How do LLMs' Preferences Affect Event Argument Extraction? CAT: Addressing Preference Traps in Unsupervised EAE

Yunhao Wei, Kai Shuang*, Zhiyi Li, Chenrui Mao
State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications
{2770278140, shuangk, iheadx, maochenrui1218}@bupt.edu.cn

Abstract

Large Language Models (LLMs) have significantly improved the performance of unsupervised Event Argument Extraction (EAE) tasks. However, the prevalent preferences in LLMs severely hinder their effectiveness in EAE, leading to what we term preference traps, namely, the prior knowledge trap, the sycophancy hallucination trap, and the output contradiction trap. Existing methods often fall into these traps due to low-quality prior knowledge, ambiguous instructions, and contradictory outputs. To address this issue, we propose **C**hoose-**A**fter-**T**hink (CAT), an unsupervised EAE framework based on LLMs. CAT divides the task into two stages: identification of event information (think stage) and selection of arguments from a candidate argument set for template filling (choose stage). Both stages employ countermeasures to address these preference traps, while the choose stage's completely constrained outputs satisfy EAE's structured-output requirements. Experimental results demonstrate that CAT (based on the local 7B model, zero-shot setting) matches the performance of the best DeepSeek-R1 API-accessible model, with a significantly lower time cost.[1]

1 Introduction

Event Argument Extraction (EAE), which aims to extract structured event information (arguments and their roles) from event texts, is a critical and highly challenging task in Information Extraction (IE). To address the limitations of supervised EAE, such as high training costs and poor model generalization, researchers have increasingly turned to unsupervised frameworks based on Large Language Models (LLMs), achieving significant progress (Xu et al., 2024). However, LLMs' preference traps in unsupervised EAE tasks severely degrade performance. Current mainstream approaches—including prompt engineering (Cai et al., 2024; Hong and Liu, 2024), chain-of-thought reasoning (Wei et al., 2023; Ma et al., 2024; Guo et al., 2025), and outputting in programming languages (Wang et al., 2023; Guo et al., 2024b)—only address some of these traps while succumbing to others (see detailed analysis in Section 5).

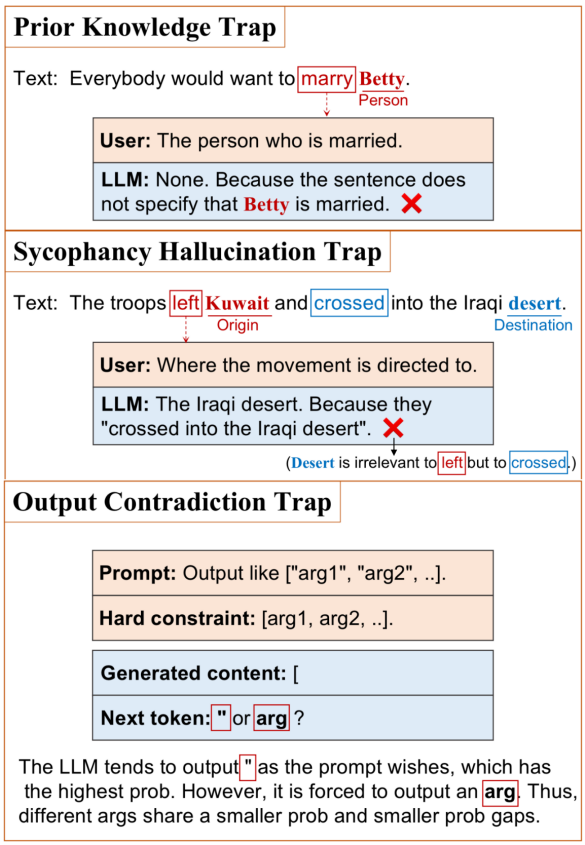

Figure 1: The illustration of three preference traps.

Just as many works have demonstrated LLMs' preferences and their impacts across various tasks (Panickssery et al., 2024; Li et al., 2024a; Nguyen et al., 2025), we summarize three preference traps in unsupervised EAE tasks that severely affect

*Corresponding author
[1]https://github.com/qawesrdtfy/CAT

Findings of the Association for Computational Linguistics: ACL 2025, pages 19529–19543
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

LLMs' performance, as shown in Figure 1. **Prior Knowledge Trap:** LLMs heavily rely on prior knowledge to identify event information, but low-quality knowledge often misleads them into ignoring or misidentifying arguments. **Sycophancy Hallucination Trap:** LLMs often generate answers that conform to users' viewpoints or requests, even if those answers are factually incorrect (Sharma et al., 2023). This behavior often results in misidentifying arguments, which are either nonexistent or irrelevant to the target event. **Output Contradiction Trap:** When LLMs are forced to generate content that conflicts with their intentions, the resulting inconsistencies can disrupt the prediction probability, significantly degrading their performance on EAE tasks.

Based on the above observations, we analyze root causes and utilize countermeasures for the three preference traps, which are simple yet effective:

Prior Knowledge Trap: The reason for falling into this trap is that prior knowledge provided fails to cover all actual annotations or that LLMs overfocus on specific patterns. The prior knowledge discussed here includes the definitions of event types and argument roles. First, we refine these definitions to better reflect actual meanings. Second, we adjust the expressions of these definitions to avoid the LLM becoming overly fixated on certain details.

Sycophancy Hallucination Trap: The reason for falling into this trap lies in the fact that LLMs prioritize the task requirement of "extracting event information". We attempt to reduce the erroneous outputs resulting from this trap. First, we provide clear output specifications for cases where arguments do not exist to reduce unspecified results. Second, we provide a description of the target event as the extraction scope to reduce irrelevant results.

Output Contradiction Trap: The reason for this trap is the contradiction between the forced output and natural output of the LLM. Addressing this conflict is key to the resolution of the trap. First, we prompt the LLM to output in its most preferred format. Second, we adjust the hard output constraints to align with the LLM's inherent habits. Thus, this contradiction is minimized as much as possible.

Based on the above analysis, we propose **C**hoose-**A**fter-**T**hink (CAT), an unsupervised EAE framework. We integrate the countermeasures above into CAT, enabling it to effectively address the preference traps. CAT splits EAE into two stages—Think (identifying event information) and Choose (outputting under constraints)—to reduce task difficulty. Unlike conventional token-level generation, our choose stage selects answers from a candidate argument set to fill the forced-output templates, considering every token's probability in the candidate argument and enabling direct control over the argument spans.

Experiments on three widely used EAE datasets demonstrate that our CAT outperforms the unsupervised baselines with lower time cost. We experimentally demonstrate the prevalence of preference traps across various families and scales of LLMs. Ablation studies confirm that CAT's employed methods successfully mitigate preference traps and enhance performance. Additionally, we investigate the time cost and model adaptability of CAT.

Our contributions include three aspects: **First**, we investigate and summarize the preference traps in LLM-based unsupervised EAE tasks. **Second**, we propose CAT, which effectively mitigates preference traps and achieves full controllability over LLMs' extraction results. CAT (based on the local 7B model, zero-shot setting) matches the performance of the best DeepSeek-R1 API-accessible model (Guo et al., 2025). **Third**, we conduct extensive experiments to explore the properties of CAT. EAE is one of the most challenging and representative IE tasks, and we hope our work can inspire further research on unsupervised unified IE frameworks.

2 Method

As illustrated in Figure 2, the CAT framework divides the EAE task into the think stage and the choose stage, with countermeasures integrated (Section 2.1). To address the prior knowledge trap, CAT refines the definitions and revises biased expressions therein (Section 2.2.1). To mitigate the sycophancy hallucination trap, CAT provides clear output specifications and restricts the extraction scope (Section 2.2.2). To tackle the output contradiction trap, CAT adjusts the prompts and hard constraints to guide the LLM output in its preferred format and natural habit (Section 2.2.3). The specific prompts are presented in Appendix A.

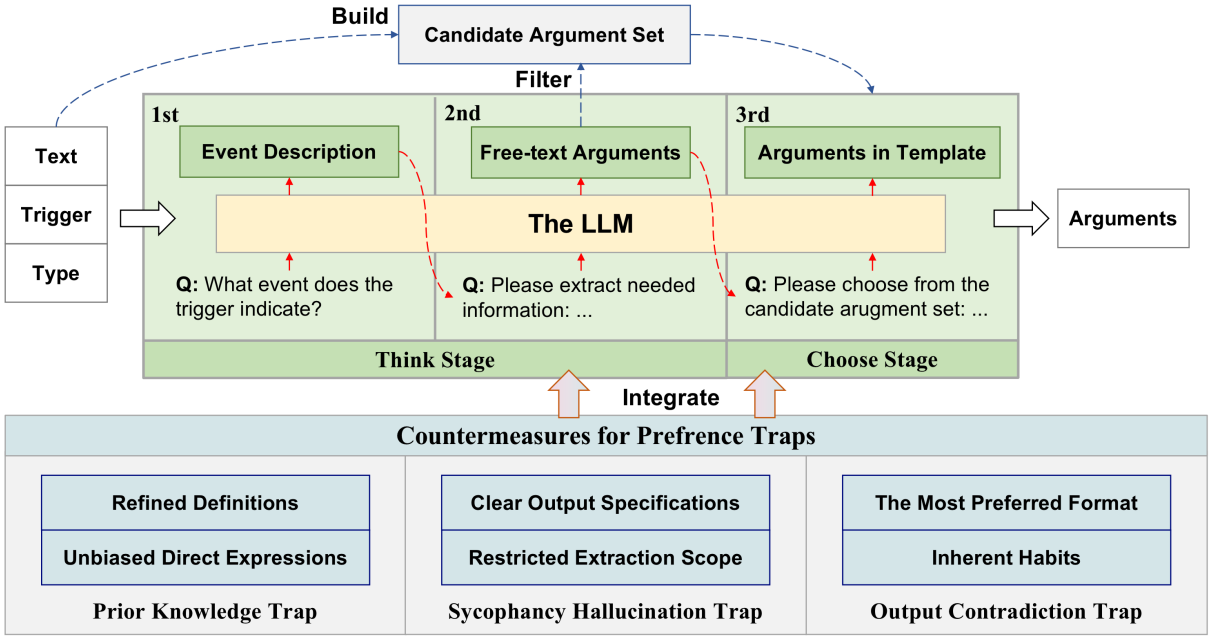

Figure 2: The framework of CAT. CAT divides the EAE task into the think stage and the choose stage. Countermeasures for preference traps are employed in the two stages.

2.1 Stage Division

The EAE task is divided into two stages: the think stage and the choose stage, involving three turns of dialogue with the LLM. To reduce error propagation, earlier inputs are reused in later turns, ensuring consistent interpretation of the event text, as indicated by the differently colored boxes in Appendix A.

2.1.1 Think Stage

The think stage focuses on identifying event information and generating results freely. This stage consists of two turns: the first turn is responsible for scoping the target event, and the second turn extracts from this scope.

The 1st turn requires the LLM to describe the event indicated by the trigger in the event text. This description serves as the extraction scope in subsequent turns to avoid extracting irrelevant information caused by the sycophancy hallucination trap.

The 2nd turn requires the LLM to extract event information within the scope from the 1st turn. Given the correlations among argument roles in an event, CAT jointly identifies all possible arguments and their roles.

2.1.2 Choose Stage

The choose stage involves the 3rd turn of dialogue with the LLM, outputting the final arguments under constraints based on the think stage's output.

The 3rd turn selects the final arguments for each role from the candidate argument set to fill the forced-output template. This set consists of entities in the text that may serve as event arguments. Unlike conventional token-level generation, CAT computes the average log-probability of tokens in the candidate argument as its selection score. Then, CAT picks the highest-scoring candidate each time and fills it into the template until the termination token or "None" outranks other candidates. This method effectively reduces the impact of abnormal initial token probabilities and ensures all generated content remains constrained.

2.1.3 Candidate Argument Set

We perform an algorithm to construct the candidate argument set, as shown in Figure 3. This algorithm extracts entities with varying degrees of modification from the event text, enabling direct control over the argument spans.

As preparation, we parse the event text into a dependency syntax tree $\mathbf{T}$ using NLP tools. We collect all dependency relations that can serve as subjects, objects, predicates, modifiers, or complements, or indicate coordination into a set $\mathbf{D}$.

The building stage 1 aims to identify the head words $\mathbf{C}$, which are bare entities: If the word's dependency relation is in $\mathbf{D}$, we add it to $\mathbf{C}$. The building stage 2 aims to expand each head word $\mathbf{c}$ to get entities with varying degrees of modifica-

tion: First, we find all subtrees rooted at **c**'s sons, denoted as **cT**. Second, we expand **c** bidirectionally and consecutively, using elements from **cT** as units. Third, we record all expansion results **s** from all elements in **C** as **S1**. **S1** is the candidate argument set for the text, near-perfect recall but too large.

Thus, we filter S1 for for each argument role, we remove arguments in **S1** that are absent from the results of the 2nd turn, obtaining **S**. **S** is the candidate argument set for each argument role, which can be directly used in the 3rd turn.

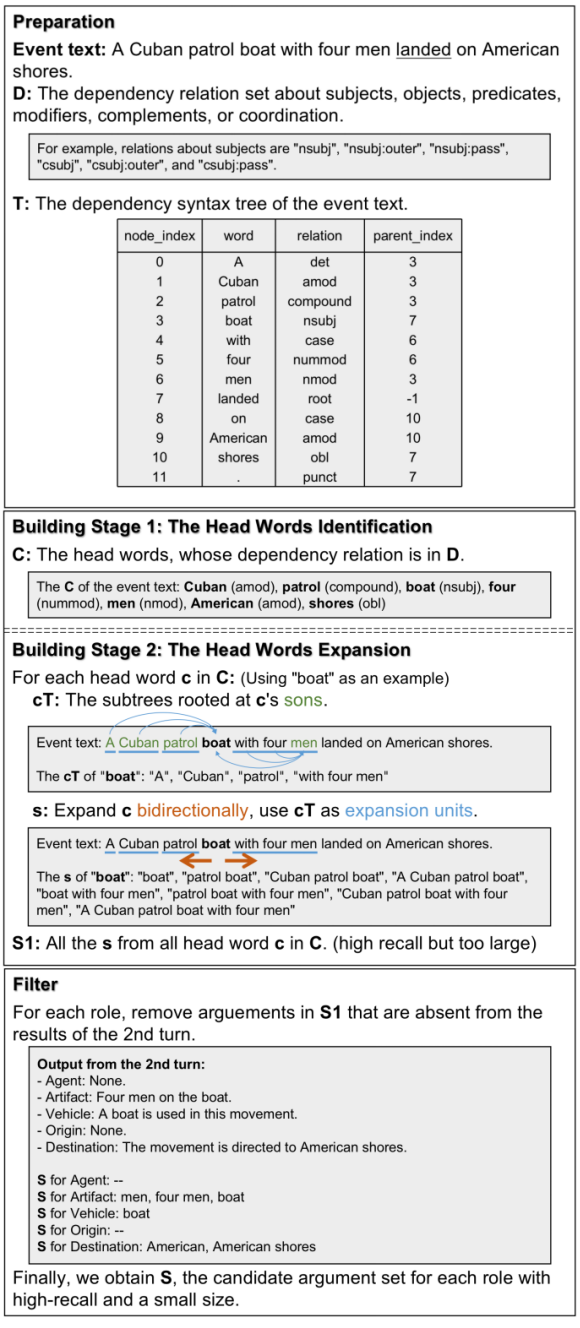

Figure 3: Construct the Candidate Argument Set: Prepare, Build and Filter

2.2 Countermeasures for Preference Traps

The three preference traps of LLMs significantly impact their performance on EAE tasks. To address these challenges, CAT employs two effective countermeasures for each trap.

2.2.1 Prior Knowledge Trap

In EAE tasks, LLMs often require crafted definitions of event types and argument roles to fully capture their intended meanings. However, low-quality definitions can mislead LLMs, resulting in the prior knowledge trap.

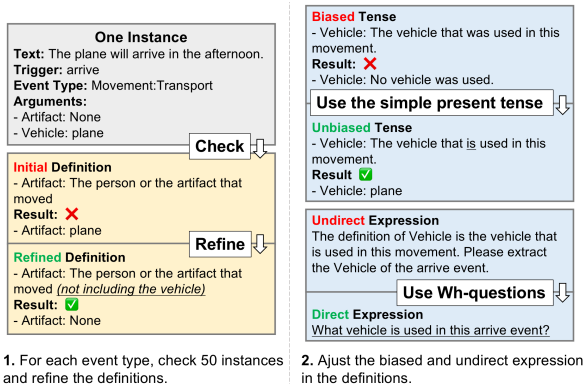

Figure 4: Countermeasures for the prior knowledge trap

To address this trap (Figure 4), the first common countermeasure is to refine these definitions. Initially, our event type definitions follow (Hsu et al., 2022), while argument role definitions follow (Doddington et al., 2004; Song et al., 2015). Then, for each event type, we sample 50 annotated instances and check their alignment with the definitions. If not, we refine the definitions.

We propose the second countermeasure to adjust the biased and undirect expressions. On one hand, we formulate definitions in the simple present tense to maintain objectivity and avoid temporal biases that might arise from other tenses. On the other hand, we describe argument role definitions using wh-questions and ask the LLM concisely and directly.

2.2.2 Sycophancy Hallucination Trap

To meet EAE requirements, LLMs often extract arguments that are not specified in the event text or irrelevant to the target event, leading to the sycophancy hallucination trap.

To address this trap (Figure 5), the first common countermeasure is to provide clear output specifications in prompts for cases where arguments do not exist. We propose the second countermeasure to

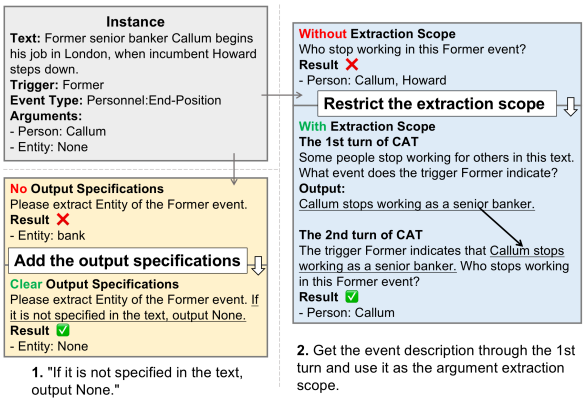

Figure 5: Countermeasures for the sycophancy hallucination trap

avoid irrelevant arguments. Specifically, we query the LLM for a description of the target event, which is exactly what the 1st turn does. The description is used as the extraction scope in the following turns.

2.2.3 Output Contradiction Trap

The LLM's output format depends on its preference, instructions, and hard constraints. The differences among them imply contradictions between the model, prompts, and already-generated content, i.e., the output contradiction trap, causing anomalies in token prediction probabilities.

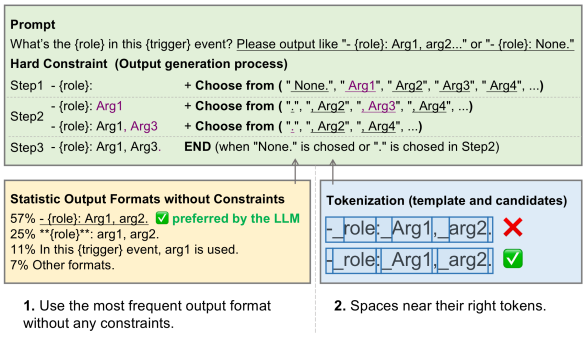

Figure 6: Countermeasures for the output contradiction trap

To address this trap (Figure 6), the first common countermeasure is to use the preferred output format. For a given LLM, we identify the most frequent output format without any constraints, then employ instructions and hard constraints to enforce this format. In this way, CAT achieves the alignment of the three factors.

We propose the second countermeasure to adjust hard constraints to align with the LLM's inherent habits. Specifically, we ensure all spaces in output templates and candidate arguments are attached to their right-side tokens, conforming to the tokenization patterns of most models (Sennrich et al., 2016).

3 Experiments Setup

3.1 Datasets

We conduct experiments on ACE05-E[2], ACE05-E$^+$ (Doddington et al., 2004), and ERE[3] (Song et al., 2015), which are popular datasets on EAE task. We follow (Wadden et al., 2019)'s and (Lin et al., 2020)'s pre-processing scripts on ACE05 and ERE. ACE2005 contains 33 event types and 22 argument roles, while ERE includes 38 event types and 21 argument roles. More dataset details are provided in Appendix Table 9.

3.2 Baselines

We compare our CAT framework with the following baseline models: (1) Degree (Hsu et al., 2022) performs supervised EAE tasks with rich weak supervision signals. (2) BART (Lewis et al., 2020) is a generative model fine-tuned on full data. (3) DeepSeek-R1 (Guo et al., 2025) is an API-accessible model that extracts arguments using Chain-of-Thought (CoT) reasoning. (4) CODE4STRUCT (Wang et al., 2023) outputs results in programming language, supporting few-shot and zero-shot. (5) DeepSeek-V3 (Liu et al., 2024) and GPT-4o (Hurst et al., 2024) exhibit strong zero-shot performance as API-accessible models. (6) Vanilia means the LLM extracts arguments without any optimization.

3.3 Evaluation Metric

We use Argument F1-score following baseline models (Wang et al., 2023) : We consider an argument to be correctly identified when the head word span of predicted text matches that of the human-annotated text (denoted as Arg-I); We consider an argument to be correctly classified if the role of a correctly identified argument matches that of the human annotation (denoted as Arg-C).

3.4 Implementation Details

We use Stanza (Qi et al., 2020) to perform dependency parsing on the event texts. The LLMs are deployed based on SGLang (Zheng et al., 2024a) on a single NVIDIA A40 GPU and adjust parameters to ensure reproducibility[4]. We limit generation

[2]https://catalog.ldc.upenn.edu/LDC2006T06
[3]https://catalog.ldc.upenn.edu/LDC2023T04
[4]https://docs.sglang.ai/references/faq.html

Method	Model	ACE05-E		ACE05-E+		ERE	
		Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
DEGREE (SFT)	Bart-b	73.5	69.0	72.0	67.9	75.6	70.0
DEGREE (SFT)	Bart-l	76.0	73.5	75.2	73.0	76.2	73.2
BART (SFT)	Bart-b	64.1	59.6	65.6	59.2	68.7	63.2
BART (SFT)	Bart-l	63.0	61.1	63.4	61.3	69.9	63.8
CODE4STRUCT (0-shot)	Qwen2.5-Coder-7B-Instruct	51.7	36.0	51.0	35.8	50.9	33.7
CODE4STRUCT (10-shot)	Qwen2.5-Coder-7B-Instruct	59.3	54.5	58.4	50.8	55.1	42.2
DeepSeek-V3 (0-shot)	DeepSeek-V3	70.7	51.2	69.6	49.6	57.2	43.0
GPT-4o (0-shot)	GPT-4o	70.4	53.7	67.5	51.7	56.8	45.5
DeepSeek-R1 (CoT)	DeepSeek-R1	**72.4**	54.9	**69.2**	52.9	57.3	45.6
Vanilia (0-shot)	Qwen2.5-7B-Instruct	49.0	33.8	47.5	32.1	27.6	19.5
CAT (0-shot)	Qwen2.5-7B-Instruct	66.2	**55.6**	63.1	**53.3**	**58.6**	**46.4**

Table 1: The overall performance of our CAT and baselines. We bold the best result and underline the second best.

to 128 new tokens, which is sufficient for all test set outputs.

4 Results and Analysis

To evaluate the performance of our CAT framework, we compare it with several strong baselines (Section 4.1). Then, we investigate the three preference traps in LLMs of different families and scales (Section 4.2). The ablation study demonstrates the effectiveness of each component in CAT (Section 4.3). Finally, we evaluate CAT's model adaptability (Section 4.4) and discuss the time cost (Appendix C).

4.1 Overall Performance

Table 1 presents the main results of all baselines and our CAT on three datasets. We observe that CAT achieves the highest Arg-C F1 score on every evaluation benchmark compared to all the unsupervised baselines. Surprisingly, CAT with a local 7B model in a zero-shot setting surpasses the latest DeepSeek-R1 API-accessible model, with relative improvements of +0.7%, +0.4%, and +0.8% in Arg-C F1 scores. It demonstrates that addressing preference traps significantly enhances model performance on EAE tasks. It also highlights CAT's ability to effectively stimulate the IE capabilities of smaller models, which is of great significance for reducing the temporal and spatial cost of LLM-based IE tasks.

4.2 Existence of Three Preference Traps

We investigate the three preference traps in EAE tasks using LLMs from different families (Mistral-v0.3 (Jiang et al., 2023), Llama-3.1 (Grattafiori et al., 2024), Qwen2.5 (Yang et al., 2024), GPT-4o (Hurst et al., 2024), and DeepSeek-V3 (Liu et al., 2024)) and of varying scales (1.5B, 7B, 14B, and 32B).Appendix B exhibits the prompts, formats, and results for each model. To ensure the reliability of the results, we conduct sampled evaluation with six random seeds [7, 14, 21, 28, 35, 42]. In this section, the average score and standard deviation of Qwen2.5-7B-Instruct are reported.

4.2.1 Prior Knowledge Trap

We compare the extraction results under definitions of different argument roles to demonstrate the existence of this trap. Specifically, under three prior knowledge settings—A's name, A's name + definition, and A's name + B's definition—we measure the LLM's F1 scores for extracting argument roles A or B in the same event type. We test 10 pairs of argument roles with 50 instances per pair. To minimize the influence of other factors, we ensure the names of A and B clearly convey their meanings, and the wording of the definitions remains consistent.

Figure 7 illustrates the Arg-C F1 score and its proportion on identification A or B. We find that the introduction of B's definition results in an improvement in F1 score on Task B (+8.2%), while a reduction on Task A (-25.4%). In other words, B's definition leads the LLM to extract more B arguments and fewer A arguments. Besides, A's definition effectively improves the LLM's performance on Task A (+6.5%). These results reflect the LLM's heavy reliance on prior knowledge, con-

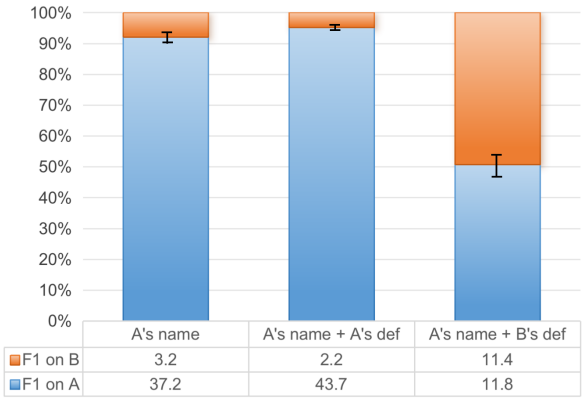

Figure 7: The LLM's performance on Identification A (blue) and Identification B (orange) under different prior knowledge.

firming the existence of this trap.

4.2.2 Sycophancy Hallucination Trap

We count the nonexistent or irrelevant arguments extracted by the LLM to confirm this trap. First, when an argument role has no matching argument in the text, we calculate the LLM's average extracted arguments per role. Then, when an argument role has no matching argument in the text and the text contains more than one event, we calculate the LLM's average extracted arguments belonging to other events per role. Finally, we employ CAT's countermeasures and compare the new results with the above ones.

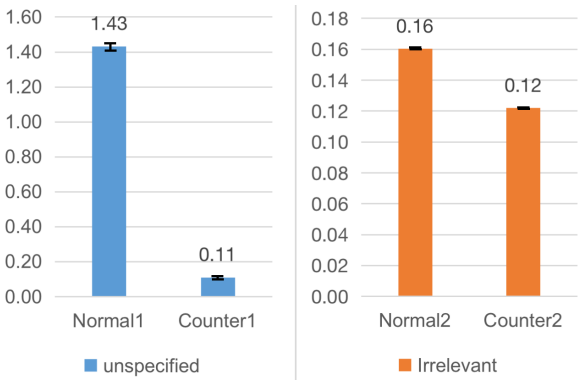

Figure 8: The average number of extracted arguments by the LLM under different settings: "Counter1" means the clear output specifications are provided and "Counter2" means the extraction scope are restricted. "Normal" denotes the bare LLM without the countermeasure.

Figure 8 presents all the results. We observe that without countermeasures, the LLM extracts 1.43 arguments unspecified and 0.16 arguments belonging to other events per role on average. However, after employing the countermeasures, the corresponding values decrease to 0.11 and 0.12, respectively. This demonstrates that LLMs tend to extract nonexistent or irrelevant arguments to meet users' extraction demands, namely, the sycophancy hallucination trap. CAT's countermeasures can effectively mitigate this issue.

4.2.3 Output Contradiction Trap

We compare LLM's performance with/without output contradiction to verify this trap. Specifically, we provide three output formats: raw output, JSON, and natural language. These formats are structurally similar, thereby avoiding the impact of particular formats on performance. The prompt and hard constraints each select an output format to control the LLM's generation; output contradiction arises when their chosen formats differ.

Figure 9 presents the F1 score and the log-probability gap between correct and incorrect answers under different format combinations. Higher F1 scores and larger gaps indicate better performance. We find that for each format, LLMs perform better without contradictions (AB = 11, 22, 33) than with them (except for some special cases). These results show that the output contradiction reduces the log-probability gap between correct and incorrect answers, thereby affecting model performance—confirming the trap's existence.

4.3 Ablation Study

In this section, we individually mask each method in CAT and compare the performance to study their effectiveness. Table 2 lists all the methods of CAT and their ablation experiment results.

Ablated Method	Arg-C F1
Refined definitions	48.2
Unbiased and direct expressions	52.0
Clear output specifications	47.8
Restricted extraction scope	50.2
The most preferred format	50.3
Inherent habits	48.4
Stage division	-
Choose arguments	42.2

Table 2: Ablation results of CAT's Methods. The bolded ones represent our original contributions. As a comparison, the complete CAT's Arg-C F1 is **55.6**.

From Table 2, we observe that masking each method leads to varying degrees of performance

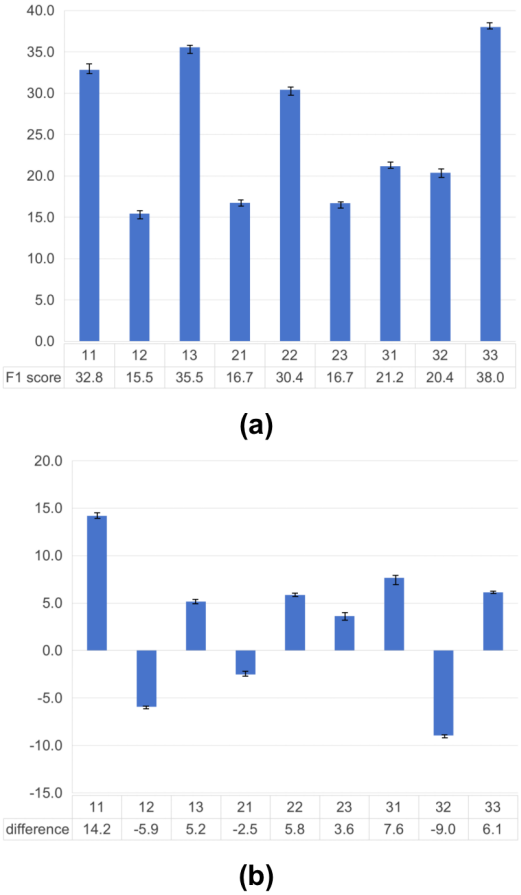

Figure 9: The LLM's performance on the EAE task under 9 combinations. (a) shows the F1 scores, (b) shows the log-probability gaps. Combinations are represented as "AB", for example, "12" indicates that we prompt the LLM to output the 1st format but enforce the 2nd one.

decline in CAT. Notably, when the choose stage reverts to the conventional token-level generation (Row 8), the F1 score sees the largest drop (-13.4%). Additionally, when CAT performs the EAE task in a single turn with the LLM (Row 7), the experiment can not complete within a normal timeframe due to the large size of the unfiltered candidate argument set. The above conclusions indicate that, first, each method employed in CAT contributes to its final performance. Second, by selecting answers from a candidate argument set to fill the forced-output templates, CAT effectively overcomes the drawbacks of conventional generation, leading to a significant improvement in performance.

4.4 Model Adaptability

We employ diverse types and sizes of LLMs as base model to evaluate CAT's model adaptability. Table 3 presents the experimental results. We observe that CAT consistently performs better than Vanilia, demonstrating its strong adaptability across different models.

CAT shows little performance variation when built upon Qwen2.5 models (7B+), as different models prefer distinct prompt details. CAT's current prompts and prior knowledge have reached their upper limit in stimulating the information extraction capabilities of these LLMs.

Model	Vanilia	CAT
Qwen2.5-1.5B-Instruct	19.5	41.4
Qwen2.5-7B-Instruct	33.8	55.6
Qwen2.5-14B-Instruct	40.8	52.5
Qwen2.5-32B-Instruct	46.3	54.0
Mistral-7B-Instruct-v0.3	22.1	48.4
Llama-3.1-8B-Instruct	26.9	50.9

Table 3: The performance (the Arg-C F1 score) of Vanilia and CAT.

5 Related works

Unsupervised event extraction: Recent advancements in unsupervised event extraction (EE) have focused on methods like prompt engineering, chain of thought, and programming language output. Prompt engineering mitigates the prior knowledge trap by refining event type definitions or optimizing prompts (Cai et al., 2024; Hong and Liu, 2024). The chain of thought approach reduces the sycophancy hallucination trap by breaking extraction into stages or reasoning (Wei et al., 2023; Ma et al., 2024; Guo et al., 2025). Programming language output leverages code LLMs to generate structured information (Wang et al., 2023; Guo et al., 2024b), addressing the output contradiction trap. While these methods address specific LLM preference traps, CAT innovatively tackles all three traps by dividing EE into think stage and choose stage, reducing abnormal token impacts and ensuring completely constrained outputs.

LLM Preferences and Their Impacts: LLM preferences significantly influence downstream task performance and AI reliability, making their study essential. Research explores inherent LLM preferences, such as understanding humor (Hessel et al., 2023), mental models of everyday objects (Gu et al., 2023), hidden biases (Bai et al., 2024), and human-like preference similarities (Li et al., 2024b). Studies also examine preference impacts

in downstream tasks, including evaluation tendencies (Panickssery et al., 2024), learning behaviors with conflicting knowledge (Li et al., 2024a), and output format preferences (Nguyen et al., 2025). To address these issues, preference optimization has emerged as a key focus. Methods include multi-modal feedback for long-range decision tasks (Zhao et al., 2024), and multi-objective alignment to mitigate the Matthew effect (Guo et al., 2024a; Zheng et al., 2024b). However, few works, like CAT, address LLM preferences without training, which is critical for advancing LLM capabilities in such scenarios.

6 Conclusion

In this work, we first experimentally investigate and summarize the preference traps in LLM-based unsupervised EAE tasks. We then propose CAT, a two-stage framework employing practical countermeasures and an innovative generation approach for LLMs. Experiments demonstrate that CAT (using a local 7B model in a zero-shot setting) matches the performance of the best DeepSeek-R1. Additionally, CAT exhibits low time cost and robust model adaptation capabilities in further experiments. Future work will extend this work to information extraction and control extraction boundary.

Limitations

First, since LLMs are built differently, our provided prompts may not be optimal for all models. Second, directly controlling extraction results via the candidate argument set needs further study. Third, some countermeasures do not establish a clear criterion for the prompt optimization process. This is partly due to LLM differences, preventing a universal standard. Additionally, prompt optimization is a heuristic process that relies on practical experimentation.

Ethics Statement

Event argument extraction (EAE) task is a well-defined task in Information Extract (IE) field. In our research and experimental process, our use of existing artifacts (e.g., datasets) was licensed and consistent with their intended use. We do not see other significant ethical concerns. Meanwhile, we keep honest in our work and our work will not be used to harm anyone.

Acknowledgements

This work was supported by National Key R&D Program of China (Grant No. 2024YFF0907400).

References

Xuechunzi Bai, Angelina Wang, Ilia Sucholutsky, and Thomas L Griffiths. 2024. Measuring implicit bias in explicitly unbiased large language models. *arXiv preprint arXiv:2402.04105*.

Zefan Cai, Po-Nien Kung, Ashima Suvarna, Mingyu Ma, Hritik Bansal, Baobao Chang, P. Jeffrey Brantingham, Wei Wang, and Nanyun Peng. 2024. Improving event definition following for zero-shot event detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2842–2863, Bangkok, Thailand. Association for Computational Linguistics.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yuling Gu, Bhavana Dalvi Mishra, and Peter Clark. 2023. Do language models have coherent mental models of everyday things? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1892–1913, Toronto, Canada. Association for Computational Linguistics.

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*.

Yiju Guo, Ganqu Cui, Lifan Yuan, Ning Ding, Zexu Sun, Bowen Sun, Huimin Chen, Ruobing Xie, Jie Zhou, Yankai Lin, et al. 2024a. Controllable preference optimization: Toward controllable multi-objective alignment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1437–1454.

Yucan Guo, Zixuan Li, Xiaolong Jin, Yantao Liu, Yutao Zeng, Wenxuan Liu, Xiang Li, Pan Yang, Long Bai, Jiafeng Guo, et al. 2024b. Retrieval-augmented code generation for universal information extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 30–42. Springer.

Jack Hessel, Ana Marasovic, Jena D. Hwang, Lillian Lee, Jeff Da, Rowan Zellers, Robert Mankoff, and Yejin Choi. 2023. Do androids laugh at electric sheep? humor "understanding" benchmarks from the new yorker caption contest. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–714, Toronto, Canada. Association for Computational Linguistics.

Zijin Hong and Jian Liu. 2024. Towards better question generation in QA-based event extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9025–9038, Bangkok, Thailand. Association for Computational Linguistics.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. DEGREE: A data-efficient generation-based event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States. Association for Computational Linguistics.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jiahuan Li, Yiqing Cao, Shujian Huang, and Jiajun Chen. 2024a. Formality is favored: Unraveling the learning preferences of large language models on data with conflicting knowledge. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5307–5320, Miami, Florida, USA. Association for Computational Linguistics.

Junlong Li, Fan Zhou, Shichao Sun, Yikai Zhang, Hai Zhao, and Pengfei Liu. 2024b. Dissecting human and LLM preferences. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1790–1811, Bangkok, Thailand. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7999–8009.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Mingyu Derek Ma, Xiaoxuan Wang, Po-Nien Kung, P Jeffrey Brantingham, Nanyun Peng, and Wei Wang. 2024. Star: Boosting low-resource information extraction by structure-to-text data generation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18751–18759.

Ngoc-Hai Nguyen, Tiviatis Sim, Hieu Dao, Shafiq Joty, Kenji Kawaguchi, Nancy Chen, Min-Yen Kan, et al. 2025. Llms are biased towards output formats! systematically evaluating and mitigating output format bias of llms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 299–330.

Arjun Panickssery, Samuel Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: Annotation of entities, relations, and events. In *Proceedings of the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Sijia Wang and Lifu Huang. 2024. Debate as optimization: Adaptive conformal prediction and diverse retrieval for event extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16422–16435, Miami, Florida, USA. Association for Computational Linguistics.

Xingyao Wang, Sha Li, and Heng Ji. 2023. Code4Struct: Code generation for few-shot event structure prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3640–3663, Toronto, Canada. Association for Computational Linguistics.

X Wei, X Cui, N Cheng, X Wang, X Zhang, S Huang, P Xie, J Xu, Y Chen, M Zhang, et al. 2023. Chatie: Zero-shot information extraction via chatting with chatgpt. arxiv. *arXiv preprint arXiv:2302.10205*.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Qi Zhao, Haotian Fu, Chen Sun, and George Konidaris. 2024. Epo: Hierarchical llm agents with environment preference optimization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6401–6415.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2024a. Sglang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, 37:62557–62583.

Yongsen Zheng, Ruilin Xu, Ziliang Chen, Guohua Wang, Mingjie Qian, Jinghui Qin, and Liang Lin. 2024b. Hycorec: Hypergraph-enhanced multi-preference learning for alleviating matthew effect in conversational recommendation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2537.

A The Prompts of CAT

The prompts used in CAT are shown in Figure 10. Information in earlier inputs are reused in later turns to reduce error propagation.

B Experiment Prompts and Results

We present the prompts used in the experiments of preference traps (Section 4.2), along with the performance of models from different families and of varying scales. DeepSeek-V3 and GPT-4o are API-accessible models. Detailed prompts and output formats are presented in Figure 11.

Table 4 presents the LLMs' performance in the prior knowledge trap experiment, which comfirms this trap exists across models. Furthermore, larger models exhibit a reduced impact from correct argument role definitions (A's definition) but an increased influence from incorrect argument role definitions (B's definition). A similar trend is observed when comparing API-accessible models with locally deployed models.

Table 5 presents the LLMs' performance in the sycophancy hallucination trap experiment, which comfirms this trap exists across models. we observe that larger models extract fewer arguments not explicitly specified in the sentence, but more irrelevant arguments (belonging to other events). Besides, the output specifications we provided for cases where arguments do not exist (Counter1) actually lead the API-accessible models to extract more unspecified arguments. This occurs because models prefer different prompt details.

Table 6 and Table 7 presents the LLMs' performance in the output contradiction trap experiment, which comfirms this trap exists across models. Additionally, when the prompt's required output format conflicts with the hard constraints, larger models experience more significant performance degradation.

C Time Cost

To compare CAT's time cost with other models, we randomly extract 50 event samples and test (s/sentence) without any efficiency optimization (eg. radix-cache). The experiment is repeated five times with different seeds. The base model used was Qwen2.5-7B-Instruct, deployed on a single NVIDIA A40 GPU. Experiment results are presented in Table 8. We find CAT achieves lower time cost. Moreover, it has potential for further time optimization.

The 1st turn

Input:

"A Cuban boat with four armed men landed on American shores, utterly undetected by the Coast Guard."
Some people or something moves from one place to another in the sentence. What does the word "landed" in this sentence indicate?

Output:

The word "landed" indicates that the Cuban boat carrying four armed men arrived or came ashore on American shores.

The 2nd turn

Input:

In this sentence "...", the word "landed" indicates that the Cuban boat carrying ...
If the sentence does not specify some information, just say "None explicitly mentioned".
Here is the information needed:
- Who promotes the movement
- Who or what moves
- ...

Output:

- Who promotes the movement: None explicitly mentioned
- Who or what moves: A Cuban boat carrying four armed men.
- ...

The 3rd turn (for each argument role)

Input:

In this sentence "...", the word "landed" indicates that the Cuban boat carrying ...
- Who or what moves: A Cuban boat carrying four armed men.
Based on the above information, answer my question about Artifact: Who or what moves?
You can only choose the best ones from the entities to answer. If it is not explicitly specified in the information, just say none.
The entities: A Cuban boat carrying four armed men, A Cuban boat, four armed men, ...

Output:

- Who or what moves: four armed men.

Figure 10: The prompts used in CAT. Each colored box corresponds to a different type of information.

Model	F1 on Task A			F1 on Task B		
	An	An + Ad	An + Bd	An	An + Ad	An + Bd
Qwen2.5-1.5B-Instruct	14.5	28.8	17.9	6.2	5.5	18.8
Qwen2.5-7B-Instruct	37.2	43.7	11.8	3.2	2.2	11.4
Qwen2.5-14B-Instruct	42.5	49.7	9.1	4.0	2.4	24.0
Qwen2.5-32B-Instruct	43.5	51.8	10.0	7.2	3.6	33.1
Mistral-7B-Instruct-v0.3	23.2	24.9	8.4	1.8	1.6	8.1
Llama-3.1-8B-Instruct	37.5	38.1	23.9	5.3	4.6	20.5
DeepSeek-V3	55.6	56.0	24.7	2.9	3.2	20.8
GPT-4o	57.4	58.1	22.5	2.8	3.0	21.8

Table 4: The LLMs' performance in the experiment of the prior knowledge trap. "An" denotes A's name. "An + Ad" denotes A's name and A's definition. "An + Bd" denotes A's name and B's definition.

Model	Unspecified		Irrelevant	
	Normal1	Counter1	Normal2	Counter2
Qwen2.5-1B-Instruct	2.24	0.43	0.15	0.10
Qwen2.5-7B-Instruct	1.43	0.11	0.16	0.12
Qwen2.5-14B-Instruct	1.84	0.27	0.19	0.13
Qwen2.5-32B-Instruct	0.72	0.05	0.18	0.13
Mistral-7B-Instruct-v0.3	1.11	0.33	0.20	0.10
Llama-3.1-8B-Instruct	1.55	1.09	0.29	0.19
DeepSeek-V3	0.66	0.78	0.23	0.18
GPT-4o	0.62	0.87	0.17	0.09

Table 5: The average number of extracted arguments in the experiment of the sycophancy hallucination trap. "Unspecified" denotes the arguments not specified in the text. "Irrelevant" denotes the arguments irrelevant to the target event. "Counter1" means the clear output specifications are provided and "Counter2" means the extraction scope are restricted. "Normal" denotes the bare LLM without the countermeasure.

Model	11	12	13	21	22	23	31	32	33
Qwen2.5-1B-Instruct	21.3	21.8	22.2	17.1	23.3	13.4	19.3	17.0	17.0
Qwen2.5-7B-Instruct	32.8	15.5	35.5	16.7	30.4	16.7	21.2	20.4	38.0
Qwen2.5-14B-Instruct	32.0	24.1	27.6	16.8	45.1	17.9	18.1	24.8	46.1
Qwen2.5-32B-Instruct	35.8	26.3	38.7	14.2	41.4	15.2	17.9	18.2	42.6
Mistral-7B-Instruct-v0.3	34.8	26.3	38.8	18.2	37.4	23.1	20.6	25.0	39.0
Llama-3.1-8B-Instruct	34.8	24.7	34.5	17.3	26.1	18.6	18.3	22.7	36.2

Table 6: The LLMs' performance (the F1 score) in the experiment of the output contradiction trap. Combinations are represented as "AB", for example, "12" indicates that we prompt the LLM to output the 1st format but enforce the 2nd one.

Model	11	12	13	21	22	23	31	32	33
Qwen2.5-1B-Instruct	3.8	-4.3	1.2	0.7	0.1	0.7	6.6	-6.5	3.1
Qwen2.5-7B-Instruct	14.2	-5.9	5.2	-2.5	5.8	3.6	7.6	-9.0	6.1
Qwen2.5-14B-Instruct	12.3	-7.7	9.1	-5.7	6.4	4.9	4.1	-11.5	12.3
Qwen2.5-32B-Instruct	18.5	-7.6	6.1	-2.5	7.7	3.3	6.6	-14.7	9.8
Mistral-7B-Instruct-v0.3	4.1	-3.6	4.3	0.0	2.0	3.8	2.9	-5.5	5.3
Llama-3.1-8B-Instruct	5.8	-6.5	4.0	-4.2	-4.3	2.3	-0.1	-7.1	5.0

Table 7: The LLMs' performance (the log-probability gap between correct and incorrect answers) in the experiment of the output contradiction trap. Combinations are represented as "AB", for example, "12" indicates that we prompt the LLM to output the 1st format but enforce the 2nd one.

The Prior Knowledge Trap

The prompt **with** argument role definition:

> {sentence}
> The word "{trigger}" in this sentence indicates that {event_definition}, which is a {event_type} event.
> Please extract "{role}" of this event from the sentence. **The "{role}" is defined as "{role_definition}".**
> If it is specified in the sentence, please output like "Answer: answer1, answer2, ...". If not, just output "Answer: none".

The prompt **without** argument role definition:

> {sentence}
> The word "{trigger}" in this sentence indicates that {event_definition}, which is a {event_type} event.
> Please extract "{role}" of this event from the sentence.
> If it is specified in the sentence, please output like "Answer: answer1, answer2, ...". If not, just output "Answer: none".

The Sycophancy Hallucination Trap

P1. The prompt **with** output specification:

> {sentence / event description}
> The word "{trigger}" in this sentence indicates that {event_definition}, which is a {event_type} event.
> **If the sentence does not specify some information, just say "None".**
> Here is the information needed: {role_definition}".
> Please output like "{role_asks_str}: your result splitted by comma" **or "{role_asks_str}: None"**

P2. The prompt **without** output specification:

> {sentence}
> The word "{trigger}" in this sentence indicates that {event_definition}, which is a {event_type} event.
> Here is the information needed:\n{role_definition}".
> Please output like "{role_definition}: your result splitted by comma"

P3. The prompt to get the **event description**:

> "{sentence}"
> {event_definition} in the sentence. **What does the word "{trigger}" in this sentence indicate?**

Normal1 = P2, Counter1 = 1, Normal2 = P1, Counter2 = P3+P1

The Output Contradiction Trap

> {sentence}
> The word "{trigger}" in this sentence indicates that {event_explain}, which is a {etype} event.
> Please extract the {role} of the {etype} event from this sentence. The definition of {role} is: {role_explain}
> Please output in the following format: {**format-Existing**}. If the answer is not explicitly specified in the information, just output {**format-Not Existing**}.

Three output formats:

Format Name	Existing	Not Existing
Raw Output (1)	arg1, arg2, ...	none.
JSON (2)	["arg1", "arg2", ...]	[]
Natural Language (3)	Answer: arg1, arg2...	Answer: none.

Figure 11: The prompts used in experiments of preference traps.

Method	Time cost (s/sentence)
CAT	5.06 (STD:0.216)
GPT-4o	10.18 (STD:1.90)
DeepSeek-R1	64.33 (STD:5.71)
CoT	> 3 (Xu et al., 2025)
Multi-Agent	> 10 (Wang and Huang, 2024)

Table 8: The time cost of different methods. The underlined results are cited from prior works.

Dataset	Split	#Sents	#Entities	#Args
ACE05-E	Train	17,172	20,006	4,859
	Dev	923	2,451	605
	Test	823	3,017	576
ACE05-E$^+$	Train	19,216	47,554	6,607
	Dev	901	3,423	759
	Test	676	3,673	689
ERE	Train	8,886	22,831	4,372
	Dev	720	1,949	378
	Test	604	1,621	257

Table 9: Statistics of datasets