REAR: Reinforced Reasoning Optimization for Event Argument Extraction with Relation-Aware Support

Jianwen Luo, Yu Hong*, Shuai Yang, Jianmin Yao
School of Computer Science and Technology, Soochow University, Suzhou, China
{jwluo.ai, tianxianer}@gmail.com

Abstract

Event argument extraction aims to identify event arguments and classify their roles within events, whereas relation extraction classifies semantic relationships between entities. Existing methods typically design task-specific models for EAE, which restricts the integration of relation-level semantics. Consequently, they overlook the complementary cues from RE that are beneficial for argument role disambiguation. To overcome this limitation, we propose **REAR**, a Relation-aware EAE Reinforced optimization framework. REAR first conducts joint supervised optimization on reasoning-enhanced data, which serves as a warm-up to strengthen the Large Language Model (LLM)'s ability to perform EAE while incorporating auxiliary cues from RE. Subsequently, it applies reinforcement learning to explore diverse reasoning trajectories and derive near-optimal strategies for integrating relation-level signals into EAE. Experiments on the ACE-E, ACE-E$^+$ and ERE benchmarks demonstrate that REAR consistently surpasses previous decoder-only LLM methods, achieving F1-score gains of at least 0.9%, 2.2% and 1.6%, respectively. [1]

1 Introduction

Event Argument Extraction (EAE) is a crucial task in Information Extraction (IE), aimed at identifying and classifying arguments into predefined roles within event contexts (Yang et al., 2024). Meanwhile, Relation Extraction (RE) seeks to classify semantic relationships between pairs of entities (Zhao et al., 2024). Figure 1 provides an illustration of these two tasks. Traditionally, EAE and RE have been addressed independently. Existing approaches (Yang et al., 2024; Wang and Huang, 2024) design separate models for EAE without explicitly incorporating relation-level semantics, thereby overlooking the complementary cues that RE can provide for argument role disambiguation. This isolation often leads to incomplete contextual understanding, especially in cases where entity interactions are crucial for accurate argument classification. Indeed, RE can provide valuable auxiliary information for EAE. For instance, classifying the semantic relations between entities can offer cues for disambiguating their respective event roles.

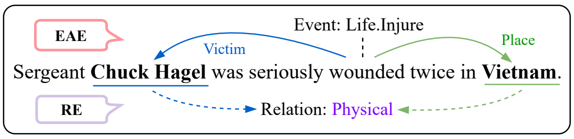

EAE
Event: Life.Injure
Victim
Place
Sergeant **Chuck Hagel** was seriously wounded twice in **Vietnam**.
RE
Relation: Physical

Figure 1: An illustrative example of EAE and RE. EAE aims to classify “Chuck Hagel” and “Vietnam” into roles of *Victim* and *Place*, respectively. RE seeks to classify the relation between these two entities as *Physical*.

Despite the auxiliary benefits of RE, incorporating it to enhance EAE via multi-task Supervised Fine-Tuning (SFT) remains challenging. SFT suffers from the inability to fully exploit the semantic cues provided by RE, as it relies on limited annotated reasoning paths (Wang et al., 2023a; Trung et al., 2024). This limitation hinders the model's ability to capture the subtle interactions between event arguments and relational information, thereby restricting the performance gains for EAE. In contrast, Reinforcement Learning (RL) offers a promising alternative. Not only does RL enable dynamic exploration of diverse reasoning trajectories, but it also allows iterative trial-and-error to autonomously discover near-optimal solutions (Jaech et al., 2024; DeepSeek-AI, 2025). Through this process, RL can effectively integrate cross-task semantic reasoning to leverage RE signals more comprehensively to strengthen EAE.

Building on these observations, we propose **REAR**, a two-stage Relation-aware EAE Reinforced optimization framework designed to

---

[*] Corresponding author

[1] The source code is publicly available at `https://github.com/jwluo-ai/REAR`

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 7957–7972
November 4-9, 2025 ©2025 Association for Computational Linguistics

address the limitations of conventional SFT. In the first stage, REAR leverages reasoning-enhanced data to perform joint supervised optimization over EAE and RE, providing the Large Language Model (LLM) with a warm-up that facilitates reasoning. In the second stage, RL-based optimization is applied to transcend the constraints of static annotated reasoning paths. The LLM dynamically explores diverse reasoning trajectories, engages in iterative trial-and-error to refine its decision-making process, and discovers near-optimal strategies for integrating relation-level semantics into EAE.

We experimentally validate REAR on the ACE-E, ACE-E$^+$ and ERE benchmark datasets (Doddington et al., 2004; Song et al., 2015). REAR substantially outperforms previous decoder-only LLM methods on these benchmarks. These results highlight the advantage of integrating relation-aware semantics with dynamic reasoning trajectories and underscore the effectiveness of REAR in modeling cross-task interactions between EAE and RE tasks.

2 Approach

In this section, we first formalize the EAE and RE tasks for reproducibility (§2.1). Subsequently, we detail our two-stage optimization strategy illustrated in Figure 2, joint supervised fine-tuning (§2.2) and joint reinforced fine-tuning (§2.3).

2.1 Task Definition

EAE aims to identify and classify arguments in a sentence associated with a specified event. Formally, given an input comprising the sentence, a set of entities, the event type, the trigger word and corresponding roles, the goal of the LLM is to output a sequence of arguments, each explicitly associated with a particular role. **RE** aims to classify the predefined relation type between a pair of arguments within a sentence. Detailed prompt templates for both tasks are provided in Appendices D and E. Note that if a sentence contains multiple events or relations, each is processed as a separate input.

2.2 Joint Supervised Warm-up

As depicted in Figure 2, the joint supervised warm-up stage leverages a combined training set from both EAE and RE tasks, forming tuples of $(\boldsymbol{x}, \boldsymbol{e}, \boldsymbol{y})$, where $\boldsymbol{x}$ denotes the input question derived from datasets, $\boldsymbol{e}$ represents the reasoning explanation automatically generated via the DeepSeek-R1 API[2] and $\boldsymbol{y}$ is the corresponding ground-truth answer.

To explicitly capture the joint nature of EAE and RE, we combine their reasoning-annotated examples into a unified training pool[3], training a single LLM policy model $\boldsymbol{\pi_\theta}$. The serialized tokens generative process of $\boldsymbol{\pi_\theta}$ is defined as a trajectory τ comprising a sequence of states and actions:

$$\tau = (s_0, a_0, s_1, a_1, \ldots, a_{T-1}, s_T, a_T) \quad (1)$$

where each action a_t is a token generated by $\boldsymbol{\pi_\theta}$ at timestep t, and the state s_t includes the initial input along with previously generated tokens. The sequence generation terminates upon producing a special token $a_T = \langle | end\_of\_sentence | \rangle$.

The joint supervised training objective combines EAE and RE training instances into one unified token-level cross-entropy loss, computed over trajectories sampled from combined training pool $\mathcal{P}$:

$$\mathcal{L}_{SFT}(\boldsymbol{\theta}) = -\mathbb{E}_{\tau\sim\mathcal{P}}\left[\sum_{t=0}^{T} \log\left(\boldsymbol{\pi_\theta}(a_t|s_t)\right)\right] \quad (2)$$

2.3 Joint Reinforced Optimization

Following the warm-up stage, we further optimize the joint model via RL based on the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017). Figure 2 (bottom pipeline) visualizes this RL process. Questions sampled via on-policy sampling from the combined task pool are reprocessed to generate updated reasoning explanations ($\boldsymbol{e}'$) and predictions ($\boldsymbol{y}'$). These outputs are compared to ground-truth answers ($\boldsymbol{y}$) to obtain rewards, driving the joint LLM policy optimization.

We explicitly model joint task rewards to reflect improvements across both tasks simultaneously. Specifically, to address the issue of reward sparsity(Riedmiller et al., 2018; Trott et al., 2019; Trung et al., 2024), we define sparse, terminal-state rewards for each task. For the EAE task, the reward function $\mathcal{R}_{\text{EAE}}$ is formulated as:

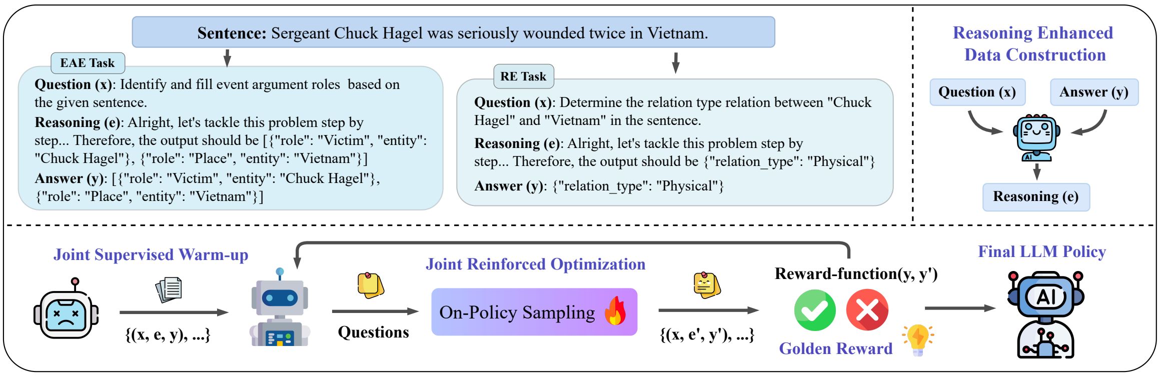
$$\mathcal{R}_{EAE}(\boldsymbol{y}, \boldsymbol{y}') = \begin{cases} \kappa, & \text{if } |\boldsymbol{y}| = 0 \wedge |\boldsymbol{y}'| = 0 \\ \mathcal{F}(\boldsymbol{y}, \boldsymbol{y}'), & \text{if } |\boldsymbol{y}| > 0 \\ 0, & \text{on exception} \end{cases} \quad (3)$$

where exceptions occur when the LLM produces structurally invalid outputs causing parsing errors. The scaling factor κ balances EAE and RE rewards, experimentally set to 2 based on validation results

---

[2] `https://platform.deepseek.com`

[3] Note that we utilize all EAE instances from the training set, whereas RE instances are included only if both arguments involved serve as event arguments within the same sentence.


Sentence: Sergeant Chuck Hagel was seriously wounded twice in Vietnam.

Question (x): Identify and fill event argument roles based on the given sentence.

Reasoning (e): Alright, let's tackle this problem step by step... Therefore, the output should be [{"role": "Victim", "entity": "Chuck Hagel"}, {"role": "Place", "entity": "Vietnam"}]

Answer (y): [{"role": "Victim", "entity": "Chuck Hagel"}, {"role": "Place", "entity": "Vietnam"}]

Question (x): Determine the relation type relation between "Chuck Hagel" and "Vietnam" in the sentence.

Reasoning (e): Alright, let's tackle this problem step by step... Therefore, the output should be {"relation_type": "Physical"}

Answer (y): {"relation_type": "Physical"}

Figure 2: Overview of our REAR framework. Top-left illustrates EAE and RE task definitions. Top-right shows reasoning enhanced data construction via DeepSeek-R1. Bottom pipeline depicts the LLM policy model evolving from joint supervised warm-up through joint reinforced optimization into the final optimized policy.

on the ACE-E$^+$ development set (see Section 3.5). Moreover, function $\mathcal{F}(\boldsymbol{y}, \boldsymbol{y}')$ is computed as:

$$\mathcal{F}(\boldsymbol{y}, \boldsymbol{y}') = \max\left(\frac{|\boldsymbol{y} \cap \boldsymbol{y}'| \cdot \kappa}{\max(|\boldsymbol{y}|,\ |\boldsymbol{y}'|)}, \sigma - \mu \cdot \max(|\boldsymbol{y}|,\ |\boldsymbol{y}'|), 0.1\right) \quad (4)$$

where the term $(\frac{|\boldsymbol{y} \cap \boldsymbol{y}'|}{\max(|\boldsymbol{y}|, |\boldsymbol{y}'|)})$ measures the proportion of correctly extracted event arguments, ranging between 0 and 1. The term $\left(\sigma - \mu \cdot \max(|\boldsymbol{y}|,\ |\boldsymbol{y}'|)\right)$ penalizes overly verbose predictions, especially crucial when $|\boldsymbol{y}| = 0$. The hyperparameters σ and μ are empirically set to 0.4 and 0.1, respectively. (We validate $\mathcal{F}(\boldsymbol{y}, \boldsymbol{y}')$ in Section 3.4.2.)

Similarly, for the RE task, we define:

$$\mathcal{R}_{RE}(\boldsymbol{y}, \boldsymbol{y}') = \begin{cases} 1, & \text{if } \boldsymbol{y} = \boldsymbol{y}' \\ 0.1, & \text{if } \boldsymbol{y} \neq \boldsymbol{y}' \wedge \boldsymbol{y}' \neq \texttt{null} \\ 0, & \text{on exception} \end{cases} \quad (5)$$

where exceptions involve structurally invalid outputs or undefined relation predictions.

Following Zheng et al. (2023), the total reward integrates the task-specific rewards and a KL divergence (Kullback and Leibler, 1951) term between the current policy and the initial policy $\pi_{\boldsymbol{\theta}}^{(0)}$:

$$\begin{aligned} \mathcal{R}_{total}(\boldsymbol{y}, \boldsymbol{y}') = \mathcal{R}_{EAE}(\boldsymbol{y}, \boldsymbol{y}') + \mathcal{R}_{RE}(\boldsymbol{y}, \boldsymbol{y}') \\ -\beta \cdot KL\left(\boldsymbol{\pi}_{\boldsymbol{\theta}}(\cdot|s_t), \boldsymbol{\pi}_{\boldsymbol{\theta}}^{(0)}(\cdot|s_t)\right) \end{aligned} \quad (6)$$

where β is dynamically adjusted during training via an adaptive KL controller (Ziegler et al., 2019a).

We jointly optimize policy and value objectives through PPO. The policy and value loss is defined as (see detailed derivations in Appendix A):

$$\begin{aligned} \mathcal{L}_{policy}(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{e} \sim \boldsymbol{\pi}_{\theta_{\text{old}}}} \Bigg[\min\left(\frac{\boldsymbol{\pi}_{\boldsymbol{\theta}}(a_t|s_t)}{\boldsymbol{\pi}_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}_t, \right. \\ \left. \text{clip}\left(\frac{\boldsymbol{\pi}_{\boldsymbol{\theta}}(a_t|s_t)}{\boldsymbol{\pi}_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}_t\right) \Bigg] \end{aligned} \quad (7)$$

$$\begin{aligned} \mathcal{L}_{value}(\boldsymbol{\phi}) = \frac{1}{2}\, \mathbb{E}_{\boldsymbol{e} \sim \boldsymbol{\pi}_{\theta_{\text{old}}}} \Bigg[\max\Bigg(\|V_{\boldsymbol{\phi}}(s_t) - \hat{R}_t\|^2, \\ \|\text{clip}\left(\hat{R}_t - V_{\phi}(s_t), \hat{A}_t - \epsilon, \hat{A}_t + \epsilon\right)\|^2 \Bigg) \Bigg] \end{aligned} \quad (8)$$

where $\boldsymbol{\pi}_{\theta_{\text{old}}}$ is the previous LLM policy. Advantage $\hat{A}_t$ is computed via generalized advantage estimation (Schulman et al., 2018). $V_{\boldsymbol{\phi}}(s_t)$ denotes the estimated state value. Return estimate $\hat{R}_t$ is computed by $\hat{A}_t$ and $V_{\boldsymbol{\phi}}(s_t)$. ϵ is the clipping factor.

The joint RL loss combines these components:

$$\mathcal{L}_{\text{RL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{\text{policy}}(\boldsymbol{\theta}) + \alpha \cdot \mathcal{L}_{\text{value}}(\boldsymbol{\phi}) \quad (9)$$

where the scaling factor α is set to 0.1, following prior work (Dai et al., 2023; Trung et al., 2024).

3 Experimentation

3.1 Experimental Setup

We evaluate our approach on the ACE-E, ACE-E$^+$ (Doddington et al., 2004) and ERE (Song et al., 2015). Following prior works (Lin et al., 2020; Yang et al., 2024), we preprocess ERE and construct two standard variants ACE-E and ACE-E$^+$ from the ACE2005 corpus (Doddington et al., 2004). We report argument classification F1 score, where arguments must be correctly classified with accurately predicted roles. Additional experimental setup details, including detailed dataset descriptions, introductions of the LLMs and hyperparameters are provided in Appendix B.

3.2 Main Results and Analysis

Table 1 presents the overall performance of our REAR framework on three benchmark datasets.

Optimization Stages	ACE-E		ACE-E$^+$		ERE	
	EAE	RE	EAE	RE	EAE	RE
Original Model	42.3	50.8	41.0	58.4	55.9	71.8
Supervised Warm-up	61.4	75.8	62.0	76.2	59.8	74.0
Reinforced Optimization	**63.3**	**76.9**	**64.1**	**77.4**	**62.2**	**76.1**

Table 1: Main results on ACE-E, ACE-E$^+$ and ERE datasets. We report F1-score (%) for EAE and accuracy (%) for RE. We highlight the best result.

Method	Model	ACE-E	ACE-E$^+$	ERE
ChatGPT-14 (Li et al., 2023)	GPT-3.5	28.9	30.9	-
ChatIE (Wei et al., 2023)	GPT-3.5	29.5	-	-
G-PTLM (Lin et al., 2023)	GPT-J-6B	-	31.2	29.6
ChatGPT-IE (Han et al., 2023)	GPT-4	34.4	36.3	-
RLQG (Hong and Liu, 2024)	LLaMA-2-13B	41.5	-	
	Qwen-7B	43.2	-	-
Code4UIE (Guo et al., 2024)	Text-Davinci-002	57.0	-	-
Debate-EE (Wang and Huang, 2024)	LLaMA-3-8B + GPT-3.5	56.0	-	-
	Gemini-Pro + GPT-3.5	59.5	-	-
Code4Struct (Wang et al., 2023b)	Text-Davinci-003	60.4	-	-
Scented-EAE (Yang et al., 2024)	LLaMA3-8B	62.4	61.9	60.6
REAR (Ours)	LLaMA3-8B	**63.3**	**64.1**	**62.2**

Table 2: Comparison of REAR with recent baselines. All numbers represent F1-score (%) for EAE.

Compared with the original model (DeepSeek-R1-Distill-LLaMA-3B [4]), both the supervised warm-up and the reinforced optimization stages yield substantial improvements. Taking ACE-E$^+$ as an example, the supervised warm-up already brings remarkable gains, with EAE improving by +21.0 F1-score and RE by +17.8 accuracy points. The subsequent reinforced optimization further contributes +2.1 F1-score and +1.2 accuracy points, showing that RL continues to refine the model beyond the benefits of supervised learning.

A similar pattern is consistently observed on ACE-E and ERE, which confirms that the two-stage optimization strategy is effective across different datasets. The supervised warm-up already yields strong gains, confirming the benefit of integrating reasoning-enhanced supervision across EAE and RE. Further applying reinforced optimization leads to consistent performance boosts on all benchmarks, achieving the best F1-scores in both tasks. This indicates that reinforcement learning enables the LLM to explore diverse reasoning trajectories. In addition, it refines the decision-making process, allowing the LLM to leverage relation-aware signals for argument role disambiguation. Notably, RE benefits as an auxiliary task, with accuracy consistently boosted. Overall, the two-stage optimization demonstrates clear effectiveness.

Training Data	Optimization Stages	EAE	RE
-	Original Model	41.0	58.4
EAE-only	Supervised Warm-up	57.1 ↑	52.9 ↓
	Reinforced Optimization	62.9 ↑	52.1 ↓
RE-only	Supervised Warm-up	35.6 ↓	74.3 ↑
	Reinforced Optimization	35.1 ↓	76.2 ↑
Mixed Data	Supervised Warm-up	62.0 ↑	76.2 ↑
	Reinforced Optimization	**64.1** ↑	**77.4** ↑

Table 3: Ablation results with task-specific training data. The table demonstrates how different training data compositions (EAE-only, RE-only, and Mixed Data) affect model performance across two tasks.

3.3 Comparison with Recent Work

Table 2 presents the overall comparison of REAR against several recent baselines. Baseline methods based on decoder-only LLMs are described in Appendix C. REAR consistently achieves higher performance compared with existing baseline methods. Our approach achieves F1-score of 63.3%, 64.1% and 62.2% on the ACE-E, ACE-E$^+$ and ERE datasets, respectively. These results outperform prior methods based on decoder-only LLMs on the EAE task. We attribute these improvements to the joint optimization paradigm adopted by REAR. Our approach simultaneously optimizes EAE and RE tasks, capturing inter-task correlations. The integration of reasoning-enhanced explanations further enriches supervision, promoting deeper semantic understanding. Additionally, RL with carefully designed sparse terminal rewards guides the LLM towards accurate predictions.

3.4 Ablation Study

3.4.1 Task-specific Analysis

Table 3 shows ablation results with task-specific training data. We compare the original model, EAE-only training, RE-only training, and mixed-task training under two optimization stages. Results are reported as F1-score (%) for EAE and accuracy (%) for RE on the ACE-E$^+$ dataset. The arrows (↑ / ↓) denote performance changes relative to the original model. From the table, we observe that EAE-only training improves EAE but reduces RE, while RE-only training improves RE but reduces EAE. Mixed-task training improves both tasks, and reinforced optimization achieves the best overall performance.

We compare the performance of the original

---

[4] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B

Reward Variant	Precision	Recall	F1-score
Original (ours)	**61.4**	67.1	**64.1**
w/o Verbosity Penalty	60.8 ↓	**67.2** ↑	63.8 ↓
Strict Partial Reward	60.7 ↓	64.3 ↓	62.4 ↓
Fully Sparse Reward	60.8 ↓	59.4 ↓	60.1 ↓

Table 4: Ablation results of different reward variants.

model, single-task training (EAE-only or RE-only), and mixed-task training under different optimization stages. Results are reported as F1-score (%) for EAE and accuracy (%) for RE on the ACE-E$^+$ dataset. As shown in Table 3, single-task training fails to learn either task, while mixed-task training yields strong performance. The arrows (↑ / ↓) indicate whether the performance improves or declines compared with the original model baseline.

3.4.2 Reward Function Components Analysis

We conduct ablation experiments on the reward components in Section 2.3, specifically testing four variants:

- **Original (ours)**: retains proportional reward, verbosity penalty and minimum reward threshold.
- **w/o Verbosity Penalty**: removes the penalty term that discourages overly verbose predictions.
- **Strict Partial Reward**: grants full reward only when the predicted argument set is identical to the ground-truth set, any partial overlap receives only a minimal reward (0.1).
- **Fully Sparse Reward**: grants reward only when the entire prediction is completely correct. Any partially correct prediction receives zero reward for both EAE and RE.

From Table 4, removing the verbosity penalty (w/o Verbosity Penalty) slightly increases recall but decreases precision, as the LLM produces more arguments without penalty, introducing additional false positives. Conversely, stricter reward criteria (Strict Partial Reward and Fully Sparse Reward) reduce recall. This occurs because stricter rewards encourage conservative predictions, restricting the model to fewer high-confidence extractions and thus harming generalization.

3.5 Impact of Reward Scaling Factor

As defined in Section 2.3, the EAE reward introduces a scaling factor κ to balance EAE and RE signals. We evaluate $\kappa \in \{0.5, 1, 2, 4, 8\}$ on the development sets of ACE-E, ACE-E$^+$ and ERE (Figure 3). Results show a bell-shaped trend: performance peaks at $\kappa = 2$ on ACE-E (62.9%) and ACE-E$^+$ (63.9%), while ERE achieves its best F1-score at $\kappa = 4$ (62.4%). Too small κ weakens EAE, whereas excessively large κ reduces cross-task synergy. Therefore, we adopt $\kappa = 2$ for ACE-E/ACE-E$^+$ and $\kappa = 4$ for ERE in our final experiments.

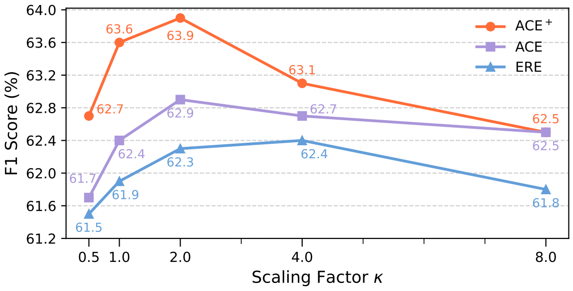


Figure 3: Effect of varying reward scaling factor κ.

4 Conclusion

In this paper, we propose a reinforced optimization framework designed to exploit semantic interactions between EAE and RE. Our approach enhances semantic reasoning through initial joint supervised training with reasoning explanations, followed by joint reinforced optimization to dynamically explore richer reasoning trajectories. In the future, we plan to explore the effectiveness with alternative model architectures, such as encoder-decoder and encoder-only models. Additionally, we aim to investigate RL methods that do not require large-scale reasoning annotated data, further enhancing the practical applicability of our REAR.

Limitations

Despite promising results, our approach exhibits several limitations. Firstly, our reinforced optimization framework requires careful tuning of hyperparameters, such as reward scaling factors and KL divergence constraints, which may limit generalization to new tasks. Moreover, the computational overhead introduced by RL exceeds traditional supervised optimization, potentially hindering scalability. Another limitation is that our approach currently considers pairwise relationships within individual sentences, neglecting more complex cross-sentence or document-level semantic interactions, leaving room for further exploration.

Acknowledgment

The research is supported by National Science Foundation of China (62376182).

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, et al. 2023. Gpt-4 technical report.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2023. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

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*.

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

Ridong Han, Chaohao Yang, Tao Peng, Prayag Tiwari, Xiang Wan, Lu Liu, and Benyou Wang. 2023. An empirical study on information extraction using large language models. *arXiv preprint arXiv:2305.14450*.

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.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. 2023. Evaluating chatgpt's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness. *arXiv preprint arXiv:2304.11633*.

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, Online. Association for Computational Linguistics.

Zizheng Lin, Hongming Zhang, and Yangqiu Song. 2023. Global constraints with prompting for zero-shot event argument classification. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2527–2538, Dubrovnik, Croatia. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. 2018. Learning by playing solving sparse reward tasks from scratch. In *Proceedings of ICML*.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-dimensional continuous control using generalized advantage estimation. *Preprint*, arXiv:1506.02438.

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

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, Denver, Colorado. Association for Computational Linguistics.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. 2019. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. In *Proceedings of NeurIPS*.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, Bangkok, Thailand. Association for Computational Linguistics.

Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. `https://github.com/kingoflolz/mesh-transformer-jax`.

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.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. 2023a. Instructuie: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Xingyao Wang, Sha Li, and Heng Ji. 2023b. 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.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. Chatie: Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.

Yu Yang, Jinyu Guo, Kai Shuang, and Chenrui Mao. 2024. Scented-EAE: Stage-customized entity type embedding for event argument extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5222–5235, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. 2024. A comprehensive survey on relation extraction: Recent advances and new frontiers. *ACM Computing Surveys*, 56(11):1–39.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. 2023. Secrets of rlhf in large language models part i: Ppo. *Preprint*, arXiv:2307.04964.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019a. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019b. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Derivation of RL Objectives

This section provides a detailed mathematical derivation of the reinforcement learning objectives employed in Section 2.3, specifically focusing on the Proximal Policy Optimization (PPO) framework (Schulman et al., 2017).

A.1 KL Divergence Regularization

To further stabilize training and prevent excessive divergence from the initial policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(0)}$, we introduce a KL-divergence penalty (Ziegler et al., 2019b) into the reward formulation:

$$\begin{aligned} \mathcal{R}_{total}(\boldsymbol{y}, \boldsymbol{y}') = \mathcal{R}_{EAE}(\boldsymbol{y}, \boldsymbol{y}') + \mathcal{R}_{RE}(\boldsymbol{y}, \boldsymbol{y}') \\ -\beta \cdot KL\left(\boldsymbol{\pi}_{\boldsymbol{\theta}}(\cdot|s_t), \boldsymbol{\pi}_{\boldsymbol{\theta}}^{(0)}(\cdot|s_t)\right) \end{aligned} \tag{10}$$

where β is dynamically adjusted by an adaptive KL controller (Ziegler et al., 2019a).

A.2 Optimization Objective

Given the total reward $\mathcal{R}_{\text{total}}$, the reinforcement learning objective aims to maximize the expected cumulative discounted reward:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim\boldsymbol{\pi}_{\boldsymbol{\theta}}}\left[\sum_{t=1}^{n} \gamma^{t-1} \cdot \mathcal{R}_{\text{total}}(s_t, a_t)\right] \tag{11}$$

where the $\tau = (s_0, a_0, s_1, a_1, \ldots, s_n, a_n)$ consists of state-action pairs sampled from policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. The trajectory length n equals the length of each generated output sequence. To improve the stability and accuracy of policy updates, we adopt the Generalized Advantage Estimation (GAE) approach (Schulman et al., 2018), which provides a balanced trade-off between bias and variance in estimating the advantage function.

A.3 Generalized Advantage Estimation

Following Schulman et al. (2018), the GAE advantage function $\hat{A}_t$ is computed as:

$$\hat{A}_t = \sum_{l=0}^{n-t} (\gamma \cdot \lambda)^l \cdot \delta_{t+l} \tag{12}$$

where $\gamma = 1$ and $\lambda = 0.95$ are hyperparameters. Each temporal difference term $\delta_{t'}$ is:

$$\begin{aligned} \delta_{t'} = &\mathcal{R}_{\text{total}}(s_{t'}, a_{t'}, s_{t'+1}) + \\ &\gamma \cdot V_{\boldsymbol{\phi}}(s_{t'+1}) - V_{\boldsymbol{\phi}}(s_{t'}) \end{aligned} \tag{13}$$

with terminal state value $V_{\boldsymbol{\phi}}(s_{n+1}) = 0$. Specifically, the value network V_ϕ is constructed by appending a linear value head to the last hidden states of the policy model obtained after the warm-up phase (Ziegler et al., 2019b; Trung et al., 2024).

The return estimate $\hat{R}_t$ is obtained by combining the advantage estimate and value predictions:

$$\hat{R}_t = \hat{A}_t + V_{\boldsymbol{\phi}}(s_t) \tag{14}$$

A.4 PPO Surrogate Objective

To ensure stable policy updates, PPO employs the clipped surrogate objective:

$$\begin{aligned} J_{\text{PPO}}(\theta) = \mathbb{E}_{\tau\sim\pi_{\theta_{\text{old}}}}\Big[\min\Big(r_t(\theta) \cdot \hat{A}_t, \\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big) \cdot \hat{A}_t\Big)\Big] \end{aligned} \tag{15}$$

where the probability ratio $r_t(\boldsymbol{\theta})$ is defined as:

$$r_t(\boldsymbol{\theta}) = \frac{\boldsymbol{\pi}_{\boldsymbol{\theta}}(a_t|s_t)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)} \tag{16}$$

We set the clipping parameter $\epsilon = 0.2$, following established practices (Schulman et al., 2017; Dai et al., 2023; Trung et al., 2024).

A.5 Value Function Approximation

We approximate expected returns using a value network $V_{\boldsymbol{\phi}}$, parameterized by $\boldsymbol{\phi}$ and optimize it with a clipped Mean Squared Error (MSE) objective:

$$\begin{aligned} \mathcal{L}_{\text{value}}(\phi) = \frac{1}{2}\mathbb{E}_{\tau\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}\big[\max\big((V_\phi(s_t) - \hat{R}_t)^2, \\ \text{clip}(V_\phi(s_t) - V_{\phi_{\text{old}}}(s_t), -\epsilon, \epsilon)^2\big)\big] \end{aligned} \tag{17}$$

where clipping ensures stable updates.

A.6 Combined RL Objectives

The full reinforcement learning objective combines both policy and value function objectives:

$$\mathcal{L}_{\text{RL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{\text{policy}}(\boldsymbol{\theta}) + \alpha \cdot \mathcal{L}_{\text{value}}(\boldsymbol{\phi}) \tag{18}$$

where the policy loss is defined as $\mathcal{L}_{\text{policy}}(\boldsymbol{\theta}) = -J_{\text{PPO}}(\boldsymbol{\theta})$ and the balancing hyperparameter is set to $\alpha = 0.1$, consistent with prior research (Dai et al., 2023; Trung et al., 2024).

B Experimental Setup Details

Table 5 details the hyperparameters employed during our two-stage optimization framework (Joint Supervised Warm-up and Joint Reinforced Optimization), as well as those for evaluation.

We conduct experiments on three dataset variants derived from the widely-used ACE2005 corpus (Doddington et al., 2004) and ERE (Song et al.,

Parameter	Value	Definition
Joint Supervised Warm-up		
$lora\_r$	32	Rank of the LoRA adapters during SFT
$lora\_\alpha$	64	Scaling factor controlling LoRA update magnitude in SFT
$lora\_target$	all	Modules targeted by LoRA adapters in SFT
lr	1.0e-5	Initial learning rate for SFT training
$batch\_size$	1	Number of examples per gradient update step in SFT
$gradient\_accum$	2	Steps of gradients accumulated before parameter updates in SFT
Joint Reinforced Optimization		
γ	1	Discount factor for future rewards in RL
λ	0.95	GAE parameter balancing bias and variance in RL advantage estimation
ϵ	0.2	PPO clipping parameter for policy update stability
α	0.1	Scaling factor balancing policy and value losses in RL
κ	2	Scaling factor for balancing EAE and RE task rewards
σ	0.4	Base reward used in EAE sparsity penalty calculation
μ	0.1	Penalty coefficient for overly verbose predictions in EAE
$lora\_r$	32	Rank of the LoRA adapters during RL optimization
$lora\_\alpha$	64	Scaling factor for LoRA updates in RL optimization
$lora\_target$	all	Modules targeted by LoRA adapters in RL optimization
lr	1.0e-5	Learning rate for RL optimization
$batch\_size$	25	Batch size for policy gradient updates during RL
$init\_kl\_coef$	1	Initial KL penalty coefficient in RL
$target$	1	Target KL divergence value for adaptive KL control in RL
$horizon$	1000	Horizon length for adaptive KL controller in RL
do_sample	True	Enable stochastic decoding during RL
$temperature$	0.6	Sampling temperature set during RL
top_p	0.95	Nucleus sampling threshold during RL
Model Evaluation		
do_sample	False	Deterministic decoding for evaluation
$temperature$	0	Temperature set for greedy inference

Table 5: The hyperparameters used in our experiment.

Dataset	Split	#Sents	#Entities	#Arguments
ACE-E	Train	17,172	20,006	4,895
	Dev	923	2,451	605
	Test	832	3,017	576
ACE-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	1,946	378
	Test	604	1,621	257

Table 6: Statistics of datasets

2015). ACE2005 contains 33 event types and 22 argument roles and ERE includes 38 event types and 21 argument roles. ACE2005 corpus and ERE contain comprehensive annotations for events, entities and relations, making it particularly suitable for sentence-level evaluations of both the EAE and RE tasks. Specifically, we follow the preprocessing procedures (Lin et al., 2020; Yang et al., 2024) from ACE2005 corpus to construct the two dataset variants: ACE-E and ACE-E$^+$. The detailed statistics for each dataset split are presented in Table 6.

Due to space constraints in Table 2, abbreviated model names were used. Here we provide the full model names and corresponding references:

- **GPT-3.5** refers to ChatGPT, GPT-3.5-Turbo and GPT-3.5-Turbo-Instruct models (Ouyang et al., 2022).
- **GPT-4** denotes the GPT-4 model introduced by OpenAI (Achiam et al., 2023).
- **GPT-J-6B** is the GPT-J model with 6 billion parameters (Wang, 2021).
- **Text-Davinci-002** and **Text-Davinci-003** are GPT-3 series instruct models optimized for instruction-following tasks (Brown et al., 2020).
- **Gemini-Pro** represents Google's Gemini-Pro model (Team et al., 2024).
- **LLaMA-2-13B** denotes Meta's LLaMA-2 model with 13 billion parameters (Touvron et al., 2023).
- **LLaMA-3-8B** primarily refers to the LLaMA-3-8B-Instruct model (Grattafiori et al., 2024) and DeepSeek-R1-Distill-Llama-8B model[5]. Additionally, in our experiments, we specifically use the DeepSeek-R1-Distill-Llama-8B as baseline.

C Related Work

Recent works explored leveraging LLMs for EAE.

---

[5] `https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B`

Li et al. (2023) evaluated ChatGPT in OpenIE, highlighting plausible yet poorly calibrated outputs under standard IE scenarios. Wei et al. (2023) proposed ChatIE, transforming zero-shot IE into multi-turn QA. To enhance zero-shot performance, Lin et al. (2023) introduced G-PTLM, employing global constraints and prompting strategies without annotated training data. Meanwhile, Han et al. (2023) systematically analyzed GPT-4 under various prompting settings, uncovering performance gaps with supervised methods.

Reinforcement learning based and retrieval-augmented methods have further advanced performance. Hong and Liu (2024) proposed RLQG, improving QA-based extraction via inverse prompting and question-quality rewards. Guo et al. (2024) introduced Code4UIE, formulating IE as code generation with Python-defined event schemas and retrieval-enhanced prompting.

Multi-agent systems like Debate-EE (Wang and Huang, 2024) optimized extraction via diverse-RAG and adaptive conformal prediction, surpassing single-agent methods. Similarly, Wang et al. (2023b) leveraged structured python outputs from code-capable LLMs, integrating external constraints for enhanced performance.

Recently, Yang et al. (2024) proposed Scented-EAE, explicitly modeling entity-type semantics via stage-customized embeddings. It strengthens semantic association at the input stage by augmenting entity-role correspondence, preserves semantic integrity using a BIO-aware joint learning mechanism at the intermediate stage.

D Detailed Illustration of Prompts

The prompt for the EAE task is illustrated in Figure 4. The structures for “argument_roles” and “template” associated with specific events are derived from the work of (Yang et al., 2024). Each event type is associated with distinct “argument_roles” and a unique “template”. To construct the complete EAE prompt, we need to follow the input structure shown in the “Examples” section of this EAE prompt and replace “[Insert Input Here]” with the integrated input information. An example of the input and corresponding output of the EAE task is presented in Appendix E.

Figure 5 presents the prompt designed for the RE task. The “Relation Type Definitions” section pre-defines 6 distinct types of relations, from which the LLM is expected to select the most appropriate one as the answer. To construct the complete RE prompt, we should follow the input format demonstrated in the “Examples” section of the prompt and replace “[Insert Input JSON Here]” and “[Insert Input TEXT Here]” with the corresponding integrated input data.

Figure 6 displays the prompt design for the Auto-Reasoning-EAE setting. In this framework, both the input and the corresponding output of the EAE task are provided to the LLM, which is then instructed to generate a reasoning style reasoning process that bridges the input and output. The prompt guides the model to follow a structured reasoning path by analyzing the event type, contextual clues, entity-role compatibility.

The Auto-Reasoning prompt designed for the RE task is shown as Figure 7. Both the input text and the corresponding relational output are given to the LLM, which is prompted to generate a concise reasoning that explains the reasoning process.

E EAE Input and Output Example

Figure 8 provides the complete prompt input to the LLM for the EAE task. Figure 9 presents the corresponding output produced by the LLM based on the input shown in Figure 8.

~~~
# Task Description
Task: Event Argument Extraction
Objective: Identify and fill event argument roles based on the given sentence, entity
mentions, event trigger, and argument schema.
Output: A JSON array of objects with keys `role` and `entity`. Omit roles with no matching
entities.
## Input Specifications
1. Sentence: A sentence containing an event.
2. Entity Mentions: Pre-annotated entities with types (e.g., `Person`, `Geo-Political
Entity`).
3. Event Trigger:
   - `word`: The trigger word (e.g., "wounded").
   - `event_type`: Event type (e.g., "Life.Injure").
4. Argument Schema:
   - `argument_roles`: Definitions for each role (e.g., `"Victim": "People or
Organization"`).
   - `template`: Event structure template (e.g., `"Agent injured Victim with Instrument at
Place."`).
## Output Requirements
1. Strict Matching: Use only entities from `entity_mentions`.
2. Multi-Argument Roles: A role may have multiple arguments (e.g., two "Attacker"
entries).
3. Output Format:
```json
[{"role": "[Role_name]", "entity": "[Entity]"}, ...]
```
## Examples
### Example 1:
#### Input:
```json
{
  "sentence": "Sergeant Chuck Hagel was seriously wounded twice in Vietnam.",
  "entity_mentions": [
    {"entity": "Sergeant", "entity_type": "Person"},
    {"entity": "Chuck Hagel", "entity_type": "Person"},
    {"entity": "Vietnam", "entity_type": "Geo-Political Entity"}
  ],
  "trigger": {"word": "wounded", "event_type": "Life.Injure"},
  "argument_roles": {
    "Victim": "People or Organization",
    "Agent": "People or Organization",
    "Place": "Location",
    "Instrument": "Vehicle or Facility"
  },
  "template": "Agent injured Victim with Instrument at Place."
}
```
#### Output:
```json
[{"role": "Victim", "entity": "Chuck Hagel"}, {"role": "Place", "entity": "Vietnam"}]
```
### Example 2:
#### Input:
```json
{
  "sentence": "But , for Israeli , Palestinian conflict there 's a lot of decisions that has
to be -- applied but until now -- she didn't put any pressure , toward and I believe this
policy has been this kind of policy, became more and more during Bush -- President Bush time
.",
  "entity_mentions": [
    {"entity": "Israeli", "entity_type": "Geo-Political Entity"},
    {"entity": "Palestinian", "entity_type": "Geo-Political Entity"},
    {"entity": "President", "entity_type": "Person"},
    {"entity": "Bush", "entity_type": "Person"}
  ],
  "trigger": {"word": "conflict", "event_type": "Conflict.Attack"},
  "argument_roles": {
    "Attacker": "People or Organization",
    "Place": "Location",
    "Target": "People or Organization",
    "Instrument": "Vehicle or Facility",
    "Victim": "People or Organization"
  },
  "template": "Attacker attacked Target hurting Victim using Instrument at Place."
}
```
#### Output:
```json
[{"role": "Attacker", "entity": "Israeli"}, {"role": "Attacker", "entity": "Palestinian"}]
```
## Your Task
### Analyze the following input and generate the output JSON:
#### Input:
```json
[Insert Input Here]
```
#### Output:
~~~

Figure 4: Illustration of the prompt for EAE task.

~~~
# Task Description
Task: Relation Extraction
Objective: Determine the relationship type between two given arguments with entity_type in a
sentence, based on predefined relation categories.
Output: A JSON object with the key `relation_type` indicating the relationship (use exactly one of
the 6 predefined types).
## Relation Type Definitions
1. Agent-Artifact: A living entity (agent) creates, uses, or interacts with an inanimate object or
artifact. For example, a person operating a vehicle or using a tool.
2. General-Affiliation: A broad association between entities that are not specifically
organizational, such as membership in a social club, informal network, or participation in a
community group.
3. Organization-Affiliation: A formal membership or employment relationship between a person/group
and an organization, such as being an employee of a company, a student at a university, or a member
of a government body.
4. Part-Whole: One entity is a component or subset of another. This includes relationships like a
"wheel" being part of a "car", or a "department" being a part of a "company".
5. Person-Social: Interpersonal relationships, including family ties, friendships, professional
collaborations, or other social connections between individuals.
6. Physical: Spatial or locational relationships, such as "located in," "found at," or "operates
within". For instance, a company being located in a specific city, or an event taking place at a
particular venue.
## Constraints
- Prioritize specific relationships (e.g., `Organization-Affiliation` over `General-
Affiliation`).
- Avoid inferring implicit relationships not explicitly supported by the sentence.
## Input Specifications
1. Sentence: A sentence containing both arguments.
2. Argument-1: The first argument and its entity_type to analyze.
3. Argument-2: The second argument and its entity_type to analyze.
## Output Requirements
1. Strict Matching: Use only the 6 predefined relation types.
2. Single Output: Must return exactly one `relation_type`.
3. Format:
```json
{"relation_type": "TYPE_NAME"}
```
## Example
### Input:
```json
{
  "sentence": "Sergeant Chuck Hagel was seriously wounded twice in Vietnam.",
  "Argument-1": {"text": "Chuck Hagel", "entity_type": "Person"},
  "Argument-2": {"text": "Vietnam", "entity_type": "Geo-Political Entity"}
}
```
Answer the relation between "Chuck Hagel" and "Vietnam" in the sentence. Only return the relation
type in JSON format.
### Output:
```json
{"relation_type": "Physical"}
```
## Your Task
Analyze the following input and generate the output JSON:
### Input:
```json
[Insert Input JSON Here]
```
[Insert Input TEXT Here]
### Output:
~~~

Figure 5: Illustration of the prompt for RE task.

~~~
# Task Description
Task: Reasoning Prompt Design:
Objective: Generate a unique thinking process for the given input, starting
with 'Alrighht, let’s break down this specific case.' Follow these principles:
1. Adapt to Input Nuances:
   - Begin by addressing the exact `event_type` (e.g., "Conflict.Attack") and
`template` (e.g., "Attacker attacked Target...").
   - Highlight how the sentence context shapes role assignments (e.g., "The
phrase ‘Israeli, Palestinian conflict’ implies mutual aggression").
2. Dynamic Entity-Role Alignment:
   - Explicitly reference entity names, their types/subtypes (e.g., "Israeli
(Geo-Political/Nation)"), and compare them to role definitions (e.g., "Attacker
requires People/Organization—matches Israeli’s subtype").
   - Reject mismatched entities with clear reasoning (e.g., "President is a
Person but not linked to the trigger’s context").
3. Trigger-Driven Logic:
   - Explain how the trigger’s position and modifiers (e.g., "‘conflict’ acts
as the central action") influence role extraction.
4. Output-Specific Justification:
   - Explicitly tie conclusions to the input’s unique data (e.g., "Only
Israeli/Palestinian fit Attacker roles here; Bush is unrelated to the event").
## Requirements:
- Start with a natural intro (e.g., "Alright, let’s break down...")
- Use bold/italics to emphasize key entities/roles.
- Avoid generic phrases—every step must reference the input’s specifics.
- Keep the chain between 150-300 words.
## Example Output Format:
```
Alright, let's tackle this problem step-by-step.
1. [Analysis step 1]
2. [Analysis step 2]
2. [Analysis step 3]
3. [Conclusion]
```
## Analyze the following input and the corresponding output to generate a
concise reasoning process.
### Input:
```json
[Insert Input Here]
```
### Corresponding Output:
```json
[Insert Output Here]
```
### Thought process:
~~~

Figure 6: Illustration of the Auto-reasoning prompt for EAE task, which guides the LLM to generate intermediate reasoning steps that explains how the output is derived from the input.

~~~
# Task Description
Task: Reasoning Prompt Design
Objective: Generate a task-specific reasoning chain that explains how to derive the output
from the input, not just the final relations.
## Relation Type Definitions
1. Agent-Artifact: A living entity (agent) creates, uses, or interacts with an inanimate
object or artifact. For example, a person operating a vehicle or using a tool.
2. General-Affiliation: A broad association between entities that are not specifically
organizational, such as membership in a social club, informal network, or participation in
a community group.
3. Organization-Affiliation: A formal membership or employment relationship between a
person/group and an organization, such as being an employee of a company, a student at a
university, or a member of a government body.
4. Part-Whole: One entity is a component or subset of another. This includes relationships
like a "wheel" being part of a "car", or a "department" being a part of a "company".
5. Person-Social: Interpersonal relationships, including family ties, friendships,
professional collaborations, or other social connections between individuals.
6. Physical: Spatial or locational relationships, such as "located in," "found at," or
"operates within". For instance, a company being located in a specific city, or an event
taking place at a particular venue.
## Follow these guidelines:
1. Context Anchoring:
   - Focus on the sentence structure (e.g., prepositions like "in," "at," or verbs like
"created," "joined").
   - Highlight contextual clues linking the two arguments (e.g., "‘wounded in Vietnam’
implies a location-based connection").
2. Relation-Type Prioritization:
   - Cross-check entity types (e.g., "Chuck Hagel (Person)" vs. "Vietnam (Geo-Political)")
against the 6 predefined relations.
   - Eliminate mismatches first (e.g., "Agent-Artifact requires an artifact; Vietnam is a
location, not an artifact").
3. Semantic Role Inference:
   - Analyze how the arguments interact:
     - Physical: Spatial prepositions ("in," "at"), locational verbs ("stationed,"
"operates").
     - Part-Whole: Terms like "part of," "belongs to," or structural hierarchies.
     - Organization-Affiliation: Job titles ("Sergeant"), employment verbs ("employed
by").
Requirements:
- Start with "Alright, let’s analyze..." and use bullet points.
- Keep the chain within 150-200 words.
## Analyze the following input and the corresponding output to generate a concise reasoning
process.
### Input:
```json
[Insert Input JSON Here]
```
### Corresponding Output:
```json
[Insert Output Here]
```
### Plese Generate the Thought process:
~~~

Figure 7: Illustration of the Auto-Reasoning prompt for RE task, which instructs the LLM to generate task-specific reasoning chains that explain how the output relation is derived from the input.

~~~
# Task Description
Task: Event Argument Extraction
Objective: Identify and fill event argument roles based on the given sentence, entity mentions, event
trigger, and argument schema.
Output: A JSON array of objects with keys `role` and `entity`. Omit roles with no matching entities.
## Input Specifications
1. Sentence: A sentence containing an event.
2. Entity Mentions: Pre-annotated entities with types (e.g., `Person`, `Geo-Political Entity`) and
sub_types (e.g., `Individual`, `Nation`).
3. Event Trigger:
 - `word`: The trigger word (e.g., "wounded").
 - `event_type`: Event type (e.g., "Life.Injure").
4. Argument Schema:
 - `argument_roles`: Definitions for each role (e.g., `"Victim": "People or Organization"`).
 - `template`: Event structure template (e.g., `"Agent injured Victim with Instrument at Place."`).
## Output Requirements
1. Strict Matching: Use only entities from `entity_mentions`.
2. Multi-Argument Roles: A role may have multiple arguments (e.g., two "Attacker" entries).
3. Output Format:
```json
[{"role": "[Role_name]", "entity": "[Entity]"}, ...]
```
## Examples
### Example 1:
#### Input:
```json
{"sentence": "Sergeant Chuck Hagel was seriously wounded twice in Vietnam.", "entity_mentions":
[{"entity": "Sergeant", "entity_type": "Person", "entity_subtype": "Individual"},{"entity": "Chuck Hagel",
"entity_type": "Person", "entity_subtype": "Individual"},{"entity": "Vietnam", "entity_type": "Geo-
Political Entity", "entity_subtype": "Nation"}],"trigger": {"word": "wounded", "event_type":
"Life.Injure"},"argument_roles": {"Victim": "People or Organization","Agent": "People or
Organization","Place": "Location","Instrument": "Vehicle or Facility"},"template": "Agent injured Victim
with Instrument at Place."}
```
#### Output:
```json
[{"role": "Victim", "entity": "Chuck Hagel"}, {"role": "Place", "entity": "Vietnam"}]
```
### Example 2:
#### Input:
```json
{"sentence": "But , for Israeli , Palestinian conflict there 's a lot of decisions that has to be --
applied but until now -- she didn't put any pressure , toward and I believe this policy has been this kind
of policy, became more and more during Bush -- President Bush time .","entity_mentions": [{"entity":
"Israeli", "entity_type": "Geo-Political Entity", "entity_subtype": "Nation"},{"entity": "Palestinian",
"entity_type": "Geo-Political Entity", "entity_subtype": "Nation"},{"entity": "President", "entity_type":
"Person", "entity_subtype": "Individual"},{"entity": "Bush", "entity_type": "Person", "entity_subtype":
"Individual"}],"trigger": {"word": "conflict", "event_type": "Conflict.Attack"},"argument_roles":
{"Attacker": "People or Organization","Place": "Location","Target": "People or Organization","Instrument":
"Vehicle or Facility","Victim": "People or Organization"},"template": "Attacker attacked Target hurting
Victim using Instrument at Place."}```
#### Output:
```json
[{"role": "Attacker", "entity": "Israeli"}, {"role": "Attacker", "entity": "Palestinian"}]
```

## Your Task
### Analyze the following input and generate the output JSON:
#### Input:
```json
{
    "sentence": "this is one technique, one possible way u.s. forces might seize a bridge, a bridge that
might be defended by iraqis.",
    "entity_mentions": [{"entity":"u.s","entity_type": "Geo-Political Entity","entity_subtype": "Nation"},
{"entity": "forces","entity_type": "Person",  "entity_subtype": "Group"},{"entity": "bridge""entity_type":
"Facility","entity_subtype": "Path"},{"entity": "bridge","entity_type": "Facility","entity_subtype":
"Path"},{"entity": "that","entity_type": "Facility","entity_subtype": "Path"},{"entity":
"iraqis","entity_type": "Person","entity_subtype": "Group"}],
    "trigger": {"word": "seize","event_type": "Transaction.Transfer-Ownership"},
    "argument_roles": {"Artifact": "Vehicle or Facility","Seller": "People or Organization","Beneficiary":
"People or Organization","Buyer": "People or Organization","Place": "Location"},
    "template": "Seller gave Buyer to Beneficiary for the benefit of Artifact at Place."
}
```
#### Output:
~~~

Figure 8: Illustration of the complete prompt input to the LLM for the EAE task.

~~~
Alright, let's tackle this problem step-by-step.

1. Event Type & Template Alignment: The event is Transaction.Transfer-
Ownership with the template "Seller gave Buyer to Beneficiary...". The
trigger "seize" implies forceful acquisition, shaping roles around ownership
transfer.
2. Artifact Identification: The phrase "seize a bridge" directly links
bridge (Facility/Path) to the Artifact role (requires Vehicle/Facility). The
duplicate "bridge" and "that" entries are redundant – only the first bridge
is valid.
3. Seller/Buyer Dynamics:
   - "u.s. forces" combines u.s (Geo-Political/Nation) and forces
(Person/Group). While "u.s" is a Nation, forces as a Group fits the Buyer
role (People/Organization) since they're seizing.
   - "defended by Iraqis" assigns iraqis (Person/Group) as the Seller – they
currently "own" the bridge through defense, matching the role’s
People/Organization requirement.
4. Excluding Irrelevant Roles:
   - Beneficiary and Place are omitted. No entity indicates a recipient
(Beneficiary) or specific location beyond the bridge itself.
   - The second "bridge" and "that" (Facility/Path) lack contextual ties to
the trigger’s action.

Conclusion: Only bridge (Artifact), forces (Buyer), and iraqis (Seller)
align with the template’s roles based on the trigger’s context and entity
subtypes. The U.S. as a Nation isn’t directly tied to the seizure action –
the focus is on the executing group (forces).
```. Therefore, the output should be [{"role": "Artifact","entity":
"bridge"},{"role": "Seller","entity": "iraqis"},{"role": "Buyer","entity":
"forces"}].

</think>

```json
[{"role": "Artifact","entity": "bridge"},{"role": "Seller","entity":
"iraqis"},{"role": "Buyer","entity": "forces"}]
~~~
```

Figure 9: The LLM’s output corresponding to the input in Figure 8. The content before the </think> tag represents the model’s reasoning reasoning process, while the content after the tag presents the extracted event arguments.
```