ProUIE: A Macro-to-Micro Progressive Learning Method for LLM-based Universal Information Extraction

Wenda Liu[1]*, **Zhigang Song**[1]*, **Shuai Nie**[1]†, **Guangyao Liu**[1], **Lisung Chen**[1]
Binyu Yang[1], **Yaran Chen**[2], **Peng Zhou**[1], **Hongzhen Wang**[1], **Yuchen Liu**[1]
Wenyue Hu[1], **Jiaming Xu**[1], **Runyu Shi**[1], **Ying Huang**[1]
[1]Xiaomi Corporation, Beijing, China
[2]Xi'an Jiaotong-Liverpool University, Suzhou, China
{liuwenda3, songzhigang, nieshuai}@xiaomi.com

Abstract

LLM-based universal information extraction (UIE) methods often rely on additional information beyond the original training data, which increases training complexity yet often yields limited gains. To address this, we propose ProUIE, a Macro-to-Micro progressive learning approach that improves UIE without introducing any external information. ProUIE consists of three stages: (i) macro-level Complete Modeling (CM), which learns NER, RE, and EE along their intrinsic difficulty order on the full training data to build a unified extraction foundation, (ii) meso-level Streamlined Alignment (SA), which operates on sampled data with simplified target formats, streamlining and regularizing structured outputs to make them more concise and controllable, and (iii) micro-level Deep Exploration (DE), which applies GRPO with stepwise fine-grained rewards (SFR) over structural units to guide exploration and improve performance. Experiments on 36 public datasets show that ProUIE consistently improves unified extraction, outperforming strong instruction-tuned baselines on average for NER and RE while using a smaller backbone, and it further demonstrates clear gains in production-oriented information extraction.

1 Introduction

Universal information extraction (UIE) has emerged as a prominent direction in both academia and industry, aiming to unify diverse information extraction tasks under a unified framework. With the rapid progress of large language models (LLMs), numerous works on information extraction have adopted structured outputs, instruction tuning, and schema-driven formulations, moving beyond task-specific pipelines (Pang et al., 2023; Wei et al., 2024; Wan et al., 2023; Li et al., 2025) toward more general and flexible paradigms. However, many LLM-based UIE methods still rely on additional information beyond the original training data (Lu et al., 2022; Wen et al., 2025; Guo et al., 2023; Liao et al., 2025; Wang et al., 2023), such as extra schema cues, external resources, or complex alignment and verification pipelines, thereby increasing training complexity while often yielding limited performance gains.

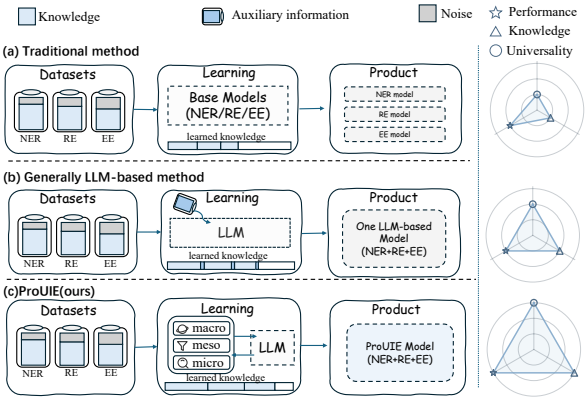

Figure 1: Comparison of UIE methods. (a) Traditional single-task methods train separate models for different extraction targets. (b) Generally LLM-based UIE unifies tasks but often introduces auxiliary signals beyond standard annotations. (c) ProUIE improves UIE without introducing any external information through a Macro-to-Micro progressive learning pipeline.

As illustrated in Figure 1, existing UIE methods can be broadly grouped into two paradigms: (a) Traditional single-task methods train and deploy separate models for different IE tasks, resulting in a straightforward pipeline but limited universality, and (b) Generally LLM-based methods unify multiple targets into a single LLM via structured generation and instruction-based interfaces. Representative systems include schema-driven generation frameworks (Lu et al., 2022), latent structure-aware generative modeling (Fei et al., 2023), multi-task instruction tuning for unified extraction (Wang

* Equal contribution.
† Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 21737–21750

et al., 2023; Xiao et al., 2024), guideline-based supervision to enhance zero-shot extraction (Sainz et al., 2024), and schema/knowledge coding with additional training stages (Li et al., 2024). In addition, targeted distillation from large proprietary LLMs has been used to obtain broad NER ability under limited direct supervision (Zhou et al., 2024). While these methods significantly advance UIE, many of them share a common design pattern: they introduce additional information beyond the original training data to stabilize learning and improve performance, thereby making the training pipeline heavier and less streamlined.

This naturally raises a key question: *Is it possible to improve LLM-based UIE without introducing any external information?* In this work, we answer this in the affirmative by rethinking the learning process itself: (i) how to progressively build a strong unified extraction foundation, (ii) how to make structured outputs concise and controllable with minimal overhead, and (iii) how to more fully exploit the model's capacity through fine-grained optimization objectives. This motivates a training framework that strengthens UIE while keeping the supervision source unchanged and the pipeline simple.

To address this, we propose **ProUIE**, a macro-to-micro progressive learning framework that improves UIE without introducing any external information. ProUIE organizes training into three increasingly fine-grained stages: macro-level Complete Modeling (CM), meso-level Streamlined Alignment (SA), and micro-level Deep Exploration (DE).

Specifically, the macro-level CM stage adopts supervised fine-tuning on the full training data using only standard annotations, serving as the foundation for unified extraction. Rather than relying on external information to constrain learning, CM aims to fully exploit the supervision contained in the original training data and establish a stable starting point for subsequent refinement. Built on CM, the meso-level SA stage further fine-tunes the model with a focus on the more complex RE/EE tasks, where structured generations tend to be verbose, inconsistent, or redundant. Concretely, SA randomly samples a subset of RE/EE training instances for focused refinement and trains the model to streamline outputs and remove redundancy, making the structured outputs more concise and controllable. Importantly, SA does not introduce any external resources or extra supervision; it improves controllability by tightening the output structure while staying within the standard-annotation regime. Finally, the micro-level DE stage targets structural errors that are difficult to resolve with conventional supervised learning. DE applies GRPO (Shao et al., 2024) together with our UIE-specific reward design, Stepwise Fine-grained Reward (SFR), which provides coarse-to-fine supervision over structural units to guide exploration. The key idea of SFR is to decompose structural correctness into progressively finer units and assign rewards across levels: for NER, SFR defines stepwise rewards at the entity-type and entity-value levels; for RE, at the relation-type, relation-instance, and entity-pair levels; and for EE, at the event-type, trigger, role, and argument levels. By shaping rewards from coarse to fine, DE can directly correct structural errors while still using only the original training data.

Our contributions are summarized as follows:

- We propose **ProUIE**, a macro-to-micro progressive learning method that improves UIE without introducing any external information, and organizes training into three stages: **CM**, **SA**, and **DE**.
- We design **SFR**, a UIE-specific reward mechanism that provides stepwise fine-grained rewards over structural units, and integrate it with GRPO in the DE stage to guide exploration and further improve extraction quality.
- Extensive experiments on 36 public datasets demonstrate that ProUIE consistently outperforms strong baselines, and further delivers clear gains in production-oriented information extraction.

2 Method

2.1 Problem Definition

Universal Information Extraction (UIE) aims to extract structured information from an input text x under a unified generation interface, covering three tasks: named entity recognition (NER), relation extraction (RE), and event extraction (EE). Given standard training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where y_i is the task-specific structured annotation, our goal is to learn a single LLM-based generator that produces structured outputs for all tasks *without using any external auxiliary information*.

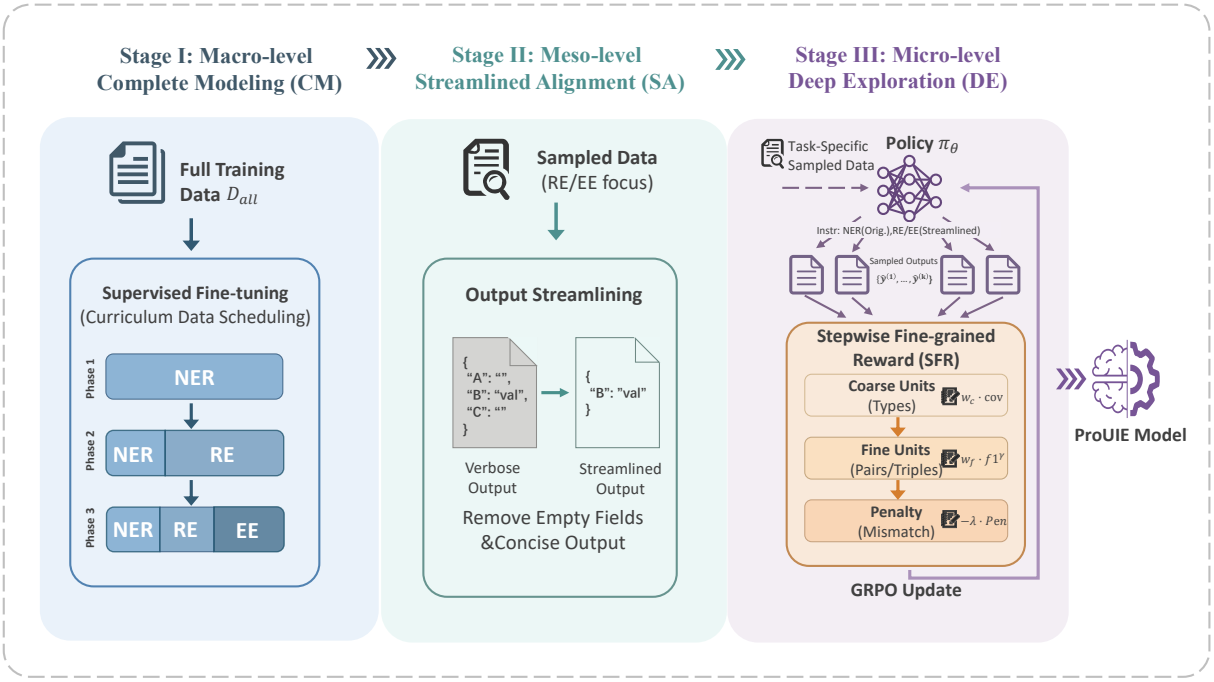

Figure 2: The framework of ProUIE.

2.2 Overview

We propose **ProUIE**, a macro-to-micro progressive learning framework with three stages: **Complete Modeling (CM)**, **Streamlined Alignment (SA)**, and **Deep Exploration (DE)**. CM builds a unified extraction foundation via task-ordered supervised training on full data. SA improves controllability by fine-tuning on sampled data with streamlined targets. DE further unlocks the model's capacity and boosts final performance via GRPO with our **Stepwise Fine-grained Reward (SFR)**. Figure 2 illustrates the framework of ProUIE.

2.3 Stage I: Complete Modeling (CM)

The Complete Modeling (CM) stage establishes a unified extraction foundation for ProUIE. At this macro level, we perform supervised fine-tuning on the full training data using only the original task annotations, without introducing any external signals or auxiliary supervision.

CM follows an intrinsic task-difficulty progression from NER to RE and EE, while maintaining unified generation behavior through mixed-task training. Specifically, CM is organized into three phases. In the first phase, the model is trained solely on NER data to acquire basic entity-level extraction capability. In the second phase, NER and RE data are jointly used, with a sampling ratio of 2:8, enabling the model to extend from entity recognition to relation modeling while retaining stable NER performance. In the third phase, NER, RE, and EE data are trained together with a ratio of 3:3:4, allowing the model to internalize increasingly complex extraction structures under a unified interface.

Rather than relying on external constraints, this phased design encourages the model to gradually absorb diverse extraction patterns while fully exploiting the supervision contained in the original training data. As the first stage of ProUIE, CM provides a stable and general extraction starting point for subsequent stages that focus on output alignment and fine-grained optimization. We provide the CM prompt templates in Appendix A.2.

2.4 Stage II: Streamlined Alignment (SA)

Built on CM, the meso-level **Streamlined Alignment (SA)** stage improves the *controllability* of structured generation, especially for the more complex RE/EE tasks where outputs are often verbose. SA does *not* introduce any external resources. Instead, it tightens the *target format* while staying strictly within the original annotation regime.

Formally, SA fine-tunes the model on a randomly sampled subset $\mathcal{D}_{\text{SA}} \subset \mathcal{D}$ that primarily consists of RE/EE instances. For each training example, we transform the original structured target into a *streamlined* target by removing redundant fields and standardizing the representation. For RE, we further enforce concise outputs by omitting empty

relation fields. For EE, we remove empty event-type entries, reducing invalid types and improving the conciseness and controllability of the outputs. This encourages the model to produce concise, stable, and controllable outputs, which also provides a cleaner structural basis for the subsequent micro-level optimization in DE. We provide the SA prompt templates in Appendix A.2.

2.5 Stage III: Deep Exploration (DE)

2.5.1 GRPO with task-ordered phases

After CM and SA, remaining errors are largely *structural*: correct high-level types may be paired with wrong or missing slots, and redundant units may be generated. DE addresses this by applying GRPO (Shao et al., 2024) and rewarding sampled generations according to structural correctness.

To respect intrinsic task difficulty and stabilize optimization, DE proceeds in three GRPO phases:

$$\mathrm{DE}_{\mathrm{NER}} \rightarrow \mathrm{DE}_{\mathrm{RE}} \rightarrow \mathrm{DE}_{\mathrm{EE}}. \tag{1}$$

In each phase, for every (x, y) we sample K candidates $\{\hat{y}^{(k)}\}_{k=1}^{K} \sim \pi_\theta(\cdot \mid x)$, compute rewards $\{R^{(k)}\}$, and update π_θ with GRPO using group-relative advantages. Optionally, the final reward can be clipped to $[0, 1]$.

2.5.2 Stepwise Fine-grained Reward (SFR)

SFR evaluates a structured output from coarse units to fine units. Coarse units encourage correct high-level structure (e.g., entity/relation/event types), while fine units focus on slot-level correctness (e.g., type–entity pairs, relation triples, event arguments). A lightweight penalty discourages over/under-generation.

From outputs to structural units. We convert both gold output y and prediction $\hat{y}$ into unit sets. For each task we use one coarse set and one (or a few) fine sets:

- **NER:** coarse T (entity types), fine P (type–entity pairs).
- **RE:** coarse T (relation types), fine R (type, head, tail triples), and supportive fine sets H (type, head) and A (type, tail).
- **EE:** coarse E (event types) and T (event type, trigger), fine correctness measured by aligned role–argument pairs within matched trigger groups.

Reward form. Let $\mathrm{cov}(\cdot)$ denote coverage on a coarse set, $\mathrm{f1}(\cdot)$ denote micro-F1 on a fine set, and $\mathrm{pen}(\cdot)$ denote a mismatch penalty. SFR follows a coarse-to-fine form:

$$R = w_c \,\mathrm{cov} + w_f \,(\mathrm{f1})^{\gamma} - \lambda \,\mathrm{pen}, \qquad \gamma > 0. \tag{2}$$

Definitions. Given gold set A_g and predicted set A_p, we define:

$$\mathrm{cov}(A_g, A_p) = \frac{|A_g \cap A_p|}{\max(1, |A_g|)}. \tag{3}$$

$$\mathrm{f1}(A_g, A_p) = \frac{2|A_g \cap A_p|}{|A_g| + |A_p|}. \tag{4}$$

$$\Delta(A_g, A_p) = \frac{|A_g \triangle A_p|}{\max(1, |A_g|)}. \tag{5}$$

$$D_{\mathrm{jac}}(A_g, A_p) = 1 - \frac{|A_g \cap A_p|}{\max(1, |A_g \cup A_p|)}. \tag{6}$$

Instantiations for NER/RE/EE. We now specify cov, f1, and the penalty term for each task.

NER. Coverage is computed on entity types, fine correctness is computed on type–entity pairs, penalties use normalized symmetric difference:

$$\begin{aligned} R_{\text{SFR-NER}} = {} & w_t \,\mathrm{cov}(T_g, T_p) + w_p \left(\mathrm{f1}(P_g, P_p)\right)^{\gamma_{\text{NER}}} \\ & - \lambda_t \,\Delta(T_g, T_p) - \lambda_p \,\Delta(P_g, P_p). \end{aligned} \tag{7}$$

RE. Triple correctness is the main signal, head and tail F1 provides additional dense feedback. We use Jaccard distance as a bounded mismatch penalty on types and triples:

$$\begin{aligned} R_{\text{SFR-RE}} = {} & w_t \,\mathrm{cov}(T_g, T_p) + w_h \,\mathrm{f1}(H_g, H_p) \\ & + w_a \,\mathrm{f1}(A_g, A_p) + w_r \left(\mathrm{f1}(R_g, R_p)\right)^{\gamma_{\text{RE}}} \\ & - \lambda_t \,D_{\mathrm{jac}}(T_g, T_p) - \lambda_r \,D_{\mathrm{jac}}(R_g, R_p). \end{aligned} \tag{8}$$

EE. We use event/trigger coverage as coarse signals. For fine-grained correctness, we compute an aligned argument-level score F_{full}: within each event type, each gold trigger group is matched to at most one predicted trigger group by maximum overlap of role–argument pairs, then we aggregate TP/FP/FN over role–argument pairs to obtain F_{full}. We apply power stretching $\tilde{F}_{\text{full}} = (F_{\text{full}})^{\gamma_{\text{EE}}}$. A piecewise penalty prioritizes coarse errors first (event mismatch, then trigger mismatch, then residual fine error):

$$\mathrm{pen}_{\text{ee}} = \begin{cases} \lambda_E \,\delta_E, & \delta_E > 0, \\ \lambda_T \,\delta_T, & \delta_E = 0 \ \wedge \ \delta_T > 0, \\ \lambda_F \,(1 - F_{\text{full}}), & \delta_E = 0 \ \wedge \ \delta_T = 0. \end{cases} \tag{9}$$

Algorithm 1: ProUIE Training Pipeline

Input: training data $\mathcal{D}$; base policy π_θ; group size K.
Output: trained policy π_θ.

Stage I (CM): Supervised fine-tuning on full $\mathcal{D}$ with original targets, following NER→RE→EE.;
Stage II (SA): Sample $\mathcal{D}_{\text{SA}} \subset \mathcal{D}$ (mainly RE/EE); fine-tune with streamlined targets.;
Stage III (DE): Three GRPO phases;
for *each minibatch* $\{(x, y)\} \sim \mathcal{D}_{\text{NER}}$ **do** `// Phase 1: NER`
  foreach (x, y) *in minibatch* **do**
    Sample K outputs $\{\hat{y}^{(k)}\}_{k=1}^{K} \sim \pi_\theta(\cdot|x)$;
    Compute rewards $\{R^{(k)}_{\text{SFR-NER}}\}$ via Eq. (7);
    Update π_θ with GRPO;
for *each minibatch* $\{(x, y)\} \sim \mathcal{D}_{\text{RE}}$ **do** `// Phase 2: RE`
  foreach (x, y) *in minibatch* **do**
    Sample K outputs $\{\hat{y}^{(k)}\}_{k=1}^{K} \sim \pi_\theta(\cdot|x)$;
    Compute rewards $\{R^{(k)}_{\text{SFR-RE}}\}$ via Eq. (8);
    Update π_θ with GRPO;
for *each minibatch* $\{(x, y)\} \sim \mathcal{D}_{\text{EE}}$ **do** `// Phase 3: EE`
  foreach (x, y) *in minibatch* **do**
    Sample K outputs $\{\hat{y}^{(k)}\}_{k=1}^{K} \sim \pi_\theta(\cdot|x)$;
    Compute rewards $\{R^{(k)}_{\text{SFR-EE}}\}$ via Eq. (10);
    Update π_θ with GRPO;
return π_θ;

Algorithm 2: ProUIE Inference

Input: input text x; task specification $\tau \in \{\text{NER}, \text{RE}, \text{EE}\}$.
Output: structured extraction output $\hat{y}$.
Generate $\hat{y} \sim \pi_\theta(\cdot|x, \tau)$ using the trained model;
return $\hat{y}$;

where $\delta_E = \Delta(E_g, E_p)$ and $\delta_T = \Delta(T_g, T_p)$. The EE reward is:

$$R_{\text{SFR-EE}} = w_E \,\text{cov}(E_g, E_p) + w_T \,\text{cov}(T_g, T_p) + w_F \,\tilde{F}_{\text{full}} - \text{pen}_{\text{ee}}. \tag{10}$$

2.6 Training and Inference Procedure

The learning procedure of ProUIE is presented in Algorithm 1, and its inference procedure is provided in Algorithm 2.

3 Experiments

We organize our experiments around the following research questions:

- **RQ1:** How well does ProUIE perform compared with representative UIE baselines?
- **RQ2:** What are the contributions of each stage (CM/SA/DE) to the final performance?
- **RQ3:** Does SA improve generation efficiency by reducing the length of structured outputs?
- **RQ4:** Which reward granularity matters in DE, and how does the SFR affect learning?
- **RQ5:** How does ProUIE perform for real-world information extraction under strict schema-constrained requirements?

3.1 Experimental Setup

Tasks and datasets. We evaluate three UIE tasks: NER, RE, and EE. Following prior UIE work, we report span-level micro-F1 for NER and RE, and report Trigger F1 / Argument F1 for EE. NER results are summarized in Table 1, where datasets marked with * serve as out-of-domain (OOD) evaluations. RE and EE results are reported in Table 2 and Tables 3–4, respectively. Detailed dataset statistics are provided in Appendix A.1 (Table 8).[1]

Baselines. We compare ProUIE with representative UIE baselines, including traditional schema-driven UIE systems and instruction-tuned/generative UIE models (e.g., UIE (Lu et al., 2022), USM, InstructUIE (Wang et al., 2023), GoLLIE (Sainz et al., 2024), KnowCoder (Li et al., 2024), UniversalNER (Zhou et al., 2024)). In our production-oriented extraction evaluation, we further compare against Qwen2.5 and Qwen3 (Yang et al., 2025) series models.

Evaluation metric. Following prior UIE work (e.g., Wang et al. (2023)), we report span-level micro-F1 (%) for NER/RE and Trigger/Argument F1 for EE. Unless otherwise stated, all results are from a single run with fixed hyperparameters.

Implementation details. We instantiate ProUIE as a 4B-parameter LLM-based extractor (ProUIE (4B)), built on the Qwen3-4B (Yang et al., 2025) backbone. More training details, prompts, and hyperparameters are provided in Appendix A.3.

3.2 Comparison with Baselines (RQ1)

NER. Table 1 reports NER results on diverse benchmarks, including OOD sets (*). Overall, ProUIE achieves the best average score (**79.44**), surpassing UniversalNER (+3.56 F1) and InstructUIE (+8.17 F1). It is particularly strong on biomedical datasets (e.g., `AnatEM`, `bc2gm`,

[1] All datasets, models, and code used in this work are solely for academic research purposes and were not used for any commercial activities.

Table 1: NER results (F1, %). Highest in each row is **bold**; second best is underlined.

Dataset	BERT-base	InstructUIE (11B)	GollIE (7B)	GollIE (13B)	GollIE (34B)	KnowCoder (7B)	UniversalNER (7B)	ProUIE (4B)
ACE2005	87.30	79.94	88.10	89.40	**89.60**	86.10	86.69	86.15
AnatEM	85.82	88.52	–	–	–	86.40	88.65	**89.09**
bc2gm	80.90	80.69	–	–	–	82.00	82.42	**83.87**
bc4chemd	86.72	87.62	–	–	–	–	89.21	**91.97**
bc5cdr	85.28	**89.02**	87.50	87.90	88.40	89.30	89.34	88.93
broad twitter	58.61	80.27	–	–	–	78.30	**81.25**	79.27
CoNLL2003	92.40	91.53	92.80	93.00	93.10	**95.10**	93.30	92.41
FabNER	64.20	78.38	–	–	–	**82.90**	81.87	80.49
FindVehicle	87.13	87.56	–	–	–	**99.40**	98.30	98.00
GENIA-Ent	73.30	75.71	–	–	–	76.70	77.54	**79.46**
HarveyNER	**82.26**	74.69	–	–	–	–	74.21	69.50
multiNERD	91.25	90.26	–	–	–	**96.10**	93.73	95.91
ncbi-disease	80.20	86.21	85.40	86.50	85.80	83.80	**86.96**	86.95
Ontonotes	**91.11**	88.64	83.40	84.00	84.60	88.20	89.91	88.33
polyglot-NER	**75.65**	53.31	–	–	–	–	65.67	74.80
tweetNER7	56.49	65.95	–	–	–	–	65.77	**66.44**
wikiann	70.60	64.47	–	–	–	**87.00**	84.91	83.47
wikineural	82.78	88.27	–	–	–	–	**93.28**	88.18
Movie*	–	63.00	63.00	62.50	62.40	50.00	42.40	**63.16**
Restaurant*	–	20.99	43.40	49.80	**52.70**	48.20	31.70	52.41
AI*	–	49.00	59.10	56.70	61.60	60.30	53.50	**62.36**
Literature*	–	47.21	62.70	59.70	59.10	61.10	59.40	**70.50**
Music*	–	53.16	56.20	67.80	68.40	70.00	65.00	**73.95**
Politics*	–	48.15	57.20	54.40	60.20	72.20	60.80	**78.67**
Science*	–	49.30	55.50	56.20	56.30	59.10	61.10	**61.84**
Avg	–	71.27	–	–	–	–	75.88	**79.44**

Table 2: RE results (F1, %). Highest in each row is bold; second best is underlined.

Dataset	UIE	USM	InstructUIE(11B)	KnowCoder(7B)	ProUIE(4B)
ADE_corpus	–	–	82.31	**84.30**	**84.30**
CoNLL2004	76.00	**78.84**	78.48	73.30	73.17
GIDS	–	–	**81.98**	78.00	77.96
kbp37	–	–	36.14	**73.20**	69.45
NYT	–	–	90.47	93.70	**96.41**
NYT11 HRL	–	–	**56.06**	–	52.68
SciERC	36.53	37.36	45.15	40.00	**46.76**
Semeval RE	–	–	**73.23**	66.30	68.62
Avg	–	–	67.98	–	**71.17**

bc4chemd, GENIA-Ent) while remaining competitive on general-domain benchmarks (e.g., ACE2005, CoNLL2003, Ontonotes). Moreover, ProUIE generalizes well out-of-domain, ranking best on 6/7 OOD datasets and second on Restaurant*. Overall, these results support the effectiveness of our macro-to-micro progressive learning.

RE. Table 2 reports RE results on multiple benchmarks. Overall, ProUIE attains the best average F1 among methods with reported Avg scores (**71.17**), outperforming InstructUIE by +3.19 F1. Notably, ProUIE achieves these results with a smaller backbone (ProUIE 4B vs. InstructUIE 11B / KnowCoder 7B) and without introducing any additional auxiliary information beyond the original training data. ProUIE is particularly strong on benchmarks with richer schemas and diverse relation patterns, such as NYT (96.41) and SciERC (46.76). Meanwhile, other baselines remain competitive on several datasets (e.g., Semeval RE), suggesting that performance varies with dataset characteristics.

EE. Table 3 and Table 4 report event extraction results on trigger identification and argument extraction, respectively. Overall, ProUIE shows mixed performance on EE. On PHEE, ProUIE achieves the best argument F1 (71.86), indicating that the proposed training pipeline can benefit complex argument extraction in certain domains. However, on ACE2005 and CASIE, ProUIE underperforms the strongest baseline for both triggers and arguments, suggesting that EE remains challenging under our current unified generation setting and reward configuration. We leave improving EE stability and closing the gap on general-domain EE benchmarks as an important direction for future work.

Table 3: EE Trigger Identification results (F1, %).

Dataset	BERT-base	UIE	USM	InstructUIE	ProUIE
ACE2005	72.50	73.36	72.41	**77.13**	70.65
CASIE	68.98	69.33	**71.73**	67.80	54.17
PHEE	–	–	–	**70.14**	67.87

Table 4: EE Argument Extraction results (F1, %).

Dataset	BERT-base	UIE	USM	InstructUIE	ProUIE
ACE2005	59.90	54.79	55.83	**72.94**	52.58
CASIE	60.37	61.30	63.26	**63.53**	60.75
PHEE	–	–	–	62.91	**71.86**

3.3 Stage-wise Ablation (RQ2)

To quantify the contribution of each stage, we perform stage-wise ablations:

- **w/o SA:** remove Stage II (SA), i.e., train CM→DE directly.
- **w/o DE:** remove Stage III (DE), i.e., stop after CM→SA.
- **CM only:** use only Stage I (CM) training.

Results. Table 5 shows that each stage contributes to the final performance. Removing DE results in the largest average degradation across the three tasks, with clear drops on NER and notable declines on RE and EE, highlighting the importance of our SFR-based GRPO refinement. Removing SA also consistently hurts performance, particularly on RE and EE, suggesting that streamlining stabilizes structured generation and provides a better starting point for DE.

Table 5: Stage-wise ablation study of ProUIE.

Method	NER	RE	EE
ProUIE (CM+SA+DE)	**79.44**	**71.17**	**61.73**
w/o SA (CM+DE)	78.19	68.68	57.92
w/o DE (CM+SA)	76.22	68.26	58.50
CM only	76.27	70.56	58.07

3.4 Analysis of Streamlined Alignment (SA)

We examine whether SA improves generation efficiency by shortening structured outputs for RE and EE. For this analysis, RE uses `SciERC` and `SemEval RE`, and EE uses `ACE2005` and `CASIE`. We report token-length distributions using percentiles (P50/P70/P99), where each bucket summarizes the min/mean/max token length among examples below the corresponding threshold.

Results. Table 6 shows that SA substantially compresses outputs for both tasks across the entire distribution. For RE, the mean length drops from 70/83/114 tokens (P50/P70/P99) to 13/18/74, and the P99 maximum decreases from 161 to 126. For EE, the mean length decreases from 81/105/171 to 28/37/90, and the P99 maximum is nearly halved, from 199 to 99. These consistent reductions indicate that SA effectively removes redundancy and stabilizes formatting, leading to lower decoding cost in both training (e.g., sampling in DE) and inference.

Table 6: Token-length statistics of generated outputs.

Task	Setting	P50			P70			P99		
		min	mean	max	min	mean	max	min	mean	max
RE	CM	58	70	75	76	83	85	100	114	161
	CM+SA	10	13	20	14	18	35	39	74	126
EE	CM	45	81	150	62	105	170	115	171	199
	CM+SA	16	28	34	18	37	42	48	90	99

Table 7: Coarse-to-fine reward ablation in DE.

Reward Granularity	NER (Avg)	RE (Avg)	EE (Avg)
Coarse-only	77.43	65.88	57.14
Fine-only	77.86	67.44	57.84
Coarse-to-fine SFR	**78.25**	**67.69**	**58.40**

3.5 Effect of SFR Reward Granularity in DE (RQ4)

DE applies GRPO with our stepwise fine-grained reward (**SFR**). For this analysis, we conduct task-specific DE training for NER, RE, and EE using the model after SA. For each task, 5,000 training examples are randomly sampled, and the same amount of training data is used across reward variants. We study the effect of reward granularity by ablating SFR into the following variants:

- **Coarse-only:** only coarse-grained rewards.
- **Fine-only:** only fine-grained rewards.
- **Coarse-to-fine (SFR):** full SFR combining coarse and fine rewards.

Results. Table 7 shows that **coarse-only** rewards underperform, as they mainly encourage high-level correctness but provide limited signal for fixing slot-level errors. **Fine-only** rewards improve over coarse-only by directly optimizing fine units, yet remain sensitive to early mismatches (e.g., incorrect types or grouping), which can hinder stable refinement. Overall, our **coarse-to-fine SFR** achieves the best performance across NER, RE, and EE, indicating that combining coarse guidance with fine-grained matching yields more effective and robust GRPO optimization.

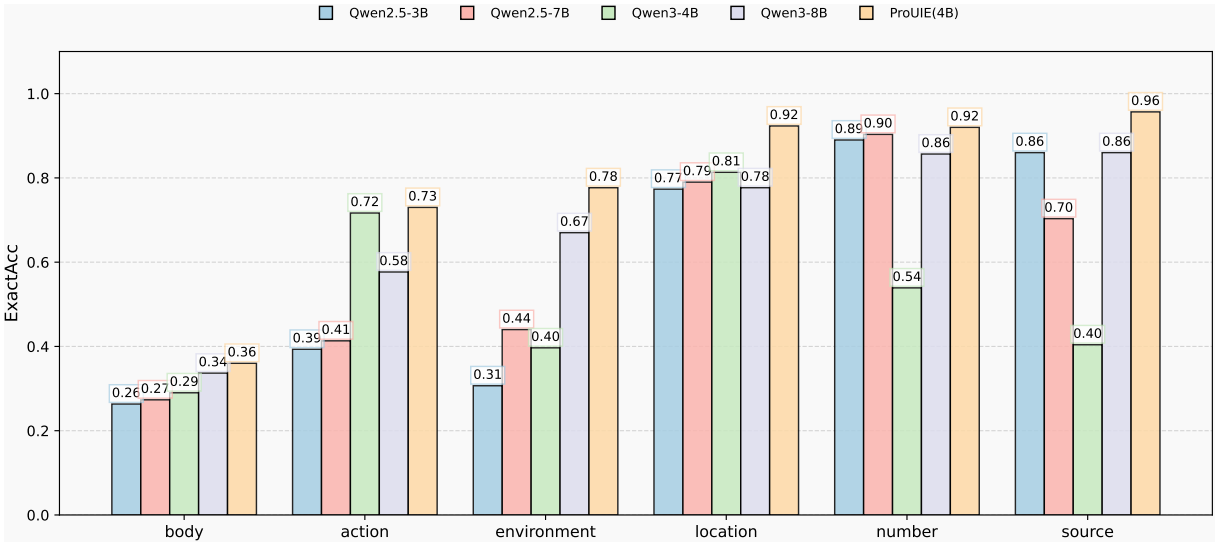

Figure 3: Production-scale mobile query understanding: slot-level exact-match accuracy on six representative fields.

3.6 Case Study

We present qualitative examples to illustrate how each stage improves structured prediction. Appendix A.4 shows representative samples where (i) CM captures the general extraction pattern but produces redundant or inconsistent structures, (ii) SA makes the structure concise and controllable, and (iii) DE further corrects hard structural errors.

3.7 Production-oriented Evaluation (RQ5)

We evaluate ProUIE on a production-oriented query understanding task with strict JSON-constrained slot extraction. Due to confidentiality, we report *ExactAcc* on six representative slots (`body`, `action`, `environment`, `location`, `number`, `source`). As shown in Figure 3, ProUIE consistently outperforms the baselines across all reported slots, demonstrating better alignment to user-facing schema constraints. Details are in Appendix A.5.

4 Related Work

Task-specific information extraction models. Conventional IE research typically develops dedicated models for individual tasks such as Named Entity Recognition (NER), Relation Extraction (RE), and Event Extraction (EE), optimizing task-specific architectures and objectives (Lample et al., 2016; Ma and Hovy, 2016; Lin et al., 2016; Liu et al., 2020). These models often achieve strong in-domain performance, but they require separate training and deployment for different targets, and their interfaces are not unified across tasks.

LLM-based universal information extraction. Universal information extraction (UIE) aims to unify heterogeneous IE tasks under a single modeling and inference framework, often by casting extraction as structured generation. Representative work formulates UIE as schema-driven generation (Lu et al., 2022) and extends it with instruction tuning to unify multiple extraction targets using LLMs (Wang et al., 2023; Xiao et al., 2024; Wei et al., 2024). More recent LLM-based UIE approaches introduce richer guidance signals or additional training stages to improve generalization and controllability, such as guideline-driven supervision (Sainz et al., 2024) and schema/knowledge coding (Bai et al., 2024; Li et al., 2024). In contrast, ProUIE improves UIE without introducing external auxiliary information beyond the original training data, and instead strengthens unified extraction via macro-to-micro progressive learning method.

5 Conclusion

In this paper, we propose ProUIE, a macro-to-micro progressive learning framework for LLM-based universal information extraction that improves UIE without introducing external auxiliary information beyond the original training data. ProUIE consists of three stages: Complete Modeling (CM), Streamlined Alignment (SA), and Deep Exploration (DE). CM builds a unified extraction foundation with task-ordered supervised training, SA streamlines and stabilizes structured outputs, and DE applies GRPO with our coarse-to-fine Stepwise Fine-grained Reward (SFR) to refine

structural correctness. Experiments and analyses demonstrate the effectiveness of ProUIE, and future work will focus on further improving event extraction and reward design.

Limitations

ProUIE shows less consistent improvements on event extraction (EE) across benchmarks, especially under a relatively small backbone setting (4B). This indicates that learning stable event structures under a unified generation interface remains challenging, and we expect further progress to mainly come from stronger event-specific modeling and more appropriate reward design.

In addition, ProUIE adopts a multi-stage training pipeline (CM→SA→DE), where the DE stage relies on GRPO with sampled generations. Although this design does not introduce external supervision beyond the original training data, it incurs additional computational cost compared to pure supervised fine-tuning, which may limit applicability in settings with constrained training budgets.

Acknowledgments

This work was supported by the Suzhou Innovation and Entrepreneurship Leading Talents Programme - Innovation Leading Talent in Universities and Research Institutes with Grant No. ZXL2025310

References

Fan Bai, Junmo Kang, Gabriel Stanovsky, Dayne Freitag, Mark Dredze, and Alan Ritter. 2024. Schema-driven information extraction from heterogeneous tables. *Preprint*, arXiv:2305.14336.

Hao Fei, Shengqiong Wu, Jingye Li, Bobo Li, Fei Li, Libo Qin, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2023. Lasuie: Unifying information extraction with latent adaptive structure-aware generative language model. *Preprint*, arXiv:2304.06248.

Yucan Guo, Zixuan Li, Xiaolong Jin, Yantao Liu, Yutao Zeng, Wenxuan Liu, Xiang Li, Pan Yang, Long Bai, Jiafeng Guo, and Xueqi Cheng. 2023. Retrieval-augmented code generation for universal information extraction. *Preprint*, arXiv:2311.02962.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *Preprint*, arXiv:1603.01360.

Bo Li, Gexiang Fang, Wei Ye, Zhenghua Xu, Jinglei Zhang, Hao Cheng, and Shikun Zhang. 2025. MPL: Multiple programming languages with large language models for information extraction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2403–2414, Vienna, Austria. Association for Computational Linguistics.

Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Xiang Li, Zhilei Hu, Long Bai, Wei Li, Yidan Liu, Pan Yang, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. 2024. Knowcoder: Coding structured knowledge into llms for universal information extraction. *Preprint*, arXiv:2403.07969.

Xincheng Liao, Junwen Duan, Yixi Huang, and Jianxin Wang. 2025. Ruie: Retrieval-based unified information extraction using large language model. *Preprint*, arXiv:2409.11673.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *Preprint*, arXiv:1603.01354.

Chaoxu Pang, Yixuan Cao, Qiang Ding, and Ping Luo. 2023. Guideline learning for in-context information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15372–15389, Singapore. Association for Computational Linguistics.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. Gollie: Annotation guidelines improve zero-shot information-extraction. *Preprint*, arXiv:2310.03668.

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

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi.

2023. Gpt-re: In-context learning for relation extraction using large language models. *Preprint*, arXiv:2305.02105.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *Preprint*, arXiv:2304.08085.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2024. Chatie: Zero-shot information extraction via chatting with chatgpt. *Preprint*, arXiv:2302.10205.

Zhihao Wen, Sheng Liang, Yaxiong Wu, Yongyue Zhang, and Yong Liu. 2025. Effective and efficient schema-aware information extraction using on-device large language models. *Preprint*, arXiv:2505.14992.

Xinglin Xiao, Yijie Wang, Nan Xu, Yuqi Wang, Hanxuan Yang, Minzheng Wang, Yin Luo, Lei Wang, Wenji Mao, and Daniel Zeng. 2024. Yayi-uie: A chat-enhanced instruction tuning framework for universal information extraction. *Preprint*, arXiv:2312.15548.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *Preprint*, arXiv:2403.13372.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2024. Universalner: Targeted distillation from large language models for open named entity recognition. *Preprint*, arXiv:2308.03279.

A Appendix

A.1 Dataset Statistics

We summarize detailed statistics of all datasets used in our experiments, including the number of labels and the train/validation/test split sizes for each task, in Table 8. For EE, the number of event types is reported, with argument roles in parentheses.

Table 8: Detailed dataset statistics. $|\mathcal{Y}|$ denotes the number of labels.

Task	Dataset	$\lvert\mathcal{Y}\rvert$	#Train	#Val	#Test
NER	ACE2005	7	15,000	971	1,060
	broad_twitter_corpus	3	5,334	2,000	2,001
	CoNLL2003	4	15,000	3,250	3,453
	multiNERD	16	134,144	10,000	10,000
	Ontonotes	18	30,000	8,528	8,262
	polyglot-NER	3	393,982	10,000	10,000
	tweetNER7	7	15,000	886	576
	wikiann	3	20,000	10,000	10,000
	wikineural	3	92,720	11,590	11,597
	AnatEM	1	5,861	2,118	3,830
	bc2gm	1	12,500	2,500	5,000
	bc4chemd	1	30,682	30,639	26,364
	bc5cdr	2	4,560	4,581	4,797
	CrossNER_AI	14	–	350	431
	CrossNER_literature	12	–	400	416
	CrossNER_music	13	–	380	465
	CrossNER_politics	9	–	540	650
	CrossNER_science	17	–	450	543
	FabNER	12	9,435	2,182	2,064
	FindVehicle	21	21,565	20,777	20,777
	GENIA	5	15,023	1,669	1,854
	HarveyNER	4	3,967	1,301	1,303
	MIT_Movie	12	–	2,442	2,442
	MIT_Restaurant	8	–	1,520	1,520
	ncbi-disease	1	5,432	923	940
RE	ADE_corpus	1	3,417	427	428
	CoNLL2004	5	922	231	288
	GIDS	4	8,526	1,417	4,307
	kbp37	18	15,917	1,724	3,405
	NYT	24	56,190	5,000	5,000
	NYT11 HRL	12	62,648	149	369
	SciERC	7	1,366	187	397
	semeval RE	10	6,507	1,493	2,717
EE	ACE2005	33(22)	3,342	327	293
	CASIE	5(26)	3,751	788	1,500
	PHEE	2(16)	2,898	961	968

A.2 Prompt Templates

For reproducibility, we provide the prompt templates used in Stage I (CM) and Stage II (SA), referenced in the main text as Figure 4–Figure 6 and Figure 7–Figure 8.

Instruction

```
You are an information extraction
 assistant. Strictly extract {len(slots)}
 slots ({', '.join(slots)}) from the user
 input. Slot values must be exact
 substrings of the input. If a slot has
 multiple values, join them with ' | ' in
 the order of first appearance and remove
 duplicates. The user input is:
```

Input Example

```
Best wishes to Kevin , Therese & their
 family as they embark on the next stage of
 their lives . JG
```

Output Example

```
{
  "location": "",
  "person": "Kevin | Therese",
  "organization": ""
}
```

Figure 4: Prompt template for NER fine-tuning in CM stage.

Instruction

```
You are an information extraction
 assistant. Strictly extract relation pairs
 for the following relation types ({",
 ".join(relation_types)}) from the user
 input. Values must be exact substrings of
 the input. If a relation has multiple
 pairs, join them with ' | ' in the order
 of first appearance, and each pair must be
 formatted as 'word1, word2'. The user
 input is:
```

Input Example

```
Instead of representing scene/object by a
 collection of isolated 3D features -LRB-
 usually points -RRB- , our algorithm uses
 a surface controlled by a small set of
 parameters .
```

Output Example

```
{
  "conjunction": "",
  "feature of": "",
  "hyponym of": "",
  "used for": "surface, algorithm",
  "part of": "",
  "compare": "",
  "evaluate for": ""
}
```

Figure 5: Prompt template for RE fine-tuning in CM stage.

Instruction

```
You are an information extraction assistant. Strictly extract events for the following event
 types from the user input and reply with a single JSON object only. The keys MUST be exactly
 these event types: [<EVENT_TYPES>]. For each event type, group arguments by trigger. Format
 each group as `TRIGGER: ROLE: argument; ROLE: argument`. Valid roles include: [<ROLE_TYPES>].
 The user input is:
```

Input Example

```
After 5 days of treatment with IL-2, the patient developed a hemorrhagic lesion that progressed
 to toxic epidermal necrolysis, as well as grade 4 pancytopenia.
```

Output Example

```
{
"potential therapeutic event": "",
"adverse event": "developed: Subject: patient; Effect: a hemorrhagic lesion that progressed to
 toxic epidermal necrolysis, as well as grade 4 pancytopenia; Treatment: 5 days of treatment
 with IL-2; Treatment.Drug: IL-2; Treatment.Time_elapsed: After 5 days"
}
```

Figure 6: Prompt template for EE fine-tuning in CM stage.

Instruction

```
You are an information extraction assistant...
Please use concise output with no empty fields. The user input is:
```

Input Example

```
..., our algorithm uses a surface controlled by a small set of parameters .
```

Output Example

```
{"used for": "surface, algorithm"}
```

Figure 7: Concise RE output for SA stage.

Instruction

```
You are an information extraction assistant...
Please use concise output with no empty fields. The user input is:
```

Input Example

```
After 5 days of treatment with IL-2, the patient developed a hemorrhagic lesion that progressed
 to toxic epidermal necrolysis, as well as grade 4 pancytopenia.
```

Output Example

```
{
"adverse event": "developed: Subject: patient; Effect: a hemorrhagic lesion that progressed to
 toxic epidermal necrolysis, as well as grade 4 pancytopenia; Treatment: 5 days of treatment
 with IL-2; Treatment.Drug: IL-2; Treatment.Time_elapsed: After 5 days"
}
```

Figure 8: Concise EE output for SA stage.

A.3 Implementation Details

Model and training setup. We use Qwen3-4B (Yang et al., 2025) as the backbone, implement supervised fine-tuning (CM/SA) with LLaMAFactory (Zheng et al., 2024), and perform GRPO-based optimization (DE) with EasyR1. The experiments were conducted primarily on 8 NVIDIA H20 GPUs.

Stage I: Complete Modeling (CM). We conduct supervised fine-tuning in three task-ordered phases. Phase 1 (NER only) uses 845,646 NER instances. Phase 2 (NER+RE, 2:8) jointly trains on 44,951 NER and 179,794 RE instances. Phase 3 (NER+RE+EE, 3:3:4) jointly trains on 7,494 NER, 7,494 RE, and 9,991 EE instances. We train each CM phase for 2 epochs.

Stage II: Streamlined Alignment (SA). We fine-tune on a sampled subset with streamlined targets, consisting of 9,000 RE and 9,000 EE instances, for 2 epochs.

Stage III: Deep Exploration (DE). Starting from the SA checkpoint, we apply GRPO in three task-specific phases. Phase 1 (NER) uses 80,000 instances, with rollout n=4 per input, and runs for 2 epochs. Phase 2 (RE) uses 50,000 instances, with rollout n=8 per input, and runs for 3 epochs. Phase 3 (EE) uses 9,991 instances, with rollout n=8 per input, and runs for 3 epochs.

SFR hyperparameters. We instantiate SFR (Section 2.5.2) with task-specific weights and penalties in Eqs. (7)–(10).

NER. In Eq. (7), we set w_t=0.2, w_p=0.8, and γ_{NER}=1.5. For mismatch penalties, we use λ_t=0.6 and λ_p=0.2 on $\Delta(T_g, T_p)$ and $\Delta(P_g, P_p)$, respectively.

RE. In Eq. (8), we set w_t=0.05, w_h=0.10, w_a=0.10, w_r=0.75, and γ_{RE}=1.3. For the bounded mismatch penalties, we use λ_t=0.15 and λ_r=0.25 on $D_{\text{jac}}(T_g, T_p)$ and $D_{\text{jac}}(R_g, R_p)$.

EE. In Eq. (10), we set w_E=0.05, w_T=0.15, w_F=0.8. For the coarse-to-fine penalty in Eq. (9), we use λ_E=1.0 for event mismatch (δ_E), λ_T=0.5 for trigger mismatch (δ_T), and λ_F=0.3 for residual fine-grained errors $(1 - F_{\text{full}})$.

We summarize the stage-wise data mixtures and example counts in Table 9, which specifies the task composition for CM/SA and the phase-wise datasets used in DE.

Input

```
Labeled data is replaced by a few
    hand-crafted rules that encode basic
    syntactic knowledge .
```

CM output

```
{
  "conjunction":"",
  "feature of":"",
  "hyponym of":"",
  "used for":"hand-crafted rules, basic
    syntactic knowledge",
  "part of":"",
  "compare":"",
  "evaluate for":""
}
```

CM+SA output

```
{
  "used for":"hand-crafted rules, basic
    syntactic knowledge"
}
```

CM+SA+DE output

```
{
  "used for":"hand-crafted rules,
    syntactic knowledge"
}
```

Figure 9: A RE case study showing progressive refinement from CM to SA to DE.

Table 9: Stage-wise data composition used in ProUIE.

Stage	NER	RE	EE	#Examples
CM (Phase 1)	✓	–	–	845,646
CM (Phase 2, 2:8)	✓	✓	–	224,745
CM (Phase 3, 3:3:4)	✓	✓	✓	24,979
SA (1:1)	–	✓	✓	18,000
DE (Phase 1)	✓	–	–	80,000
DE (Phase 2)	–	✓	–	50,000
DE (Phase 3)	–	–	✓	9,991

A.4 Case Study Examples

We provide a representative RE example to show stage-wise improvements. CM predicts the correct relation type but uses a wrong argument span and produces redundant fields; SA removes redundant fields but the argument error remains; DE recovers the correct argument pair under the same schema.

Table 10: Definitions of representative slots in the production-oriented evaluation.

Slot	Definition
body	The main subject of the query (i.e., what the user is searching for or focusing on).
action	The action expressed in the query.
environment	The environment or scenario described in the query.
location	The address/location constraint mentioned in the query.
number	The quantity associated with a body in the query.
source	The search source indicated in the query (e.g., where to search from).

A.5 Production-oriented Task Description

Business scenario. We evaluate ProUIE on a test set from an internal on-device AI search scenario, where the model performs intent understanding for mobile search queries. The extracted structured results are used to support downstream search and user-facing experiences under a fixed JSON schema. The service operates at large scale, serving over one hundred million active users.

Slot definitions. Table 10 summarizes the meanings of the six representative slots reported in Figure 3. We report *ExactAcc* at the slot level: a prediction is considered correct if and only if the predicted slot value exactly matches the gold value.