-8B-Instruct model. In our experiments, we adopt a conservative threshold of $\theta = 1.01$ to ensure that the modified text remains highly fluent and semantically consistent with the original.

Contradiction Verification. While rule-based filtering effectively removes the majority of non-compliant data, a small portion remains. To further refine the dataset, we apply a model-based approach with customized judgment prompts for more rigorous self-contradiction verification. Given a modified document $\hat{d}_i$, a set of model-generated contradictory statements $\mathcal{S}$, and a rationale r explaining the contradiction, the verification process evaluates the validity of the contradiction. Verification is conducted by querying an LLM $\mathcal{M}$, formalized as Verdict $= \mathcal{M}(\hat{d}_i, \mathcal{S}, r)$, where Verdict indicates whether a valid contradiction is detected. This method differs from DSCD by verifying only local conflicts using suffer reason r. We manually sampled a small portion of the data to validate the process. See Appendix B.2 for details.

3.3 Reinforcement Learning for Broader Reasoning and Sharper Detection

To improve the comprehensiveness of the LLM's reasoning in the DSCD task, thereby enhancing both accuracy and consistency, we propose a two-stage training strategy. In the first stage, we fine-tune the model by distilling high-quality reasoning chains from DeepSeek-R1 (DeepSeek-AI, 2025), enabling the model to preliminarily acquire effective reasoning capabilities and format. In the second stage, we introduce some task-specific reward functions and apply GRPO (Shao et al., 2024) algorithm to enhance the model's reasoning comprehensiveness through RL.

3.3.1 CoT Data Distillation for SFT

As shown in Figure 1, CoT reasoning improves model performance in DSCD. However, the generated CoT responses often suffer from incomplete logic and incorrect formatting, affecting reward signal sparsity during RL optimization. To overcome these issues, we employ knowledge distillation, utilizing high-quality reasoning chains from the powerful model, DeepSeek-R1, to help the model initially learn more reliable reasoning.

We use DeepSeek-R1's outputs on the distillation set as supervisory signals. To ensure distillation quality, we only retain responses where the powerful model's final prediction matches the ground truth. Specifically, for an input sample x, the teacher model's answer y_t, its final judgment a_t, and the true label a^*, we select samples satisfying:

$$\mathcal{D}_{\text{filtered}} = \{(x, y_t) \mid a_t = a^*\}. \tag{4}$$

Finally, we fine-tune the model using high-quality reasoning chain distillation signals.

3.3.2 Rule-based Reinforcement Learning and Reward Design

SFT has preliminarily improved output format, but it falls short of enhancing comprehensive reasoning. Specifically, it does not fully incorporate potentially contradictory sentences into the reasoning chain. As noted by Chu et al. (2025), SFT mainly promotes memorization and lacks generalization ability. Therefore, we adopt the GRPO algorithm, which eliminates the need for a separate value estimator and reduces the amount of data required. Since the training data does not include intermediate annotation information, the RL process mainly relies on reward signals from the final result. We designed three reward functions, namely R_{accuracy}, R_{cover}, and R_{format}, to guide the model's learning.

Accuracy Reward (R_{accuarcy}). To directly reward the accuracy of the model's responses, we define the Accuracy Reward. For positive samples, the reward reflects both the correctness of the model's judgment and its ability to identify contradictory sentences. For negative samples, the reward is based solely on the correctness of the judgment. To simplify, we define indicator variables $j = \mathbb{I}(\text{judge} = \text{True})$ and $e = \mathbb{I}(\text{evidence hit} = \text{True})$. Thus, the reward is defined as:

$$R_{\text{pos.}} = j \cdot \left(-1 \cdot (1 - e) + \left(1 + \frac{m}{n}\right) \cdot e\right), \tag{5}$$

$$R_{\text{neg.}} = j, \tag{6}$$

where m denotes the number of correctly matched contradiction sentences, and n is the total number of gold conflict sentences.

This formulation ensures that a correct judgment, where the evidence is accurately identified, is highly rewarded. On the other hand, a correct judgment without valid evidence incurs a penalty, inspired by Evidence Hit Rate (EHR) metric (see Section 4.2). Lastly, an incorrect judgment does not receive any reward. The variable judge is extracted using a regular expression from the content between the `<answer>...</answer>` tags to determine the model's final decision. This reward mechanism guides the model toward accuracy while penalizing redundant or irrelevant information, thereby enhancing answer accuracy.

Reference Coverage Reward (R_{cover}). We define the Reference Coverage Reward to quantify the extent to which the model's reasoning chain incorporates content from the input document, reflecting the comprehensiveness of its reasoning process. It is formally defined as:

$$R_{\text{cover}} = \frac{|\mathcal{S}_{\text{covered}}|}{|\mathcal{S}_{\text{total}}|}, \tag{7}$$

Specifically, we assign numbered tags $[i]$ to each sentence in the input document to indicate position. Let $\mathcal{S}_{\text{total}}$ denote the full set of sentences, and $\mathcal{S}_{\text{covered}}$ denote the subset of sentences explicitly referenced during the model's reasoning chain. This subset is derived from span expressions such as ($[i]$), ($[i$-$j]$), or ($[i]$-$[j]$), where each expression indicates a set of sentence indices—e.g., [1-3] denotes the set $\{1, 2, 3\}$. Using span expressions reduces the model's focus on meaningless sentences, improving reasoning efficiency. This recall-style reward encourages the model to incorporate a wider range of relevant content, thereby facilitating more comprehensive and grounded reasoning, further enhancing the model's accuracy and consistency.

Format Reward ($R_{\textbf{format}}$). To prevent forgetting of the response format acquired during the SFT stage throughout RL. We design Format Reward R_{format} to assess whether the model's output adheres to a predefined structural format. Specifically, the reward is defined as a binary indicator function:

$$R_{\text{format}} = \mathbb{I}(\text{Format is correct}), \quad (8)$$

An output is considered correctly formatted if it satisfies all of the following conditions: (1) the reasoning process is entirely enclosed within `<think>...</think>` tags and the final answer is fully encapsulated within `<answer>...</answer>` tags and explicitly includes the phrases `Judgment` and `Evidence` to denote the conclusion and its evidence, respectively; (2) no content appears outside these specified tags. Outputs violating any of these requirements receive zero reward.

4 Experiments

4.1 Experimental configurations

Dataset. For evaluation, we selected the **ContraDoc** dataset, a key benchmark for DSCD. For RL, we sampled 1,000 positive and 1,000 negative examples from the constructed Training dataset. Additionally, we applied DeepSeek-R1 distillation to the remaining data to extract 1360 positive samples and 1568 negative instances, which were subsequently used for SFT.

Baseline. Our experiment evaluates two popular open-source instruction-tuned LLMs, Llama-3.1-8B-Instruct (Llama-3.1) and Qwen-2.5-7B-Instruct (Qwen-2.5). Given the lack of advanced existing methods for the DSCD task, we adopt Zero-Shot and CoT as baselines for comparative analysis.

Hyperparameters. All experiments were conducted on 8 * NVIDIA L20 GPUs. The reported results represent the average over five independent runs to ensure the stability and reliability of the outcomes. Detailed hyperparameter settings are provided in the Appendix A.

4.2 Evaluation Metrics

We evaluate the model's performance using standard metrics defined in prior work, including **Precision**, **Recall**, **F1**, and **Accuracy**. We define $J(d)$ to detect the presence of document self-contradiction and $V(E)$ to evaluate whether evidence is correctly identified. For the *Judge then Find* task, when the model's answer is `yes`, we apply the BERTScore (Sun et al., 2022) metric to account for minor linguistic variations. If any selected evidence sentence has a BERTScore Precision or Recall greater than 0.98 compared to the reference, it is considered semantically equivalent to the ground truth and deemed correct. The model's prediction is deemed correct only when $J(d) \wedge V(E) = \text{True}$. Additionally, we adopt the **EHR** metric, which represents the proportion of samples for which correct evidence is successfully identified, given that the model has predicted `yes`. To assess the model's stability across multiple responses, we utilize the **Consistency** metric as defined in Equation (2). Furthermore, we introduce a new metric, **Reliability**, defined as the product of the model's F1 and its Consistency: $\mathcal{R} = \text{F1} \cdot \mathcal{C}$, which reflects the overall trustworthiness of the model by jointly capturing its accuracy and stability.

4.3 Main Results

RL yields substantial improvements over the baseline in both ACC, F1, and EHR. Table 1 presents a performance comparison between our method and the baseline across two tasks. In the *Binary Judgment* task, the RL method yields F1 score improvements of 5.3% and 4.7% on Llama-3.1 and Qwen-2.5, respectively, indicating a notable enhancement in judgment accuracy. Even more remarkably, accuracy increases by 10.6% and 4.7% on the two models, respectively, underscoring the efficacy of our method in improving overall classification correctness.

The advantage of our method becomes even more pronounced in the more challenging *Judge then Find* task. This task places higher demands on the model's reasoning and information extraction capabilities. Under this setting, the RL strat-

Model	Method	Binary Judgment				Judge then Find				
		Precision	Recall	F1 Score	Acc.	Precision	Recall	F1 Score	Acc.	EHR
Llama-3.1-8B-Instruct	Zero-Shot	0.585	0.183	0.279	0.523	0.435	0.100	0.162	0.481	0.544
	CoT	0.518	**0.701**	0.596	0.521	0.399	0.432	0.415	0.385	0.616
	SFT	0.575	0.699	0.631	0.588	0.450	0.423	0.436	0.449	0.605
	Ours	**0.618**	0.683	**0.649**	**0.627**	**0.517**	**0.452**	**0.482**	**0.511**	**0.661**
Qwen-2.5-7B-Instruct	Zero-Shot	0.599	0.447	0.512	0.570	0.434	0.230	0.300	0.461	0.514
	CoT	0.569	0.541	0.555	0.562	0.420	0.297	0.348	0.439	0.548
	SFT	0.579	**0.597**	0.588	0.578	0.461	0.372	0.412	0.465	0.623
	Ours	**0.619**	0.586	**0.602**	**0.609**	**0.519**	**0.390**	**0.445**	**0.511**	**0.665**

Table 1: Performance metrics related to the accuracy of Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct on *Binary Judgment* and *Judge then Find* Tasks in ContraDoc. The best results are highlighted in bold, and the second-best are underlined. All results are averaged over five runs.

Model	Method	Judge		Judge then find	
		Cons.	Rel.	Cons.	Rel.
Llama-3.1 8B-Instruct	CoT	0.585	0.349	0.596	0.247
	SFT	0.696	0.439	0.734	0.320
	Ours	**0.723**	**0.469**	**0.762**	**0.367**
Qwen2.5 7B-Instruct	CoT	0.624	0.346	0.694	0.241
	SFT	0.670	0.394	0.729	0.300
	Ours	**0.684**	**0.412**	**0.745**	**0.331**

Table 2: Consistency and Reliability Evaluation of two LLMs on ContraDoc dataset.

egy leads to F1 improvements of 6.8% and 9.7%, with accuracy gains of 12.6% and 7.2%, respectively. Notably, the Llama-3.1 model shows a dramatic leap in accuracy after incorporating the RL method, suggesting a significant boost in its reasoning ability enabled by our strategy. Furthermore, in terms of *EHR*, our method achieves additional gains of 4.5% and 11.7% on Llama-3.1 and Qwen-2.5. These results indicate that reinforcement learning not only enhances the final decision-making accuracy but also substantially improves the model's capability to locate critical supporting information.

RL significantly enhances the consistency and reliability of reasoning chains compared to the baseline. As shown in Table 2, RL significantly improves response consistency compared to CoT method. Specifically, RL achieves gains of 16.6% and 5.1% on the *Binary Judgment* task, and 27.9% and 6.8% on the *Judge then Find* task, for Llama-3.1 and Qwen-2.5, respectively. Furthermore, in terms of stability metrics, RL outperforms both CoT and SFT methods across models and tasks, as illustrated in Table 2, with particularly notable gains on the *Judge then Find* task.

Although the consistency scores remain lower than those under the zero-shot setting, this does not undermine the effectiveness of our method. The discrepancy is mainly due to how we compute consistency, based on whether the model's answers are correct. As shown in Table 1, zero-shot models frequently yield incorrect answers across most evaluation instances. However, these responses often exhibit internal logical coherence, resulting in deceptively high consistency scores. In contrast, our method enhances consistency within the context of reasoning chain more substantively and reliably. This improvement reflects a genuine alignment between correctness and internal coherence, rather than superficial fluency.

4.4 Ablation study

Model	Method	F1	Acc.	EHR	Cons.	Rel.	Cov.
Llama-3.1 8B-Instruct	SFT	0.436	0.449	0.605	0.734	0.320	0.245
	R_{format}	0.440	0.459	0.596	0.729	0.334	0.259
	$R_{\text{format\&accuracy}}$	0.450	0.496	0.625	0.757	0.341	0.249
	Ours	**0.482**	**0.511**	**0.661**	**0.762**	**0.367**	**0.849**
Qwen2.5 7B-Instruct	SFT	0.412	0.465	0.623	0.729	0.300	0.267
	R_{format}	0.436	0.470	0.662	0.719	0.317	0.270
	$R_{\text{format\&accuracy}}$	0.437	0.493	0.630	0.732	0.320	0.268
	Ours	**0.445**	**0.511**	**0.665**	**0.745**	**0.331**	**0.879**

Table 3: Ablation study results on *Judge then Find* task. Specifically, R_{format} denotes the use of only Format Reward, whereas $R_{\text{format\&acc}}$ represents the use of both the Format and Accuracy Reward. 'Cons.', 'Rel.', and 'Cov.' denote the metrics for consistency, reliability, and reasoning sentence coverage rate, respectively.

The ablation results shown in Table 3 demonstrate the effectiveness of each reward in our method. Utilizing only Format Reward improves performance over the SFT, indicating that output structure guidance is beneficial. Adding the Accuracy Reward further enhances F1 and Accuracy, suggesting that direct optimization towards task-specific objectives is crucial. Our full method

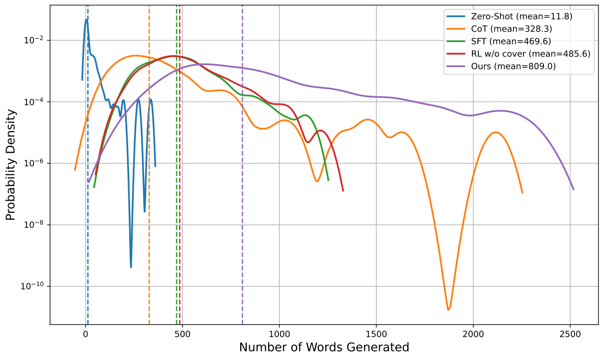


Figure 3: Comparison of output lengths using different methods on Llama-3.1. `RL w/o cover` refers to the method that does not incorporate R_{cover}.

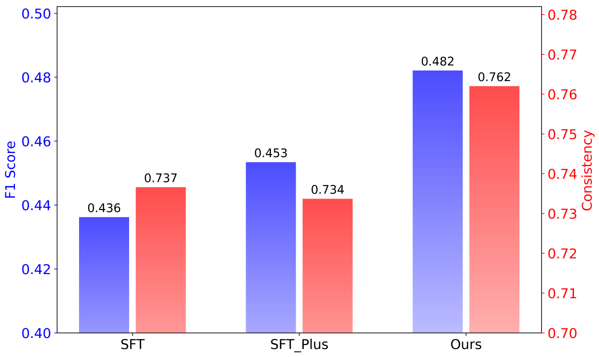


Figure 4: Comparison of the performance on the *Judge then Find* task among the baseline SFT, our method, and SFT_Plus (SFT with additional training data) based on the Llama-3.1-8B-Instruct model.

achieves the highest scores across all evaluation metrics validating the advantage of thinking comprehensively.

4.5 Further analysis

A more comprehensive and concise chain of thought is more effective. As shown in Table 3 and Figure 3, the integration of R_{cover} markedly enhances the thought coverage rate, increasing it from 24.5% to 84.9%, which corresponds to an improvement by a factor of approximately 3.47. Inevitably, this increase in coverage is accompanied by a 1.72× growth in output length. Notably, this increase is much smaller than the coverage gain, indicating that R_{cover} improves information density. We compared the output lengths across different methods, and the results illustrate the highly unstable reasoning pattern of CoT (the pronounced fluctuations in the probability density of output lengths). The model fine-tuned with SFT data distilled from a stronger model produces a more reasonable chain of thought while only slightly increasing output length. RL without R_{cover} method yields a more concise reasoning process. In contrast, our method, which reinforces reference coverage, achieves the best performance across all metrics, providing a chain of thought that is both comprehensive and concise.

Enhancing reasoning sentence coverage through RL leads to a more effective use of training samples compared to standard SFT. To investigate whether the inclusion of more data improves performance, we employed the reasoning chain data distillation approach mentioned earlier. Specifically, we distilled the two thousand samples used in the RL process and incorporated them into the original SFT dataset for further fine-tuning. As shown in Figure 4, the results indicate that, despite the increased data number, the performance of SFT_Plus did not significantly improve and even declined. This suggests that the additional data may have introduced more noise, undermining the model's effectiveness. In contrast, enhancing reasoning sentence coverage through RL substantially improved model performance, suggesting that it enables more efficient utilization of the available data. This approach notably enhanced the model's ability to detect self-contradictions at the document level, thereby increasing its reliability.

5 Conclusion

In this work, we address the challenge of Document-level Self-Contradiction Detection (DSCD), where Chain-of-Thought (CoT) prompting has shown significant response inconsistency due to incomplete or variable focus during inference. To tackle this issue, we propose a two-stage framework combining supervised fine-tuning with reinforcement learning. Our approach explicitly encourages the model to include all potentially contradictory sentences in its reasoning chain, guided by a novel reward design that balances accuracy, reference coverage, and structural consistency. To the best of our knowledge, we are the first to incorporate reinforcement learning into document-level self-contradiction detection. Experimental results demonstrate substantial improvements in LLMs' accuracy and consistency, highlighting their enhanced reliability in detecting document-level self-contradictions.

6 Limitations

Although our method effectively enhances the performance of large language models (LLMs) in detecting document-level self-contradictions, several limitations remain. Achieving a balance between maintaining a comprehensive reasoning chain and ensuring its conciseness continues to be a significant challenge. Furthermore, due to hardware constraints, our evaluation was restricted to models with approximately 8 billion parameters. Additionally, the lack of publicly available datasets annotated with location-specific information for document-level self-contradictions limits the scope of broader evaluation; consequently, all experiments were conducted exclusively on the ContraDoc dataset. Moreover, our current approach to constructing document-level self-contradictions focuses exclusively on the textual modality. The generation and detection of multimodal document-level contradictions, involving images, tables, or other media, represent promising directions for future research.

References

Ismail Badache, Sébastien Fournier, and Adrian-Gabriel Chifu. 2018. Predicting contradiction intensity: Low, strong or very strong? In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1125–1128.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Marie-Catherine De Marneffe, Anna N Rafferty, and Christopher D Manning. 2008. Finding contradictions in text. In *Proceedings of acl-08: Hlt*, pages 1039–1047.

DeepSeek-AI. 2024. Deepseek-v3 technical report.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Etienne Fortier-Dubois and Domenic Rosati. 2023. Using contradictions improves question answering systems. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 827–840.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Cheng Hsu, Cheng-Te Li, Diego Saez-Trumper, and Yi-Zhan Hsu. 2021. Wikicontradiction: Detecting self-contradiction articles on wikipedia. In *2021 IEEE international conference on big data (Big Data)*, pages 427–436. IEEE.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Piroska Lendvai, Isabelle Augenstein, Kalina Bontcheva, and Thierry Declerck. 2016. Monolingual social media datasets for detecting contradiction and entailment. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4602–4605.

Jierui Li, Vipul Raheja, and Dhruv Kumar. 2024. Contradoc: Understanding self-contradictions in documents with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6509–6523.

Joao Monteiro, Pierre-Andre Noel, Etienne Marcotte, Sai Rajeswar Mudumba, Valentina Zantedeschi, David Vazquez, Nicolas Chapados, Chris Pal, and Perouz Taslakian. 2024. Repliqa: A question-answering dataset for benchmarking llms on unseen reference content. *Advances in Neural Information Processing Systems*, 37:24242–24276.

OpenAI, :, Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, Jerry Tworek, Lorenz Kuhn, Lukasz Kaiser, Mark Chen, Max Schwarzer, Mostafa Rohaninejad, Nat McAleese, o3 contributors, Oleg Mürk, Rhythm Garg, Rui Shu, Szymon Sidor, Vineet Kosaraju, and Wenda Zhou. 2025. Competitive programming with large reasoning models.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–14.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zhaochen Su, Jun Zhang, Xiaoye Qu, Tong Zhu, Yanshu Li, Jiashuo Sun, Juntao Li, Min Zhang, and Yu Cheng. 2024. Conflictbank: A benchmark for evaluating the influence of knowledge conflicts in llm. *arXiv preprint arXiv:2408.12076*.

Melanie Subbiah, Faisal Ladhak, Akankshya Mishra, Griffin Thomas Adams, Lydia Chilton, and Kathleen McKeown. 2024. STORYSUMM: Evaluating faithfulness in story summarization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9988–10005, Miami, Florida, USA. Association for Computational Linguistics.

Tianxiang Sun, Junliang He, Xipeng Qiu, and Xuanjing Huang. 2022. Bertscore is unfair: On social bias in language model-based metrics for text generation. *arXiv preprint arXiv:2210.07626*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xiaofei Wen, Bangzheng Li, Tenghao Huang, and Muhao Chen. 2024. Red teaming language models for processing contradictory dialogues. *arXiv preprint arXiv:2405.10128*.

Chujie Zheng, Jinfeng Zhou, Yinhe Zheng, Libiao Peng, Zhen Guo, Wenquan Wu, Zheng-Yu Niu, Hua Wu, and Minlie Huang. 2022. Cdconv: A benchmark for contradiction detection in chinese conversations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 18–29.

A Hyperparameters

During the evaluation, we set the temperature to 0.7, top-p to 0.9, and max_new_tokens to 4096. All results are averaged over five independent runs to reduce variability.

During Supervised Fine-Tuning (SFT), the model is trained for three epochs on four NVIDIA L20 GPUs with a learning rate of 1×10^{-4} and a gradient accumulation step of 8. Training adopts BF16 mixed precision, gradient checkpointing, and DeepSpeed ZeRO-3 (Rajbhandari et al., 2021; Rasley et al., 2020) with CPU offloading. The maximum sequence length is 4096 tokens. Parameter-efficient fine-tuning is performed using LoRA (Hu et al., 2021) with rank 8, $\alpha = 32$, and a dropout rate of 0.1. The Adam optimizer (Kingma and Ba, 2014) is used with default settings for causal language modeling.

In the Reinforcement Learning (RL) phase, DeepSpeed ZeRO-3 with CPU offloading and BF16 precision are maintained. Training is conducted over one epoch with a learning rate of 5×10^{-5}, a micro-batch size of 4 per GPU, and a gradient accumulation step of 2. Gradient clipping with a maximum norm of 1.0 is applied. The maximum prompt and completion lengths are set to 8192 and 4096 tokens, respectively. LoRA is again applied with the same configuration as in SFT.

All experiments were conducted on 8 * NVIDIA L20 GPUs. The reported results represent the average over five independent runs to ensure the stability and reliability of the outcomes.

B Training and Testing Data

B.1 Training Dataset Sources

We employ two datasets that are highly unlikely to be present in the training data of large language models (LLMs): **StorySumm** (Subbiah et al., 2024) and **REPLIQA** (Monteiro et al., 2024). The **StorySumm** dataset consists of 32 short stories sourced from two fiction-oriented subreddits. To ensure both the quality and appropriateness of the content, the dataset excludes posts labeled as NSFW and those that received fewer than three upvotes. These community-vetted stories are concise—usually under one page—and lack user-written summaries. Their limited online visibility further reduces the likelihood of overlap with LLM training corpora, thus serving as a clean benchmark. In contrast, **REPLIQA** is a large-scale question-answering dataset comprising 17,954 human-authored synthetic documents, each approximately 1,000 words in length. These documents span 17 diverse thematic domains—including cybersecurity, folklore, and others—and depict entirely fictional scenarios, making them particularly well-suited for evaluating LLM generalization to genuinely novel content. From this dataset, we derive two evaluation subsets, denoted as `repliqa_1` and `repliqa_2`. To rigorously prevent data leakage during the partitioning process, we filter examples by unique ID prefixes and retain only one instance per prefix, thereby ensuring clear separation between the subsets.

B.2 Self-Contradiction Generation and Verification

We utilize LLMs to develop an automated strategy for generating contradictions. This process is underpinned by four primary operations: *Insert*, *Delete*, *Replace*, and *Swap*. Each operation type is illustrated through examples provided in Figure 2.

The implementation begins by selecting a subset of the dataset for contradiction generation. For *insert* and *replace* operations, factual sentences are first extracted via queries to a large language model. In contrast, *delete* and *swap* require only a single model query. We utilize the DeepSeek-V3 model (DeepSeek-AI, 2024) through its online API to generate modified samples $\hat{d}_i$ and corresponding contradiction sets $\mathcal{S} = \{s_1, s_2, \ldots\}$, each with an explanation r. Prompts used for each operation type are shown in Figures 7 to 12, and are designed to ensure alignment with the intended modifications. Outputs with empty or non-compliant JSON structures are discarded.

As shown in Figure 2, in the *insert* example, the added sentence contradicts the character's previously established dedication to gardening. In the *delete* case, removing the explanation that the plants were wilting renders the subsequent action—cutting back on watering—unjustified and illogical. In the *replace* case, if Mary dislikes gardening, it becomes inconsistent for her to take good care of the flowers and feel happy every day. In the *swap* case, cutting back on watering before realizing that the plants were growing rapidly introduces a temporal inconsistency—there would have been no need to reduce watering if the plants were thriving. All four types of operations can lead to self-contradictions within the document.

You are a helpful and detail-oriented language model tasked with detecting logical contradictions in documents. Below you are given:
Article: {Article}
Sentences suspected to contradict each other: {Statements}
Reason (r) explaining why these sentences may be contradictory: {Reason}
Your task is to carefully read the document and evaluate whether sentences truly contradict each other in the context of the full document. Use the reason as a clue but do not rely on it exclusively. Take into account nuances such as negation, temporal shifts, modality, or implied meanings.
Only answer "yes" or "no":

Figure 5: Prompt template for verifying suspected contradictions in a document.

To rigorously verify self-contradiction, we design specific prompts for positive and negative samples (see Figure 5 and Figure 6). For a positive sample, the model is provided with a modified document $\hat{d}_i$, a set of contradictory statements $\mathcal{S}$, and a rationale r explaining the contradiction. A response of `True` indicates a valid contradiction, and the sample is retained. Prompt template for negative samples, where only the original document d is provided. This setup mirrors the prompt format used during testing; samples yielding a no response—indicating no contradiction—are kept.

The task is to determine whether the article contains any self-contradictions. If yes, provide evidence by quoting mutually contradictory sentences in a list of strings in Python. If no, give an empty list.
Article: {Article}
Response: Form your answer in the following format (OR options are provided), Please answer the Judgment and Evidence in the prescribed format, Evidence must be a list that can be parsed by Python:
Judgment: yes OR no
Evidence: ["xxxxxx", "xxxxxx", ..., "xxxxxx"] OR []
please think step by step, and finally give the answer. (if using the CoT strategy)

Figure 6: Prompt template used for evaluation.

Leveraging the powerful model, this filtering step provides a coarse screening to remove obviously unsuitable documents but does not ensure 100% accuracy. We randomly sampled 200 entries from the dataset, among which 84.5% met the expected criteria. Experimental results demonstrate that such a dataset is sufficient for training purposes; however, further improvements in data quality are required if it is to be employed as a test set.

B.3 Data Statistics

We constructed a training dataset with 2,754 positive and 4,276 negative samples. For RL, 1,000 positive and 1,000 negative instances were randomly selected. The remaining data were distilled using DeepSeek-R1, yielding 1,360 positive and 1,568 negative samples for SFT. The evaluation was performed on the ContraDoc dataset, a standard benchmark for document semantic content detection. To examine RL's data efficiency, we also distilled the RL training data, obtaining 766 positive and 469 negative samples, which were added to the SFT dataset for further experiments.

C Case Study

In this example (see Figure 13), the Zero-Shot approach simply outputs "no" without any accompanying explanation, which is clearly inadequate for practical document-level inconsistency detection systems that require interpretable reasoning. In contrast, the Chain-of-Thought (CoT) prompting method guides the model to attend to specific sentences (e.g., sentences 9 and 11); however, due to incomplete or fragmented reasoning chains, the model still arrives at an incorrect conclusion. While CoT occasionally identifies the correct point of contradiction, it often arbitrarily overlooks other conflicting information in the document. This variability in focus across different runs results in unstable and inconsistent outputs.

By comparison, our Reinforced Reference Coverage method encourages the model to comprehensively consider relevant content throughout the entire document. This not only promotes a more thorough and balanced reasoning process but also enhances consistency across multiple runs, effectively addressing the instability observed in prior approaches.

Contradiction Types:
- **Attitudinal**: Contradiction arises from a difference in opinion or feeling.
- **Definition**: Contradiction arises from a difference in the meaning of a word or concept.
- **Logical**: Contradiction arises from a logical inconsistency.
- **Factual**: Contradiction arises from a difference in facts or events.
- **Scope**: Contradiction arises from a difference in the scope of a statement.
- **Temporal**: Contradiction arises from a difference in time or sequence of events.

Figure 7: Definitions of contradiction types used for sentence generation.

From the provided article, extract exactly five sentences corresponding to one of the following categories: Attitudinal, Definition, Logical, Factual, Scope, or Temporal. Please format your response as a Python-parsable list with exactly five elements, in the following format: ["sentence1", "sentence2", "sentence3", "sentence4", "sentence5"].
Article: {Article}

Figure 8: Prompt template used to extract factual sentences from the original document.

Article: {Article}

Generate a ‘contradicted_sentence‘ that contradicts the original sentence {Statement} in one of the following ways: Attitudinal, Definition, Logical, Factual, Scope, or Temporal. Insert the contradictory sentence at the appropriate position in the article. If there are ‘other_contradictory_sentences‘, they must clearly contradict the ‘contradicted_sentence‘.

Contradiction Types: {Contradiction Types}

The contradiction must create a scenario where the original statement cannot coexist. The goal is to introduce a fact that renders the original fact impossible, not merely negating it.

Good Examples: {Good Examples and Explanations}

Bad Examples: {Bad Examples and Explanations}

Return the result in the following JSON format, fully parsable by Python:

```
{
  "original_sentence": "xxx",//The original sentence
  "contradicted_sentence": "xxx",//The contradictory sentence
  "insert_position_sentence": "xxx",//The sentence before the insertion point
  "next_sentence_after_insert": "xxx",//The sentence after the insertion point
  "other_contradictory_sentences": ["sentence1", ...],//Supporting
      contradictory sentences
  "contradiction_type": "xxx",//Type of contradiction
  "contradiction_reason": "xxx"//Explanation of the contradiction
}
```

Figure 9: Prompt template for inserting a contradictory sentence.

Article: {Article}

Modify the original sentence {Statement} to introduce a contradiction with another sentence in one of the following ways: Attitudinal, Definition, Logical, Factual, Scope, or Temporal. If there are ‘other_contradictory_sentences‘, they must clearly contradict the ‘modified_sentence‘.

Contradiction Types: {Contradiction Types}

The contradiction must create a scenario where the original statement cannot coexist. The goal is to introduce a fact that renders the original fact impossible, not merely negating it.

Good Examples: {Good Examples and Explanations}

Bad Examples: {Bad Examples and Explanations}

Return the result in the following JSON format, fully parsable by Python:

```
{
  "original\_sentence": "xxx",//The original sentence
  "modified\_sentence": "xxx",//The modified sentence
  "other\_contradictory\_sentences": ["sentence1", ...],//Supporting
      contradictory sentences
  "contradiction\_type": "xxx",//Type of contradiction
  "contradiction\_reason": "xxx"//Explanation of the contradiction
}
```

Figure 10: Prompt template for modifying an original sentence to introduce contradiction.

Article: {Article}
Swap sentences in the article to create a contradiction in one of these areas: Attitudinal, Definition, Logical, Factual, Scope, or Temporal. If no contradiction is possible, return an empty list [] in 'modified_sentence_order'. If there are ‘other_contradictory_sentences‘, they must clearly contradict the ‘modified_sentence_order‘.
Contradiction Types: {Contradiction Types}
The contradiction must create a scenario where the original statement cannot coexist. The goal is to introduce a fact that renders the original fact impossible, not merely negating it.
Good Examples: {Good Examples and Explanations}
Bad Examples: {Bad Examples and Explanations}
Return the result in the following JSON format, fully parsable by Python:

```
{
  "original\_sentence\_order": ["sentence1", ...],//The original sentence
     order
  "modified\_sentence\_order": ["sentence2", ...],//The modified sentence
     order
  "other\_contradictory\_sentences": ["sentence3", ...],//Supporting
     contradictory sentences
  "contradiction\_type": "xxx",//Type of contradiction
  "contradiction\_reason": "xxx"//Explanation of the contradiction
}
```

Figure 11: Prompt template for modifying original sentence order to introduce contradiction.

Article: {Article}
Identify three sentences—A, B, and C—in the article. Sentence A and Sentence C should conflict, while Sentence B resolves the conflict. Removing Sentence B should create a contradiction in one of the following categories: Attitudinal, Definition, Logical, Factual, Scope, or Temporal. If there are other contradictory sentences, they must clearly contradict both Sentence A and Sentence C.
Contradiction Types: {Contradiction Types}
The contradiction must create a scenario where the original statement cannot coexist. The goal is to introduce a fact that renders the original fact impossible, not merely negating it.
Good Examples: {Good Examples and Explanations}
Bad Examples: {Bad Examples and Explanations}
Return the result in the following JSON format, fully parsable by Python:

```
{
  "sentencesA": "xxx",
  "sentencesB": "xxx",
  "sentencesC": "xxx",
  "other\_contradictory\_sentences": ["sentence1", ...],  // Supporting
     contradictory sentences
  "contradiction\_type": "xxx",  // Type of contradiction
  "contradiction\_reason": "xxx"  // Explanation of the contradiction
}
```

Figure 12: Prompt template for removing a key sentence to introduce contradiction.

Article: {ContraDoc_unique_id_3488771888_4}

Zero-Shot:

Judgment: no ✗

Evidence: [ ] ✗

CoT:

To determine whether the article contains any self-contradictions, I will carefully read through the article and look for any sentences that contradict each other.

Upon reading the article, I found a potential self-contradiction in sentences 9 and 11. Sentence 9 states: "whereupon Bêlit suddenly orders her crew to step back and spare Conan, being impressed with the Cimmerianś courage and ferocity (and being sexually attracted to him, as she immediately and forthrightly declares)."This suggests that Bêlit is impressed with Conan and wants to spare his life. However, sentence 11 states: "Survivors of butchered Stygian ships curse the name of Bêlit and her Cimmerian warrior with fierce blue eyes." This implies that Bêlit and Conan are actually enemies, and that Bêlit is not impressed with Conan. Considering these two sentences, it seems that the article contains a self-contradiction. Therefore, the answer is:

Judgment: yes ✔

Evidence: ["Bêlit suddenly orders her crew to step back and spare Conan", "Survivors of butchered Stygian ships curse the name of Bêlit and her Cimmerian warrior with fierce blue eyes"] ✗

Ours:

<think>

Okay, let's tackle this. The user wants to know if the article has any self-contradictions. I need to read through the article carefully and check for any statements that contradict each other. **Starting with the first few sentences:**

[1]Conan forcefully demands to board a trading ship, Argus....

[2-3]Captain Tito initially refuses, but Conan threatens him with ....

[4-5]We learn that Conan is a "complete land-lubber" with no sailing ....

[6-7]Upon reaching Kush's pirate-infested waters, the ship is ....

[8-9]After the captain is killed, Conan tries to rally the remaining ....

[10]Bêlit is impressed and attracted to him...Wait, the text then says that Conan refuses her offer.

[11-14] Yet, it goes on to say "they raid the Black Coast together," initiate a sadomasochistic romance, and form a strong partnership.

[15]Later, they explore ruins together ....

[17-18]Bêlit becomes possessed, gives reckless orders, her ....

[19-21]After her death, her spirit helps Conan defeat the monster....

Now, let's look for self-contradictions. The most obvious contradiction appears in this group of statements: "Bêlit offers Conan the chance to sail with her, be her chosen mate, and help lead her fierce warriors." "Unimpressed by this fiery woman, Conan refuses..."

This creates a contradiction between his stated decision and his subsequent actions. If this were a deliberate "change of heart" by the author, we would expect some narrative transition or internal reflection to show that Conan reconsidered. But the story provides no such moment. It jumps straight from ... the contradiction unresolved. Therefore ...

</think>

<Answer>

Judgment: yes ✔

Evidence: ["comparing Conan's grim god Crom with Bêlit's ... continuing their sexual romance", "Unimpressed by this fiery woman, **Conan refuses** and, for a time, they raid the Black Coast together brutally pillaging coastal towns and instilling fear into the superstitious natives"] ✔

</Answer>

Figure 13: Comparative Case Study of Llama-3.1-8B-Instruct Under Various Approaches.