Can LLMs Learn from Their Mistakes? Self-Correcting Instruction Tuning for Named Entity Recognition

Takumi Takahashi, Tomoki Taniguchi, Chencheng Zhu, Tomoko Ohkuma
Asahi Kasei Corporation
{takahashi.tkr, taniguchi.tcr, zhu.cd, okuma.td}@om.asahi-kasei.co.jp

Abstract

Recent instruction-tuned large language models (LLMs) have demonstrated remarkable performance on various downstream tasks, including named entity recognition (NER). However, previous approaches often generate incorrect predictions, particularly regarding entity boundaries and types. Many of these errors can be corrected to match the ground truth by revising the entity boundaries and/or types. In this paper, we propose a self-correcting instruction tuning approach that simultaneously learns to perform NER and correct errors through natural language instructions. Self-correcting instruction tuning requires only a standard annotated NER dataset. Supervision for self-correction can be automatically generated from error patterns observed in LLMs fine-tuned solely on NER tasks. We conducted extensive experiments on eight NER datasets with two LLMs to validate the effectiveness of the proposed approach. The results demonstrate that the proposed approach enhances NER performance by effectively correcting prediction errors and substantially reducing false positives. We further analyze the self-correction behavior to better understand how the models improve performance.

1 Introduction

Named entity recognition (NER) is a fundamental task in natural language processing that extracts entity mentions from the input text and classifies them into predefined types. Although previous studies have addressed the NER task as a sequence tagging problem (Hammerton, 2003; Lample et al., 2016; Devlin et al., 2019; Yamada et al., 2020), recent advances in large language models (LLMs) have achieved promising results by formulating it as a text generation task (Xie et al., 2023, 2024; Wang et al., 2025). Moreover, state-of-the-art approaches based on instruction tuning have further enhanced the performance of NER (Wang et al., 2023a; Sainz et al., 2024; Zhou et al., 2024; Li et al., 2024).

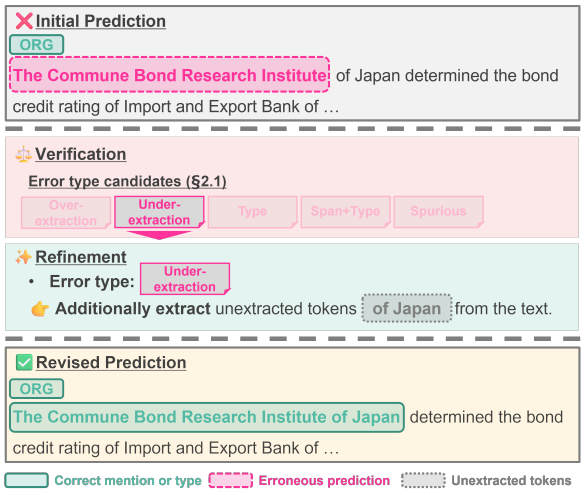

Figure 1: *Top*: Initial prediction of the fine-tuned LLM on the OntoNotes 5.0 dataset. The prediction marked with solid lines is correct, while the prediction marked with dashed lines is incorrect. *Middle*: Verification to classify the predicted entity into an error type and refinement to revise the incorrect entity based on the error type. *Bottom*: Revised prediction after the verification and refinement process.

Despite the promising results of recent approaches, LLMs still face challenges in accurately identifying entity boundaries and types. As shown in Figure 1, the model produced a plausible but incorrect prediction, even when instruction tuning was applied to the target task[1]. For instance, the predicted entity "The Commune Bond Research Institute" has incorrect span boundaries due to unextracted tokens "of Japan," while its entity type, "ORG," is correct. According to our preliminary analysis (§ 2.1), such NER errors remain prevalent.

Self-correction has recently emerged as a promising approach for improving responses generated by LLMs (Madaan et al., 2023; Kamoi et al., 2024; Gou et al., 2024; Kumar et al., 2025; Lee et al.,

[1]We employed the OLMo-2 model for this experiment from the following link: https://huggingface.co/allenai/OLMo-2-0425-1B

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1695–1712

2025). While it has demonstrated remarkable performance on reasoning tasks (Pan et al., 2023; Chen et al., 2024; Ma et al., 2025), its application to information extraction tasks remains unexplored.

In this paper, we propose a self-correcting instruction tuning approach that simultaneously learns to extract entities and correct errors. To this end, the task of self-correcting errors consists of two stages: **self-verification** and **self-refinement**, as illustrated in the middle part of Figure 1. The self-verification stage aims to assess whether the predicted entity is correct. Additionally, it classifies the erroneous entity into a predefined error type, offering clues for the subsequent self-refinement stage. In the self-refinement stage, the model revises its prediction based on the error type identified during the self-verification stage. To implement this approach, supervision for self-correction can be automatically derived from the outputs of LLMs and the ground truth labels of NER (§ 3.4). Therefore, a **manually annotated dataset for self-correction is not required**.

This paper makes the following contributions:

- We propose a self-correcting instruction tuning approach that enables LLMs to perform NER and revise their own mistakes. With this approach, only a standard annotated NER dataset is required.
- We demonstrate the effectiveness of the proposed approach through extensive experiments on eight NER datasets using two LLMs.
- We analyze the self-correction behavior to gain deeper insights into how the proposed approach improves performance.

2 Preliminary Analyses

In this section, we conduct preliminary analyses to investigate error patterns in NER and to validate the effects regarding self-correction in NER. We begin by defining common NER error types and analyzing their distribution across predictions from several models (§ 2.1). We then investigate whether LLMs can revise their own mistakes using only intrinsic knowledge, without relying on external resources or fine-tuning (§ 2.2).

2.1 NER Error Types and Their Distributions

As described in § 1, common NER errors fall into two primary categories: entity boundary errors and type errors. Following Kim et al. (2024), we define five common error types that further detail these categories.

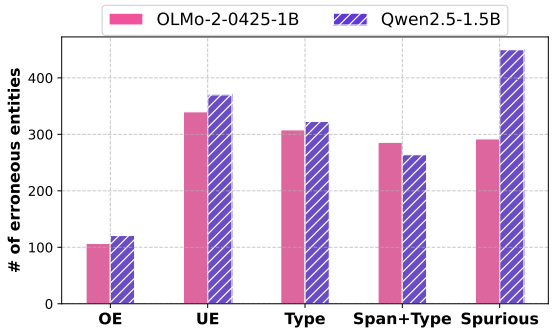

Figure 2: Distribution of NER errors on OntoNotes 5.0. Each model was fine-tuned on the target dataset using the InstructUIE framework (Wang et al., 2023a).

Over-extraction (`OE`). The predicted entity has the correct type, but its boundaries are incorrect due to unnecessary tokens.

Under-extraction (`UE`). The predicted entity has the correct type, but its boundaries are incorrect due to missing tokens.

`Type`. The predicted entity boundaries are correct, but the entity type is incorrect.

`Span+Type`. Both the entity boundaries and type are incorrect.

`Spurious`. The predicted entity does not partially match any ground truth entity boundaries.

Subsequently, we analyze the erroneous entities predicted by the fine-tuned models that appear in the actual dataset. As illustrated in Figure 2, common NER errors remain prevalent, even when models were fine-tuned on the target task. Specifically, four error types excluding `Spurious` account for 70–78% of all errors. These erroneous entities can be aligned with the ground truth by correcting entity boundaries and/or types. In contrast, `Spurious` entities that do not partially match the gold entity boundaries can be discarded to reduce false positives. These findings highlight the importance of addressing such errors to enhance model performance. Therefore, we incorporate five common error types into the self-verification stage.

2.2 Can Intrinsic Self-correction Improve NER Performance?

Self-correction has recently attracted increasing attention in the context of reasoning tasks such as arithmetic reasoning (Madaan et al., 2023), code generation (Chen et al., 2024), and logical reasoning (Pan et al., 2023). However, the potential of

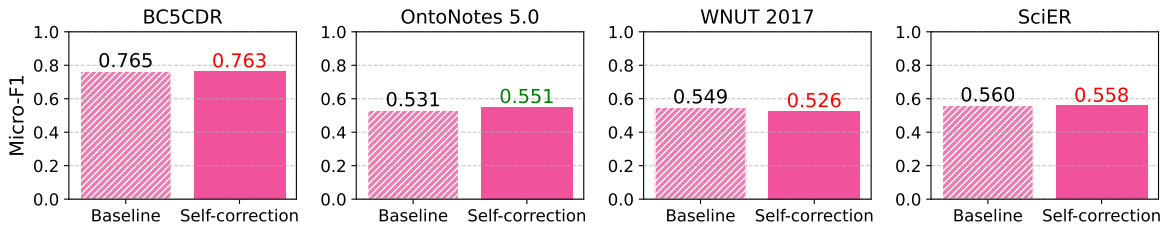

Figure 3: Performance of baseline vs. intrinsic self-correction. We used GPT-4o-mini as the LLM in the experiment. BC5CDR (Wei et al., 2016), OntoNotes 5.0 (Pradhan et al., 2013), WNUT 2017 (Derczynski et al., 2017), and SciER(Zhang et al., 2024a) were used to evaluate the effectiveness of intrinsic self-correction.

self-correction to improve information extraction remains unexplored.

In this section, we examine whether LLMs can self-correct their own mistakes in NER by relying solely on intrinsic knowledge, without access to external resources or fine-tuning. We employ GPT-4o-mini to assess its self-correction ability in NER. Figure 3 illustrates the performance of intrinsic self-correction across various NER datasets. This suggests that intrinsic self-correction still struggles to improve performance, as reported in previous work (Huang et al., 2024; Tyen et al., 2024; Kamoi et al., 2024; Zhang et al., 2025). Therefore, we are motivated to endow LLMs with self-correction capabilities via instruction tuning.

3 Method

In this section, we propose an instruction tuning approach that simultaneously learns to perform NER and correct NER errors through multi-task instruction tuning. We employ InstructUIE (Wang et al., 2023a), an architecture for instruction tuning on various information extraction tasks with natural language instructions. The overview of the proposed model is illustrated in Figure 4.

3.1 Instruction Tuning for NER

In this section, we describe a prevalent method for instruction tuning on NER. Following Wang et al. (2023a), we decompose NER into two auxiliary tasks: span extraction (SE) and entity typing (ET), which are used to train the model.

Named Entity Recognition

Formally, given an input text T and prompt P_{NER}, we aim to generate a list of named entities $S = \{s_1, \ldots, s_n\}$, where s_i is the i-th entity e_i classified with l_i, denoted as $s_i = (e_i, l_i)$. The task prompt P_{NER} consists of three parts: a task instruction I_{NER}, label candidates $L_{\text{NER}} = \{l_1, \ldots, l_M\}$, and their label descriptions $D_{\text{NER}} = \{d_1, \ldots, d_M\}$. The complete prompt is represented as $P_{\text{NER}} = [I_{\text{NER}}; L_{\text{NER}}; D_{\text{NER}}]$, where ; denotes a concatenation operator. We now summarize the input formula to the model as follows: $X = [T; P_{\text{NER}}]$. Therefore, the loss function for generating an output sequence is formalized as follows:

$$\mathcal{L} = -\sum_{t=1}^{N} \log \mathbb{P}(y_t | X, y_{<t}), \tag{1}$$

where y_t denotes a subword token of the generated sequence and N is the total number of subword tokens in the output sequence.

Span Extraction and Entity Typing

Since auxiliary tasks can be leveraged to improve NER performance (Wang et al., 2023a; Zhang et al., 2023), we employ SE as one of the auxiliary tasks. Specifically, given an input text T, task prompt $P_{\text{SE}} = [I_{\text{SE}}; L_{\text{NER}}; D_{\text{NER}}]$, and target label l_i, we define the input formula as follows: $X = [T; P_{\text{SE}}; l_i]$. Therefore, we train the model to perform entity extraction by optimizing Equation 1.

Similarly, we adopt ET as an auxiliary task to boost NER performance. In this task, given a task prompt $P_{\text{ET}} = [I_{\text{ET}}; L_{\text{NER}}; D_{\text{NER}}]$ and target mention e_i in the text T, we also describe the input formula as follows: $X = [P_{\text{ET}}; e_i]$. Finally, we train the model to perform entity typing by optimizing Equation 1. For more details, we provide the full prompts in Appendix E.

3.2 Instruction Tuning for Self-correction

Based on instruction tuning for the downstream tasks described in § 3.1, we endow the model with the ability to verify its predictions and revise its own mistakes. Following previous work in self-correction (Kamoi et al., 2024), we decompose self-correction into two primary components: self-verification (SV) and self-refinement (SR).

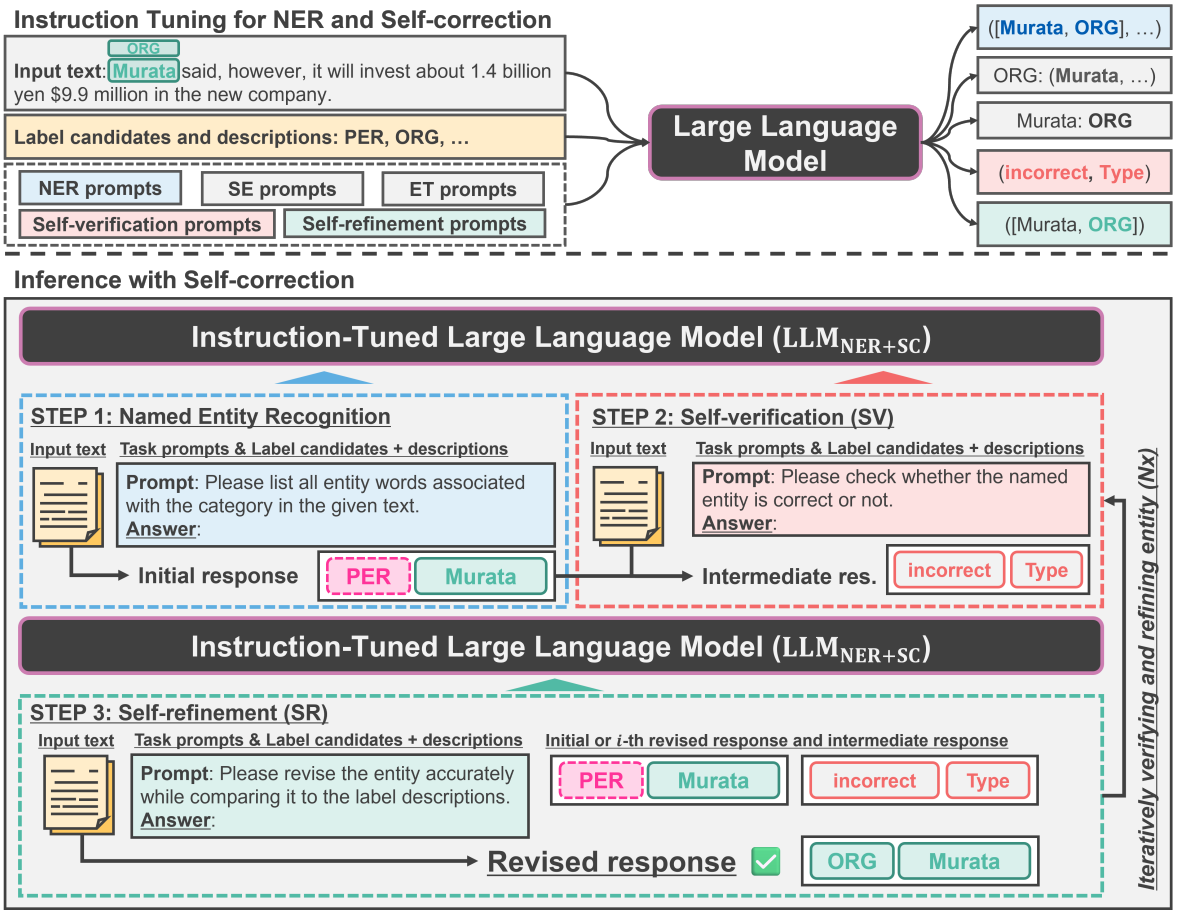

Figure 4: Overview of the proposed model. *Top*: Multi-task instruction tuning on NER and self-correction. *Bottom*: Pipeline for self-correcting NER at test time. In steps 2 and 3, the model verifies its own predictions and iteratively refines them until either the predefined number of iteration steps is reached or the prediction is verified as correct.

Self-verification

This component aims to verify whether the entity generated by the model is correct. The model further classifies its own prediction into predefined error types, as described in § 2.1. Specifically, given an input text T, task prompt $P_{\text{SV}} = [I_{\text{SV}}; D_{\text{NER}}; L_{\text{SV}}; D_{\text{SV}}]$, and predicted entity $\tilde{s}_i$, the model predicts the correctness $\tilde{v}_i$[2] and error type $\tilde{c}_i$[3] of its own prediction. Formally, given the input $X = [T; P_{\text{SV}}; \tilde{s}_i]$, we train the model to verify its own prediction by optimizing Equation 1.

Self-refinement

Finally, the incorrect entity is revised based on the error type $\tilde{c}_i$. Specifically, given an input text T, task prompt $P_{\text{SR}} = [I_{\text{SR}}; L_{\text{NER}}; D_{\text{NER}}]$, predicted entity $\tilde{s}_i$, and its verification results $\tilde{v}_i$ and $\tilde{c}_i$, the model revises its own prediction if $\tilde{v}_i$ is `incorrect` and $\tilde{c}_i$ is not `Spurious`. If $\tilde{v}_i$ is `correct`, the predicted entity is retained as the revised entity. If $\tilde{c}_i$ is `Spurious`, the predicted entity is discarded. To this end, given the input $X = [T; P_{\text{SR}}; \tilde{s}_i; v_i; c_i]$, we train the model to revise the incorrect entity by optimizing Equation 1. For more details, the full prompts are provided in Appendix E.

3.3 Inference with Self-correction

After instruction tuning on the NER and self-correction tasks, the model can perform NER more accurately by correcting its own mistakes during inference. As illustrated in the bottom part of Figure 4, the inference process consists of three components: (1) NER, (2) SV, and (3) SR.

For NER, the model generates a list of named entities $\tilde{S} = \{\tilde{s}_1, \ldots, \tilde{s}_n\}$ from the input text T as its initial response. The output sequence $\tilde{S}$ is fed into the subsequent process, SV.

In the SV stage, the model verifies its initial response $\tilde{s}_i \in \tilde{S}$ from the input text T and task prompts P_{SV} as described in § 3.2. This process yields the correctness $\tilde{v}_i$ and error type $\tilde{c}_i$ corre-

[2]The correctness candidates are `correct` and `incorrect`.

[3]The error type candidates consist of five types, as described in § 2.1 and None ($\tilde{v}_k =$ `correct`).

Algorithm 1 Self-Correction dataset creation

Require: NER dataset $\mathcal{D}_{\text{NER}} = \{\bar{d}_1, \ldots, \bar{d}_K\}$
for $i = 1$ to K **do**
  Train $\text{LLM}_{\text{NER}}^{\setminus \bar{d}_i}$ on all folds except $\bar{d}_i$
  for all u in $\bar{d}_i$ **do**
    Let S be the ground truth entities of u
    $\tilde{S} \leftarrow \text{LLM}_{\text{NER}}^{\setminus \bar{d}_i}(u)$
    for each predicted entity $\tilde{s}_k \in \tilde{S}$ **do**
      if $\tilde{s}_k \in$ ground truth S **then**
        correctness $v_k \leftarrow$ `correct`
        error type $c_k \leftarrow$ `None`
        revised entity $\hat{s}_k \leftarrow \tilde{s}_k$
      else
        $v_k \leftarrow$ `incorrect`
        $c_k \leftarrow \text{classify\_error\_type}(\tilde{s}_k, S)$
        $\hat{s}_k \leftarrow \text{revise\_entity}(\tilde{s}_k, S, c_k)$
      end if
      Add $z = (\tilde{s}_k, v_k, c_k, \hat{s}_k)$ to $\bar{d}_i$
    end for
  end for
end for
return $\mathcal{D}_{\text{NER+SC}} = \bigcup_{i=1}^{K} \bar{d}_i$

sponding to the predicted entity $\tilde{s}_i$. We select the entity $\tilde{s}_i$ as part of the final response if the verification result $\tilde{v}_i$ is `correct`. In contrast, we discard the response $\tilde{s}_i$ classified as $\tilde{c}_i =$ `Spurious`, which cannot be revised. Otherwise, the predicted entity $\tilde{s}_i$ is fed into the subsequent process, SR.

Lastly, the model revises its initial response $\tilde{s}_i$ based on the error type $\tilde{c}_i$ and label description D_{NER} in the SR stage. The output of this stage is then fed back into the SV stage to repeatedly verify its own predictions to further improve NER performance. This repetitive process will be stopped if the predicted entity $\tilde{s}_i$ is verified as `correct`, the error type $\tilde{c}_i$ is classified as `Spurious`, or the predefined number of iteration steps is reached.

3.4 Dataset Creation for Self-correction

In this section, we construct a self-correcting NER dataset $\mathcal{D}_{\text{NER+SC}}$ using only the NER dataset $\mathcal{D}_{\text{NER}}$ and a fine-tuned NER model. Specifically, we first split the NER dataset $\mathcal{D}_{\text{NER}}$ into K-fold subsets $\mathcal{D}_{\text{NER}} = \{\bar{d}_1, \ldots, \bar{d}_K\}$. For each fold $\bar{d}_i$, we train the NER model $\text{LLM}_{\text{NER}}^{\setminus \bar{d}_i}$ on the remaining $K-1$ folds and generate a list of entities $\tilde{S} = \{\tilde{s}_1, \ldots, \tilde{s}_n\}$ using the fine-tuned model. For each entity $\tilde{s}_k$, we determine its correctness $\tilde{v}_k$, error type $\tilde{c}_k$, and revised entity $\hat{s}_k$ by comparing it with the list of gold entities S. Algorithm 1 details the dataset creation process, and dataset statistics are summarized in Appendix A.2.

4 Experiments

4.1 Datasets

To evaluate the baseline and proposed approaches, we employed eight NER datasets from T-NER (Ushio and Camacho-Collados, 2021) and SciER (Zhang et al., 2024a). The T-NER dataset includes seven NER datasets, BC5CDR (Wei et al., 2016), BioNLP 2004 (Collier et al., 2004), CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003), FIN (Salinas Alvarado et al., 2015), OntoNotes 5.0 (Pradhan et al., 2013), WikiAnn (Pan et al., 2017), and WNUT 2017 (Derczynski et al., 2017), which are used to examine the effectiveness of models across various domains and label schemes in the experiment. The statistics of these datasets can be found in Appendix A.1.

4.2 Experimental Setup

Model Configurations. We employed two open-sourced LLMs, OLMo-2 (OLMo et al., 2024) and Qwen2.5 (Yang et al., 2024), as the fundamental models. We used two parameter variants, 1B/1.5B (tiny ♣) and 7B (small ♢), from the OLMo-2 and Qwen2.5 models, respectively. Additionally, we further adopted two parameter variants, 13B/14B (medium ♡) and 32B (large ♠), to assess the consistent impact of parameter scaling in LLMs (§ 4.5).

The baseline model, LLM_{NER}, was trained on NER with the auxiliary tasks (§ 3.1), while the proposed model, $\text{LLM}_{\text{NER+SC}}$, was trained on both the NER and self-correction tasks (§ 3.2). Furthermore, $\text{LLM}_{\text{NER+SC}}$ + Self-correct is a model variant that performs self-correction at test time. For instruction tuning, we adopted QLoRA (Dettmers et al., 2023) to update the parameters in LLMs due to limited computational resources. All hyperparameters in our experiments are in Appendix B.

For prediction, we employed greedy decoding to generate the output sequence of named entities, $\tilde{S} = \{\tilde{s}_1, \ldots, \tilde{s}_n\}$. Additionally, the maximum number of iteration steps in the self-correction process, as described in § 3.3, was set to 8.

Dataset Generation. To construct the self-correcting NER dataset, as described in § 3.4, we first trained the baseline models separately for each size. To collect diverse correct and incorrect predictions generated by LLMs, we applied the instruction-tuned model to each dataset using the Top-K sampling[4] (Fan et al., 2018).

[4] In this experiment, we set top-k to 40 and generated 16 diverse samples per instance.

Model	bc5cdr	bionlp	conll	fin	ontonotes	wikiann	wnut	scier	mean
♣ *Tiny models (OLMo-2 1B and Qwen2.5 1.5B)*									
OLMo_{NER}	0.870	0.731	0.910	0.605	0.835	0.709	0.431	0.789	0.735
$^{\star}\text{OLMo}_{\text{NER+SC}}$	0.877	0.730	0.907	0.674	0.849	0.705	**0.482**	**0.795**	0.752
* + Self-correct	**0.882**	**0.747**	**0.914**	**0.684**	**0.853**	**0.735**	0.414	0.794	**0.753**
† + Self-correct w/ Oracle SV	0.942	0.882	0.965	0.795	0.919	0.819	0.567	0.925	0.852
Qwen_{NER}	0.888	0.735	**0.920**	0.652	0.844	0.705	0.504	0.801	0.756
$^{\star}\text{Qwen}_{\text{NER+SC}}$	0.889	0.742	0.909	0.707	0.854	0.713	**0.506**	0.794	0.764
* + Self-correct	**0.895**	**0.743**	0.914	**0.762**	**0.862**	**0.741**	0.473	**0.805**	**0.774**
† + Self-correct w/ Oracle SV	0.955	0.888	0.970	0.840	0.922	0.824	0.617	0.934	0.869
♢ *Small models (OLMo-2 7B and Qwen2.5 7B)*									
OLMo_{NER}	0.895	0.745	0.922	0.696	0.851	0.723	**0.512**	0.801	0.768
$^{\star}\text{OLMo}_{\text{NER+SC}}$	0.894	0.746	0.927	0.720	0.857	0.724	0.509	0.805	0.773
* + Self-correct	**0.901**	**0.756**	**0.929**	**0.763**	**0.865**	**0.755**	0.497	**0.808**	**0.784**
† + Self-correct w/ Oracle SV	0.957	0.889	0.975	0.854	0.925	0.830	0.635	0.922	0.874
Qwen_{NER}	0.897	0.752	**0.929**	0.719	0.848	0.725	0.500	**0.822**	0.774
$^{\star}\text{Qwen}_{\text{NER+SC}}$	0.895	0.750	0.925	0.737	0.858	0.726	**0.509**	0.813	0.777
* + Self-correct	**0.900**	**0.756**	0.927	**0.768**	**0.868**	**0.751**	0.484	0.815	**0.784**
† + Self-correct w/ Oracle SV	0.961	0.890	0.978	0.873	0.927	0.828	0.622	0.937	0.877

Table 1: Experimental results of the baseline and proposed models on eight NER datasets. The shaded rows (gray) represent the baseline models trained solely on the NER task with the auxiliary tasks. The mark $\star$ indicates the proposed approach and † indicates the upper bound performance in self-correction with an oracle self-verification. The green and red values indicate increased and decreased performance compared to the baseline model, respectively.

Evaluation Metric. We employed the Micro-F1 score for a standard NER evaluation. To measure performance, we consider a predicted entity $\tilde{s}_i$ to be correct if it exactly matches any entity in the gold entities $S = \{s_1, \ldots, s_n\}$.

4.3 Results

Table 1 presents the main results. We observe that the self-correcting instruction tuning, denoted as $^{\star}\text{LLM}_{\text{NER+SC}}$, consistently improves overall NER performance on all model variants. The results suggest that incorporating self-correction as an auxiliary task during instruction tuning allows the model to identify and revise its mistakes, thereby improving base NER performance. Furthermore, applying self-correction to inference, denoted as * + Self-correct, yields further improvements in NER performance. While intrinsic self-correction does not perform well (§ 2.2), our approach improves NER performance by self-correcting errors at test time. Furthermore, our approaches show performance improvements in 81% and 75% (♣ and ♢) of the datasets, respectively. These results support the effectiveness of our approaches across various domains rather than being limited to specific ones.

4.4 Self-correction using Oracle Verification

We also assess the effectiveness of self-correction involving an oracle verifier that receives the ground truth of SV and then feeds it into the subsequent SR stage. As shown in the blue rows of Table 1, denoted as † w/ Oracle SV, these models have drastically improved NER performance across all datasets. These results indicate that the models are capable of effectively correcting their own errors when accurate SV outputs are provided. Therefore, the bottleneck of self-correction lies in SV rather than SR, which is consistent with previous work (Tyen et al., 2024). These findings motivate future work to enhance SV performance to further improve NER performance.

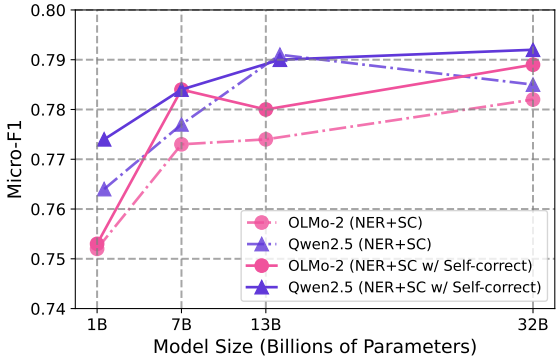

Figure 5: Performance scaling across different model sizes (averaged Micro-F1 scores over eight NER datasets). NER+SC and NER+SC w/ Self-correct refer to $^{\star}\text{LLM}_{\text{NER+SC}}$ and * + Self-correct, respectively.

4.5 Performance Scaling on Large Models

We demonstrated the effectiveness of our approach on tiny (♣) and small (♢) model sizes, as described in § 4.3. In this section, we extend the analysis to

Error type	$\text{OLMo}_{\text{NER+SC}}$	$\text{Qwen}_{\text{NER+SC}}$
None	0.919	0.928
OE	0.262	0.277
UE	0.396	0.453
Type	0.255	0.352
Span+Type	0.291	0.389
Spurious	0.528	0.591
Micro-F1	0.850	0.863
Macro-F1	0.500	0.550

Table 2: Performance of self-verification (averaged F1 scores over eight NER dataset). Tiny models (♣) are used in the experiment. Details are in Appendix D.

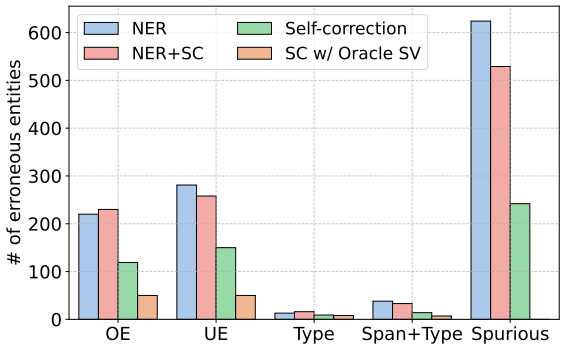

Figure 6: Distributions of erroneous entities across different error types in BC5CDR. NER, NER+SC, Self-correction, and SC w/ Oracle SV correspond to OLMo_{NER}, $\text{OLMo}_{\text{NER+SC}}$, $\text{OLMo}_{\text{NER+SC}}$ + Self-correct, and $\text{OLMo}_{\text{NER+SC}}$ + Self-correct w/ Oracle SV, respectively.

larger models, such as ♡ and ♠. As illustrated in Figure 5, our approach consistently improves NER performance as the model size increases.

5 Discussion

In this section, we analyze the self-correction behavior of our approach to gain deeper insights into the mechanisms underlying its performance improvements.

5.1 Performance of Self-verification

In § 4.4, we demonstrated that self-verification performance still has substantial room for improvement. To this end, we evaluate the performance of self-verification using two model variants, $\text{OLMo}_{\text{NER+SC}}$ (♣) and $\text{Qwen}_{\text{NER+SC}}$ (♣).

As shown in Table 2, the proposed approaches perform well on correct entities (None). However, the performance of these models on other error types (i.e., incorrect entities) remains insufficient. Unlike previous studies (Huang et al., 2024; Zhang et al., 2025), our results suggest that the proposed approach prevents the excessive refinement of correct entities. However, it still struggles to adequately revise its own mistakes. Therefore, in the subsequent section, we further analyze erroneous entities by error type in predictions.

5.2 Distribution of Erroneous Entities

Figure 6 illustrates the distribution of erroneous entities across different error types. Comparing OLMo_{NER} with $\text{OLMo}_{\text{NER+SC}}$, our approach reduces the number of erroneous entities across several error types. This indicates that self-correcting instruction tuning not only improves NER performance but also helps prevent the generation of erroneous entities. Additionally, $\text{OLMo}_{\text{NER+SC}}$ + Self-correct further reduces the generation of erroneous entities across all error types, especially Spurious. Therefore, the proposed approach contributes to further refining the LLM's predictions in NER. The detailed case study can be found in Appendix C.

5.3 Ablation Study

Role of Self-correcting Instruction Tuning

In this section, we investigate how self-correcting instruction tuning endows LLMs with the capability to understand correct and incorrect patterns in NER. As described in § 3.1, we employed two auxiliary tasks, SE and ET. These tasks are designed to enhance the model's understanding of entity boundaries and types, while self-correction aims to refine this understanding. To assess their impact, we conduct an ablation study on instruction tuning.

As shown in Table 3, we first observe that NER performance of the baseline model, OLMo_{NER}, decreases when it is trained solely on the NER task (i.e., w/o SE and ET). This indicates that the auxiliary tasks, SE and ET, play an important role in better understanding the NER task via instruction tuning.

In contrast, self-correcting instruction tuning without auxiliary tasks (i.e., w/o SE and ET) does not negatively affect NER performance. Furthermore, this result is consistent with applying self-correction at test time. These findings suggest that SV and SR tasks endow LLMs with the capability to understand not only correct and incorrect patterns in NER but also characteristics of entity boundaries and types.

Model	bc5cdr	bionlp	conll	fin	ontonotes	wikiann	wnut	scier	mean
♣ *w/o self-correcting instruction tuning*									
OLMo$_{\text{NER}}$	0.870	0.731	0.910	0.605	0.835	0.709	0.431	0.789	0.735
w/o SE and ET	0.852	0.730	0.916	0.417	0.842	0.715	0.451	0.785	0.714
♣ *w/ self-correcting instruction tuning*									
OLMo$_{\text{NER+SC}}$	0.877	0.730	0.907	0.674	0.849	0.705	0.482	0.795	0.752
w/o SE and ET	0.879	0.735	0.917	0.655	0.845	0.712	0.466	0.793	0.750
w/o SV	0.873	0.734	0.908	0.667	0.848	0.712	0.471	0.791	0.750
w/o SR	0.880	0.737	0.913	0.660	0.849	0.708	0.460	0.794	0.750
OLMo$_{\text{NER+SC}}$ + Self-correct	0.882	0.747	0.914	0.684	0.853	0.735	0.414	0.794	0.753
w/o SE and ET	0.882	0.747	0.923	0.673	0.855	0.738	0.414	0.795	0.753
w/o SV	0.874	0.730	0.912	0.691	0.854	0.713	0.474	0.795	0.755
w/o SR	0.886	0.740	0.917	0.720	0.856	0.734	0.362	0.794	0.751

Table 3: Results of the ablation study on self-correcting instruction tuning (Micro-F1 scores).

Self-verification vs. Self-refinement

As shown in Table 3, we also investigate how introducing SV and SR contributes to the NER task. For the base model OLMo$_{\text{NER+SC}}$, SV and SR tasks contribute equally to the downstream task. However, when applying self-correction at test time (i.e., OLMo$_{\text{NER+SC}}$ + Self-correct), NER performance varies depending on SV performance. For the WNUT 2017 dataset, although NER performance substantially increases when applying self-correction without SV, it drastically decreases when SR alone is excluded[5]. This is because SV performance on WNUT 2017 is relatively lower than that on other datasets, as shown in Table 10. In contrast, NER performance of OLMo$_{\text{NER+SC}}$ + Self-correct is relatively high on BC5CDR, BioNLP 2004, and WikiAnn, where SV performance is also high. Therefore, the SV task plays a more critical role in correcting NER errors, enabling the model to revise its own mistakes more precisely.

5.4 Impact of Iterative Self-correction

We conduct a quantitative analysis of the trade-offs between the number of self-correction iterations and performance improvements to better understand how iterative self-correction contributes to overall performance. To this end, we used 1,000 randomly sampled instances from the entire `Dev` split of OntoNotes 5.0. For a comprehensive analysis, we employed OLMo$_{\text{NER+SC}}$ with different parameter sizes, denoted as ♣, ♢, ♡, and ♠.

As illustrated in the left part of Figure 7, the proposed approach increases NER performance by up to 1.93% after three iterations. While applying self-correction only once yields sufficient performance improvements, iterative self-correction is consistently more effective. Furthermore, since all variants reach performance saturation after four iterations, the associated computational cost is likely acceptable for real-world applications.

We further analyze the relationship between precision and recall to better understand how the proposed approach behaves during iterative self-correction. As illustrated in the middle part of Figure 7, self-correction substantially improves overall precision by up to 6.12% compared to the baseline model OLMo$_{\text{NER+SC}}$, which does not apply self-correction. However, as illustrated in the right part of Figure 7, self-correction moderately decreases recall by up to 2.51% when the number of iterations is one, resulting in a trade-off between precision and recall. Nevertheless, iterative self-correction slightly recovers its recall after two iterations, indicating that the proposed approach can revise its own mistakes arising from self-correction.

6 Related Work

6.1 Named Entity Recognition

Traditional approaches formulate NER as a sequence tagging problem, predicting the probability distribution over entity types and assigning a predefined label to each token (Hammerton, 2003; Lample et al., 2016; Devlin et al., 2019; Yamada et al., 2020). In contrast, generative approaches have been proposed to formulate NER as a text generation task (Xie et al., 2023, 2024; Wang et al., 2025). Based on instruction tuning approaches, models have shown remarkable performance in a wide range of datasets (Wang et al., 2023a; Sainz et al., 2024; Zhou et al., 2024; Li et al., 2024). However, erroneous predictions, including entity boundaries and types, remain a persistent challenge.

To address this issue, Li et al. (2023) explored

[5]The detailed inference process for self-correction without SV and SR is described in Appendix B.

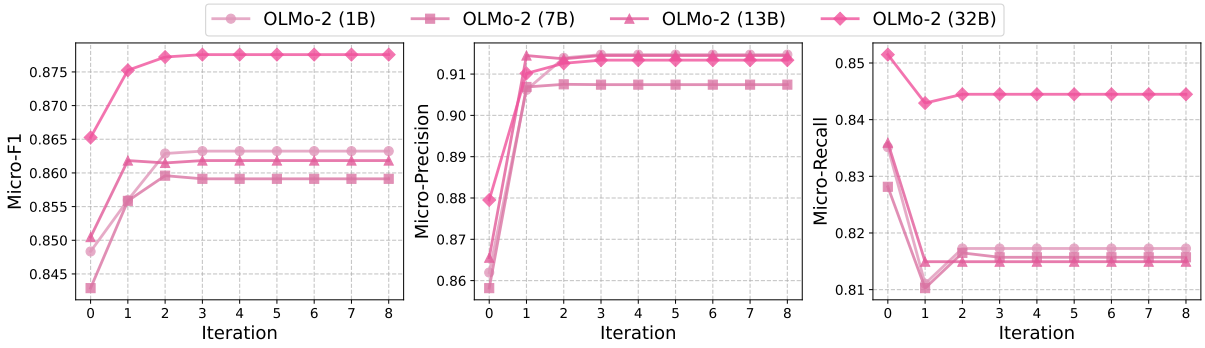

Figure 7: Performance improvements over iterative self-correction on OntoNotes 5.0. $\text{OLMo}_{\text{NER+SC}}$ with different model sizes was used in the experiments. An iteration count of 0 indicates that the model extracts entities without self-correction; for higher counts, self-correction is applied iteratively.

an approach to correct erroneous entities without re-training, by leveraging a gazetteer that maps correct entities to their types. The approach is a training-free method to revise its own mistakes, but relies heavily on the gazetteer. Meanwhile, Wang et al. (2025) introduced a self-verification strategy for NER to validate predicted entities, but it does not focus on error correction. Additionally, Kim et al. (2024) proposed a post-hoc approach to revise incorrect entities via knowledge-grounded reasoning based on self-consistency (Wang et al., 2023b). However, this approach has two shortcomings: (1) an elaborate knowledge base is not always available, and (2) self-consistency-based refinement ignores entities with minority votes.

This study focuses on correcting erroneous entities in LLMs through self-correcting instruction tuning, which enables an LLM to perform NER and correct its own mistakes. Notably, the dataset used for self-correcting instruction tuning can be constructed solely from a standard NER dataset.

6.2 Self-correction

Self-correction aims to improve responses from LLMs by verifying their own responses and refining them (Madaan et al., 2023; Chen et al., 2024; Pan et al., 2023; Gou et al., 2024; Kamoi et al., 2024; Kumar et al., 2025). Despite the significant capabilities of LLMs, recent studies have reported negative results, showing that *intrinsic* self-correction often fails to correct mistakes (Huang et al., 2024; Zhang et al., 2025). To tackle this issue, recent studies have explored the effectiveness of supervised fine-tuning (Ranaldi and Freitas, 2024; Zhang et al., 2024b; Lee et al., 2025). While these approaches have focused mainly on reasoning tasks (Pan et al., 2023; Chen et al., 2024; Ma et al., 2025), little work has explored self-correction to enhance performance in information extraction tasks, such as NER.

This study is the first attempt to explore the effectiveness of self-correction in NER through self-correcting instruction tuning.

7 Conclusion

In this paper, we proposed a self-correcting instruction tuning approach that simultaneously learns to perform NER and correct its own mistakes. Our approach requires only a standard annotated NER dataset, as supervision for self-correcting instruction tuning can be derived from the outputs of LLMs. Experimental results demonstrated remarkable performance improvements across eight NER datasets using two LLMs. We further analyzed the behavior of self-correction to better understand our approach and bridge the gap toward future research.

As future work, we first aim to improve self-verification to further enhance NER performance. We also plan to explore the application of our approach to other information extraction tasks, such as relation extraction.

Limitations

While our approach demonstrated notable performance improvements across eight widely used NER datasets and two LLMs, several limitations remain. First, the zero-shot setting, widely used in prior work to assess the generalization ability of models (Wang et al., 2023a; Sainz et al., 2024; Zhou et al., 2024), is not directly focused on this study. Therefore, our approach may be suboptimal in this scenario. We plan to evaluate the effectiveness of our approach under such conditions in

future work. Second, the iterative self-correction process at test time increases the computational cost compared to previous methods. This issue can be mitigated by adjusting the number of iterations in the self-correction process, but the trade-offs between efficiency and performance need further exploration.

Ethics Statement

This research was conducted in accordance with the ACL Ethics Policy. This study focuses on named entity recognition using self-correcting instruction tuning. We employed two open-sourced LLMs, OLMo-2 (OLMo et al., 2024) and Qwen2.5 (Yang et al., 2024), both of which are released under the Apache 2.0 license. We used them in accordance with the terms of their respective licenses. The datasets used in our experiments may contain personally identifiable information (PII), as is common in NER tasks. However, all datasets were publicly released for research purposes, and we used them strictly within that context. No manual inspection or annotation of sensitive content was performed, and our models were not trained to identify or generate personal information beyond the existing dataset annotations.

Acknowledgments

We used ABCI 3.0 provided by AIST and AIST Solutions.

References

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.

Nigel Collier, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland. COLING.

Michael Han Daniel Han and Unsloth team. 2023. Unsloth.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 172–175.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440.

Seoyeon Kim, Kwangwook Seo, Hyungjoo Chae, Jinyoung Yeo, and Dongha Lee. 2024. VerifiNER: Verification-augmented NER via knowledge-grounded reasoning with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2441–2461, Bangkok, Thailand. Association for Computational Linguistics.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 2025. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. 2025. ReVISE: Learning to refine at test-time via intrinsic self-verification. In *Forty-second International Conference on Machine Learning*.

Kuai Li, Chen Chen, Tao Yang, Tianming Du, Peijie Yu, Dong Du, and Feng Zhang. 2023. Type enhanced BERT for correcting NER errors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7124–7131, Toronto, Canada. Association for Computational Linguistics.

Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Lixiang Lixiang, 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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8758–8779, Bangkok, Thailand. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. 2025. S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22632–22654, Vienna, Austria. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2024. 2 olmo 2 furious. *Preprint*, arXiv:2501.00656.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Leonardo Ranaldi and Andre Freitas. 2024. Self-refine instruction-tuning for aligning reasoning in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2325–2347, Miami, Florida, USA. 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. In *The Twelfth International Conference on Learning Representations*.

Julio Cesar Salinas Alvarado, Karin Verspoor, and Timothy Baldwin. 2015. Domain adaption of named entity recognition to support credit risk assessment. In *Proceedings of the Australasian Language Technology Association Workshop 2015*, pages 84–90, Parramatta, Australia.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13894–13908, Bangkok, Thailand. Association for Computational Linguistics.

Asahi Ushio and Jose Camacho-Collados. 2021. T-NER: An all-round python library for transformer-based named entity recognition. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 53–62, Online. Association for Computational Linguistics.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, Guoyin Wang, and Chen Guo. 2025. GPT-NER: Named entity recognition via large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4257–4275, Albuquerque, New Mexico. Association for Computational Linguistics.

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. 2023a. Instructuie: Multi-task instruction tuning for unified information extraction. *Preprint*, arXiv:2304.08085.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Jiao Li, Thomas C. Wiegers, and Zhiyong Lu. 2016. Assessing the state of the art in biomedical relation extraction: overview of the biocreative v chemical-disease relation (cdr) task. *Database*, 2016:baw032.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tingyu Xie, Qi Li, Jian Zhang, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2023. Empirical study of zero-shot NER with ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7935–7956, Singapore. Association for Computational Linguistics.

Tingyu Xie, Qi Li, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2024. Self-improving for zero-shot named entity recognition with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 583–593, Mexico City, Mexico. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Jiasheng Zhang, Xikai Liu, Xinyi Lai, Yan Gao, Shusen Wang, Yao Hu, and Yiqing Lin. 2023. 2INER: Instructive and in-context learning on few-shot named entity recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3940–3951, Singapore. Association for Computational Linguistics.

Qi Zhang, Zhijia Chen, Huitong Pan, Cornelia Caragea, Longin Jan Latecki, and Eduard Dragut. 2024a. SciER: An entity and relation extraction dataset for datasets, methods, and tasks in scientific documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13083–13100, Miami, Florida, USA. Association for Computational Linguistics.

Qingjie Zhang, Di Wang, Haoting Qian, Yiming Li, Tianwei Zhang, Minlie Huang, Ke Xu, Hewu Li, Liu Yan, and Han Qiu. 2025. Understanding the dark side of LLMs' intrinsic self-correction. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27066–27101, Vienna, Austria. Association for Computational Linguistics.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. 2024b. Small language models need strong verifiers to self-correct reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15637–15653, Bangkok, Thailand. Association for Computational Linguistics.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2024. UniversalNER: Targeted distillation from large language models for open named entity recognition. In *The Twelfth International Conference on Learning Representations*.

A Dataset Statistics

In this section, we provide detailed statistics for each dataset used in the experiments.

A.1 NER Datasets

In this paper, we conducted comprehensive experiments on eight NER datasets, as described in § 4. For T-NER (Ushio and Camacho-Collados, 2021), the datasets were downloaded from the Hugging Face repository: `https://huggingface.co/tner/datasets`, while SciER (Zhang et al., 2024a) was obtained from the GitHub repository:

Dataset	# types	Train	Dev	Test
BC5CDR	2	5,228	5,330	5,865
BioNLP 2004	5	16,619	1,927	3,856
CoNLL 2003	4	14,041	3,250	3,453
FIN	4	1,018	150	305
OntoNotes 5.0	18	59,924	8,528	8,262
WikiAnn	3	20,000	10,000	10,000
WNUT 2017	6	2,395	1,009	1,287
SciER	3	5,575	713	854

Table 4: Dataset statistics for NER. Train, Dev, and Test indicate the number of sentences in each split.

https://github.com/edzq/SciER. For experiments and analyses, we divided each dataset into three splits: Train, Dev, and Test. We used Test split in the main experiment and ablation study, as described in § 4 and § 5.3. We also used Dev split in analyses, as described in § 2, § 5.1, § 5.2, and § 5.4. The statistics of these datasets are shown in Table 4.

A.2 Self-correction Datasets

To perform self-correcting instruction tuning, we first generated datasets to train the model, as described in § 3.4. The statistics of the generated datasets are shown in Table 5 and 6.

B Implementation Details

Training Configurations. We employed two open-sourced LLMs, OLMo-2 (OLMo et al., 2024) and Qwen2.5(Yang et al., 2024). These models can be downloaded from the following repositories: OLMo-2 (https://huggingface.co/allenai/OLMo-2-0425-1B) and Qwen2.5 1.5B (https://huggingface.co/Qwen/Qwen2.5-1.5B), respectively. We used these models via the Hugging Face Transformers library (Wolf et al., 2020). For instruction tuning, we employed QLoRA (Dettmers et al., 2023) with $\alpha = 128$ and $r = 64$ to update the learnable parameters in LLMs. The models were trained for 3 epochs using the AdamW optimizer (Loshchilov and Hutter, 2019) with a batch size of 32 and a learning rate of 2×10^{-5}. We used a cosine learning rate schedule with a warmup ratio of 0.1. To avoid overfitting, we applied early stopping based on the NER Micro-F1 score. We trained the model on a single NVIDIA H200 GPU, which took approximately 3 to 32 hours[6] to complete 3 epochs. To reduce computation time during instruction tuning, we employed the Unsloth library (Daniel Han and team, 2023).

Inference Configurations. For inference, we used greedy decoding to generate named entities, their verification results, and their revisions. The maximum number of subword tokens was set to 128, 12, and 16 for NER, SV, and SR, respectively. In § 5.3, self-correction without SV directly revises all predicted entities once. Additionally, self-correction without SR applies only the SV component after NER prediction. Specifically, a predicted entity is discarded if the SV component predicts the verification result $\tilde{v}_i$ as incorrect, and included in the final outputs if $\tilde{v}_i$ is predicted as correct.

[6]The training time depends heavily on the model and dataset sizes.

C Case Study

In this section, we present and discuss representative case studies that shed light on the model's strengths and limitations.

UE error correction. Table 7 presents an example of UE error correction. The baseline model, OLMo_{NER}, fails to extract any correct entities. In contrast, the proposed model, $\text{OLMo}_{\text{NER+SC}}$, successfully extracts the correct entity ('STZ', 'chemical'), although an incorrect one is also extracted. The proposed model with self-correcting inference, $\text{OLMo}_{\text{NER+SC}}$ + Self-correct, retains only the correct entity while the incorrect entity is discarded during the self-verification process. Since the error type of the discarded entity is UE which can be revised during self-refinement, designing a more accurate refinement process is left for future work.

Type error correction. Table 8 presents an example of Type error correction. Although the baseline model generates the correct span boundaries, it misclassifies their types. The proposed model $\text{OLMo}_{\text{NER+SC}}$ successfully recognizes entity mentions and classifies them with the correct type. This demonstration suggests that self-correcting instruction tuning enables LLMs to better understand the characteristics of entity types, as discussed in § 5.3.

Spurious error correction. Table 9 represents an example of Spurious error correction. The baseline model extracts an incorrect entity, which has no partial overlap with any ground truth entity. On the other hand, the proposed model, $\text{OLMo}_{\text{NER+SC}}$, successfully extracts correct entities that are missed by the baseline model. Furthermore, the proposed model with self-correcting inference, $\text{OLMo}_{\text{NER+SC}}$ + Self-correct, also extracts correct entities while discarding the incorrect entity classified as Spurious.

Dataset	Correct	OE	UE	Type	Span+Type	Spurious
♣ OLMo-2						
BC5CDR	8,063	873	731	57	111	3,297
BioNLP 2004	37,170	4,268	7,074	4,335	3,714	13,561
CoNLL 2003	22,151	424	682	1,779	572	1,823
FIN	590	76	80	67	91	2,022
OntoNotes 5.0	68,638	9,528	10,871	13,683	13,788	32,394
WikiAnn	20,132	1,122	4,527	11,439	3,541	14,937
WNUT 2017	1,147	212	221	636	327	5,679
SciER	15,364	2,665	2,410	1,352	1,262	10,210
♣ Qwen2.5						
BC5CDR	8,116	634	598	50	88	1,862
BioNLP 2004	37,454	4,108	6,700	4,263	3,731	12,592
CoNLL 2003	22,220	252	561	1,295	500	1,138
FIN	589	48	72	28	77	1,640
OntoNotes 5.0	69,010	6,866	12,462	15,029	12,051	22,709
WikiAnn	19,643	547	2,192	6,561	1,599	6,503
WNUT 2017	1107	102	129	298	114	1,334
SciER	15,107	1,568	1,388	966	778	3408

Table 5: Dataset statistics for self-correction in the Train split. Each number represents the number of entities in the category. To generate these datasets, we used two LLMs fine-tuned on the target dataset: OLMo-2 (♣) and Qwen2.5 (♣). The number of sentences in each dataset is equal to the NER dataset.

Dataset	Correct	OE	UE	Type	Span+Type	Spurious
♢ OLMo-2						
BC5CDR	8,152	526	570	28	58	1,573
BioNLP 2004	37,609	4,217	7,204	4,346	3,655	11,789
CoNLL 2003	22,317	244	544	1,110	427	774
FIN	613	50	98	43	68	1,493
OntoNotes 5.0	69,347	5,990	9,989	12,149	9,061	20,132
WikiAnn	19,419	342	1,485	3,567	875	4,356
WNUT 2017	1,130	83	108	263	90	1,130
SciER	15,116	1,426	1,350	970	698	2,984
♢ Qwen2.5						
BC5CDR	8,155	470	483	38	58	1,201
BioNLP 2004	37,370	3,381	6,492	4,132	3,231	9,557
CoNLL 2003	22,312	205	463	1,208	389	608
FIN	600	38	82	32	49	838
OntoNotes 5.0	69,118	4,939	8,366	10,705	7,726	14,884
WikiAnn	19,609	421	1,765	4,992	1,230	4,643
WNUT 2017	1,102	54	93	263	78	772
SciER	15,180	1,190	1,164	734	685	2,328

Table 6: Dataset statistics for self-correction in the Train split. Each number represents the number of entities in the category. To generate these datasets, we used two LLMs fine-tuned on the target dataset: OLMo-2 (♢) and Qwen2.5 (♢). The number of sentences in each dataset is equal to the NER dataset.

Example of UE error correction on BC5CDR (Wei et al., 2016)	
Input Text T	In Alzheimer's disease groups, rats were injected with STZ - icv bilaterally (3 mg / kg) in first day and 3 days later, a similar STZ - icv application was repeated.
Ground Truth S	[['Alzheimer's disease', 'disease'], ['STZ', 'chemical']]
Model	*Prediction*
OLMo_{NER}	[['s disease', 'disease']]
$\text{OLMo}_{\text{NER+SC}}$	[['s disease', 'disease'], ['STZ', 'chemical']]
$\text{OLMo}_{\text{NER+SC}}$ + Self-correct	[['STZ', 'chemical']]

Table 7: Example of an UE error correction from the SciER dataset. The correct and incorrect entities are highlighted in the table. The format of the ground truth and prediction is as follows: [['entity', 'type'], . . .].

Example of Type error correction on SciER (Zhang et al., 2024a)	
Input Text T	In addition both Granard Town Council and Longford Town Council were abolished.
Ground Truth S	[['Granard Town Council', 'location'], ['Longford Town Council', 'location']]
Model	*Prediction*
OLMo_{NER}	[['Granard Town Council', 'organization'], ['Longford Town Council', 'organization']]
$\text{OLMo}_{\text{NER+SC}}$	[['Granard Town Council', 'location'], ['Longford Town Council', 'location']]
$\text{OLMo}_{\text{NER+SC}}$ + Self-correct	[['Granard Town Council', 'location'], ['Longford Town Council', 'location']]

Table 8: Example of a Type error correction from the SciER dataset. The correct and incorrect entities are highlighted in the table. The format of the ground truth and prediction is as follows: [['entity', 'type'], . . .].

Example of Spurious error correction on BioNLP 2004 (Collier et al., 2004)	
Input Text T	Fewer virus-exposed cells from elderly donors stained for Fos and Jun at each data point compared with cells from young donors.
Ground Truth S	[['Fos', 'protein'], ['Jun', 'protein']]
Model	*Prediction*
OLMo_{NER}	[['virus-exposed cells', 'cell line']]
$\text{OLMo}_{\text{NER+SC}}$	[['virus-exposed cells', 'cell line'], ['Fos', 'protein'], ['Jun', 'protein']]
$\text{OLMo}_{\text{NER+SC}}$ + Self-correct	[['Fos', 'protein'], ['Jun', 'protein']]

Table 9: Example of a Spurious error correction from the BioNLP 2004 dataset. The correct and incorrect entities are highlighted in the table. The format of the ground truth and prediction is as follows: [['entity', 'type'], . . .].

Error type	bc5cdr	bionlp	conll	fin	ontonotes	wikiann	wnut	scier	mean
None	0.951	0.912	0.976	0.989	0.942	0.918	0.782	0.884	0.919
OE	0.562	0.235	0.000	0.000	0.359	0.222	0.256	0.463	0.262
UE	0.376	0.525	0.478	0.000	0.594	0.697	0.140	0.362	0.396
Type	0.000	0.317	0.328	0.000	0.327	0.376	0.344	0.345	0.255
Span+Type	0.333	0.330	0.143	0.000	0.656	0.473	0.235	0.157	0.291
Spurious	0.612	0.386	0.359	0.857	0.169	0.925	0.485	0.432	0.528
Micro-F1	0.903	0.832	0.943	0.968	0.883	0.859	0.629	0.778	0.850
Macro-F1	0.534	0.505	0.461	0.402	0.562	0.639	0.410	0.489	0.500

Table 10: Detailed self-verification performance of $\text{OLMo}_{\text{NER+SC}}$ (♣) on eight NER datasets.

Error type	bc5cdr	bionlp	conll	fin	ontonotes	wikiann	wnut	scier	mean
None	0.956	0.905	0.978	1.000	0.951	0.923	0.787	0.921	0.928
OE	0.548	0.146	0.000	0.000	0.488	0.222	0.385	0.430	0.277
UE	0.494	0.539	0.773	0.000	0.594	0.650	0.133	0.437	0.453
Type	0.500	0.373	0.395	0.000	0.357	0.418	0.348	0.422	0.352
Span+Type	0.444	0.409	0.476	0.000	0.746	0.367	0.318	0.353	0.389
Spurious	0.630	0.361	0.500	0.933	0.471	0.922	0.386	0.529	0.591
Micro-F1	0.912	0.817	0.954	0.989	0.901	0.863	0.634	0.837	0.863
Macro-F1	0.640	0.507	0.582	0.418	0.644	0.624	0.427	0.561	0.550

Table 11: Detailed self-verification performance of $\text{Qwen}_{\text{NER+SC}}$ (♣) on eight NER datasets.

D Self-verification Performance

Tables 10 and 11 present the detailed SV performance of $\text{OLMo}_{\text{NER+SC}}$ (♣) and $\text{Qwen}_{\text{NER+SC}}$ (♣), complementing Table 2.

E Prompts

We provide comprehensive prompts for each task as listed in Table 12 and 13. As described in § 3, each prompt consists of the following components: (1) task instruction I, (2) label candidates L, and (3) label descriptions D. These prompts were used during training and inference phases.

Prompts for NER on SciER (Zhang et al., 2024a)	
Task instruction I_{NER}	Please list all the entity words associated with the category in the given text. Output format should be {'ner': [['entity', 'type'], ['entity', 'type'],...]}
Label candidates L_{NER}	The entity type candidates are `dataset`, `task`, and `method`.
Label description D_{NER}	The descriptions of each label are as follows: `dataset`: A realistic collection of data that is used for training, validating or testing the algorithms. These datasets can consist of various forms of data such as text, images, videos, or structured data. For example, MNIST, COCO, AGNews, IMDb, etc. `task`: A task in machine learning refers to the specific problem or type of problem that a ML/AI model is designed to solve. Tasks can be broad, like classification, regression, or clustering, or they can be very specific, such as Pedestrian Detection, Autonomous Driving, Sentiment Analysis, Named Entity Recognition and Relation Extraction. `method`: A method entity refers to the approach, algorithm, or technique used to solve a specific task/problem. Methods encompass the computational algorithms, model architectures, and the training procedures that are employed to make predictions or decisions based on data. For example, Convolutional Neural Networks (CNNs), Dropout, data augmentation, recurrent neural networks, etc.
Prompts for SE on SciER (Zhang et al., 2024a)	
Task instruction I_{SE}	Please list all entity words in the text that fit the target label described below. The target label to extract entities is `[PLACEHOLDER]`. Output format should be {'se': [['entity', 'type'], ['entity', 'type'],...]}
Label candidates L_{NER}	Same as NER (L_{NER}).
Label description D_{NER}	Same as NER (D_{NER}).
Prompts for ET on SciER (Zhang et al., 2024a)	
Task instruction I_{ET}	Given options, please list all entity types of all the listed entity words. Output format should be {'et': [['entity', 'type'], ['entity', 'type'],...]}
Label candidates L_{NER}	Same as NER (L_{NER}).
Label description D_{NER}	Same as NER (D_{NER}).

Table 12: Prompts for the NER, SE, and ET tasks on SciER. The variable `[PLACEHOLDER]` in I_{SE} is replaced with the target label l_i. Label candidates and their descriptions in the SE and ET tasks are omitted, as these prompts are same as NER.

Prompts for SV on SciER (Zhang et al., 2024a)	
Task instruction I_{SV}	Given the input text, label candidates, and their descriptions, please check whether or not the named entity which is given as the input is correct. Additionally, if the named entity is incorrect, please classify the error type of the incorrect entity from the given options. Note that the output of the error type should be `None` if the entity is correct. The output should be in the JSON format like this: {'sv': (`CORRECTNESS`, `ERROR_TYPE`) } - `CORRECTNESS`: `correct` or `incorrect` - `ERROR_TYPE`: `Over-extraction`, `Under-extraction`, `Type`, `Span+Type`, `Spurious`, or `None`
Label description D_{NER}	Same as NER (D_{NER}).
Label candidates L_{SV}	The correctness candidates are `correct` and `incorrect`. The error type candidates are `Over-extraction`, `Under-extraction`, `Type`, `Span+Type`, `Spurious`, and `None`.
Label description D_{SV}	The definitions of each error type are as follows: `Over-extraction`: The type of input entity is correct but the input entity includes redundant tokens compared with the gold entity so that several tokens should be discarded. `Under-extraction`: The type of input entity is correct but the input entity lacks several tokens that are included in the gold entity so that several tokens from the input text should be added. `Type`: The input entity is exactly matched to the gold entity but the type of the input entity is incorrect compared with the type of the gold entity. `Span+Type`: Both the input entity and its type is incorrect. `Spurious`: A completely incorrect entity that gold annotation does not exist.
Prompts for SR on SciER (Zhang et al., 2024a)	
Task instruction I_{SR}	Given the entity extracted from the input text, please revise the entity correctly while carefully comparing the entity to the label descriptions. Note that the output should be the same for the input entity if the entity was correct (i.e., no need to revise). The output should be in the JSON format like this: {'sr': ('entity', 'type')}
Label candidates L_{NER}	Same as NER (L_{NER}).
Label description D_{NER}	Same as NER (D_{NER}).

Table 13: Prompts for the SV and SR tasks on SciER. Label candidates and their descriptions in the SV and SR tasks are omitted, as these prompts are same as NER.