GuideNER: Annotation Guidelines Are Better than Examples for In-Context Named Entity Recognition

Shizhou Huang[1], **Bo Xu**[2], **Yang Yu**[1], **Changqun Li**[1], **Xin Alex Lin**[1, 3*]

[1]School of Computer Science and Technology, East China Normal University, Shanghai, China
[2]School of Computer Science and Technology, Donghua University, Shanghai, China
[3]Shanghai Key Laboratory of Multidimensional Information Processing
huangshizhou@ica.stc.sh.cn, xubo@dhu.edu.cn, {52205901014, 52215901009}@stu.ecnu.edu.cn, xlin@cs.ecnu.edu.cn

Abstract

Large language models (LLMs) demonstrate impressive performance on downstream tasks through in-context learning (ICL). However, there is a significant gap between their performance in Named Entity Recognition (NER) and in fine-tuning methods. We believe this discrepancy is due to inconsistencies in labeling definitions in NER. In addition, recent research indicates that LLMs do not learn the specific input-label mappings from the demonstrations. Therefore, we argue that using examples to implicitly capture the mapping between inputs and labels in in-context learning is not suitable for NER. Instead, it requires explicitly informing the model of the range of entities contained in the labels, such as annotation guidelines. In this paper, we propose GuideNER, which uses LLMs to summarize concise annotation guidelines as contextual information in ICL. We have conducted experiments on widely used NER datasets, and the experimental results indicate that our method can consistently and significantly outperform state-of-the-art methods, while using shorter prompts. Especially on the GENIA dataset, our model outperforms the previous state-of-the-art model by 12.63 F1 scores.

Code — https://github.com/JinFish/GuideNER

Introduction

Large language models (LLMs) demonstrate impressive performance on downstream tasks through in-context learning (ICL), which allows them to generalize to unseen cases using only a few input-label pairs (examples) without requiring further fine-tuning (Liu et al. 2022; Min et al. 2022). However, there is a significant gap between their performance in Named Entity Recognition (NER) and in fine-tuning methods (Li et al. 2023). We believe this discrepancy is due to inconsistencies in labeling definitions in NER. For example, in ACE05 (Walker et al. 2006), pronouns (such as "he") are annotated as `Person`, whereas in CoNLL03 (Tjong Kim Sang and De Meulder 2003), they are not.

Intuitively, the input-label pairs provided in ICL allow the LLM to accurately understand the relationship between the inputs and the labels, and thus the range of entities corresponding to the labels. However, (Min et al. 2022) has shown that even if labels in the label space (the set of labels for this task) are randomly assigned to inputs, there is almost no difference in model performance compared to using the correct input-label pairs. This indicates that LLMs do not learn the specific input-label mappings from the examples but rather grasp the general definition of the task itself.

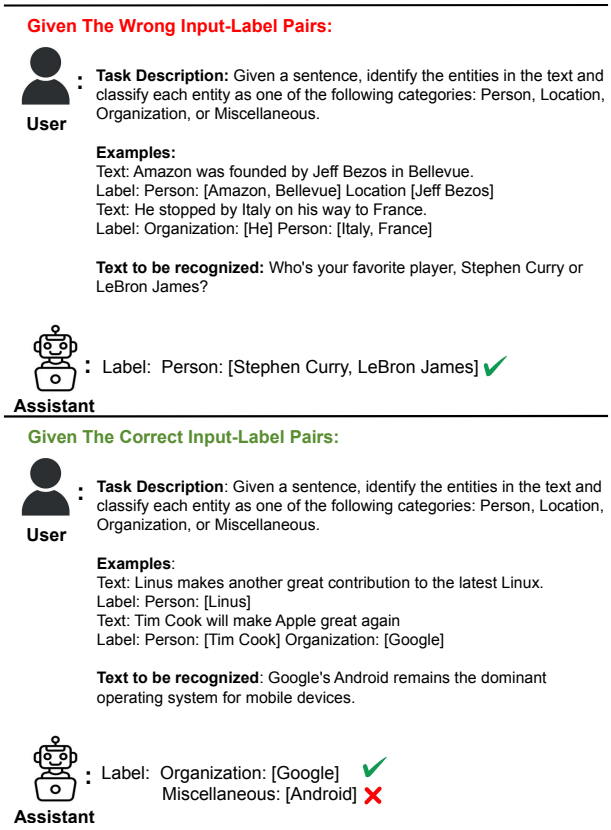

Figure 1: Two examples of LLM for NER within ICL.

The above phenomenon is also observed in the current advanced LLMs, including GPT-4 and Gemini. As shown in Figure 1, even with incorrect input label pairs (e.g., labeling Amazon as Person), the LLM can still correctly classify `Stephen Curry` and `LeBron James` as `Person`. Furthermore, even when provided with correct

*Corresponding Author.

input-label pairs and given that the operating system `Linux` is not categorized as any entity type in the examples, the LLM still categorizes the operating system `Android` as `Miscellaneous`. This further illustrates that the LLM does not explicitly learn the mapping relations between inputs and does not capture the range of entities corresponding to each label.

Therefore, we argue that using examples to implicitly capture the mapping between inputs and labels in ICL is not suitable for NER. Instead, it is necessary to explicitly inform the model of the range of entities represented in the labels, such as through annotation guidelines. Considering that datasets do not always provide annotation guidelines or that such guidelines can be excessively lengthy, e.g., the ACE entity annotation guidelines (Linguistic Data Consortium 2008) exceed 70 pages. In this paper, we propose GuideNER, which uses LLMs to summarize concise annotation guides as contextual information in ICL.

Specifically, we first use the LLM to infer patterns from each input-label pair in the training set, and then verify these patterns for errors or ambiguities. Next, we summarize the accurate patterns to create concise and comprehensive annotation guidelines. Finally, the LLM uses these summarized guidelines to make predictions. Essentially, our approach is similar to other ICL methods in that we utilize the entire training set. However, unlike other methods that dynamically retrieve examples related to the input from the training set for contextual information, we condense the entire training set into static annotation guidelines. We also use examples to ensure that the model returns a consistent format.

Our main contributions are summarized as follows:

- We introduce a novel approach that leverages LLMs to summarize dataset annotation guidelines, addressing inconsistencies in label definitions and enhancing NER performance within ICL.
- We propose a new ICL paradigm for NER, which does not require dynamic example retrieval for examples from the training set. Instead, our approach uses a fixed static prompt created by condensing the entire training set.
- We have conducted experiments on widely used NER datasets, and the experimental results indicate that our method can consistently and significantly outperform state-of-the-art methods, while using shorter prompts. Especially on the GENIA dataset, our model outperforms the previous state-of-the-art model by 12.63 F1 scores.

Related Work

Generative Named Entity Recognition

Named Entity Recognition (NER) is a crucial task in natural language processing, aiming to identify entities from plain text and classify them into predefined categories. Traditionally, NER has been framed as a sequence labeling task (Chiu and Nichols 2016; Devlin et al. 2019). However, with the notable advancements in generative large language models in text understanding and generation, some works (Cui et al. 2021; Paolini et al. 2021; Lu et al. 2022; Wang et al. 2023b) have redefined NER as a generation task, leveraging generative models to convert plain text into structured information. For instance, (Lu et al. 2022) introduces a unified text-to-structure generation model by training a T5 (Raffel et al. 2020) on a large, heterogeneous dataset sourced from the web, enabling adaptive generation of the target structure. (Wang et al. 2023b) employs instruction-based fine-tuning of a FlanT5 (Chung et al. 2024) for the NER task, facilitating the transformation from text to structured output.

Additionally, code, as a formal language, is particularly suited for structural prediction. Some approaches (Li et al. 2023; Sainz et al. 2024) model the NER task as a code generation task, employing various programming paradigms to achieve structured prediction, where (Sainz et al. 2024) adds descriptions of the labels as guidelines to the prompt and performs instruction tuning, enabling the model to learn and follow the guidelines.

In-Context Learning

Since the introduction of GPT-3 (Brown et al. 2020), its impressive In-Context Learning (ICL) capabilities have garnered significant attention. GPT-3 can generalize to unseen cases by providing only a few contextual examples without requiring further fine-tuning. Current ICL research focuses on the selection of examples used to provide contextual information.

In the NER, (Li et al. 2023) uses a method of randomly selecting a few examples from the training set, which is a straightforward approach. (Guo et al. 2023) employs the KNN method to select examples from the training set that are semantically close to the input examples at the sentence level. Given that NER is a token-level task, (Wang et al. 2023a) further proposes retrieving similar examples based on token-level representations.

In relation extraction and event extraction, (Pang et al. 2023) enables the model to extract rules for each instance and dynamically retrieve them during inference. However, due to semantic differences between rules and inputs, it is difficult to ensure that the retrieved rules align with the model's input. This challenge is amplified in named entity recognition tasks, where diverse input patterns further constrain the effectiveness of dynamic rule retrieval.

In this paper, we consider that the key to ICL is to provide contextual information rather than to provide examples, we propose to use the annotation guidelines as contextual information to explicitly provide clearer and effective contextual information.

Overview

In this section, we first formulate the Named Entity Recognition task and then give an overview of our method: GuideNER.

Task Formulation

Given a sentence with n tokens $X = (x_1, x_2, \ldots, x_n)$, the task of Named Entity Recognition is to extract a set of (e, t) pairs form X, where e is the entity name (e.g., Stephen Curry) and t is the corresponding entity type (e.g., Person). The entity name is a sequence of tokens from X and the entity type belongs to a pre-defined entity type set Y.

Overall Framework

Our overall framework of GuideNER is shown in Figure 2, which consists of three main steps: (1) **Summarize Patterns**: This step involves summarizing patterns from the input-label pairs in the training set; (2) **Self-Verify**: This step involves reviewing the inputs and patterns to verify if they yield the correct labels, filtering out patterns that do not predict the correct labels; and (3) **Aggregate Rules**: This step condenses all the rules into concise annotation guidelines and uses these guidelines to enable the LLM to make predictions.

Method

Step 1: Summarize Patterns

In this step, we focus on summarizing patterns and rules from each input-label pair in the training set. The training set is defined as $D = \{(X_i, Y_i)\}_{i=1}^{M}$, where M represents the total number of samples in the training set. Each sample consists of an input sentence X_i and a corresponding set of labels Y_i, where $Y_i = \{(e_j, t_j)\}_{j=1}^{N}$. Here, N denotes the number of labels, and each label is a tuple consisting of an entity name e_j and its type t_j.

The summarization process is guided by a `Summary Prompt` $\mathcal{P}_1$, which consists of three text segments:

1. **Task description**: "Task: Summarize the generic rules for each named entity category in the named entity recognition task. The order of the summarized rules should strictly correspond to the order of the annotations.".
2. **Examples**: This segment begins with "Examples:" followed by examples selected from the training set that includes three components: "Input Text: {*input text in example*}", "Annotations: {*labels in example*}", and "Output: {*handwritten patterns*}". The *handwritten patterns* are in JSON format, e.g., {*"location": ["city", "capital"]*}, where the keys are entity categories and the values are a list of patterns that match the category. Each combination of a key and a pattern in the list is treated as a rule. For example, *("location", "city")* represents one rule.
3. **Input-Label Pairs**: This segment begins with "Summarize for:" followed by the input X_i and labels Y_i from D to "Input text" and "Annotations", respectively. It ends with "Output:", which prompts for the model summarization rules.

Next, we fill X_i and Y_i into the $\mathcal{P}_1$ to create a complete prompt and enter it into LLM to summarize the rules.

$$Pa_i = \{r_j\}_{j=1}^{N} = LLM(\mathcal{P}_1(X_i, Y_i)) \quad (1)$$

where LLM represents the Large Language Model and Pa_i is the patterns corresponding (X_i, Y_i), where each rule r_j is a combination of a key and a pattern from the list, as mentioned above.

Step 2: Self-Verify

In this step, we address the potential issues of incorrect or ambiguous patterns summarized by the model, which could lead to error propagation. The goal is to verify that the summarized rules accurately predict the corresponding entities and their types.

Since the model summarized rules in the order of the labels in the first step, each rule corresponds to a label (e_j, t_j). This results in a set of pairs $\{(e_j, t_j), r_j\}_{j=1}^{N}$. To validate these rules, we use a `Verify Prompt` $\mathcal{P}_2$, which is also consists of three text segments:

1. **Task description**: "Task: Please identify Entity from the given text and patterns.".
2. **Examples**: This segment begins with "Examples:" followed by examples selected from the training set that includes three components: "Input Text: {*input text in example*}", "Patterns: {*handwritten patterns*}", and "Output: {*labels in example*}.
3. **Input-Pattern Pairs**: This segment begins with "Identify:" followed by the input X_i and patterns Pa_i to "Input text" and "Patterns", respectively. It ends with "Output:", which prompts for the model output predicted results based input and patterns.

The prompt is then input into the LLM to predict entities and types as follows:

$$\{(\hat{e}_j, \hat{t}_j)\}_{i=1}^{Z} = LLM(\mathcal{P}_2(X_i, Pa_i)) \quad (2)$$

where $\{(\hat{e}_i, \hat{t}_i)\}_{i=1}^{Z}$ are the predicted entities and types corresponding to the text and summarized rules, Z is the number of predicted labels.

We then compare the predicted entities and types $\{(\hat{e}_i, \hat{t}_i)\}_{i=1}^{Z}$ against the correct labels $\{(e_j, t_j)\}_{j=1}^{N}$. If the predicted $(\hat{e}_i, \hat{t}_i)$ matches (e_j, t_j), the corresponding rule r_j is considered accurate; otherwise, it is deemed incorrect. This validation process further ensures that only accurate rules are used further.

Step 3: Aggregate Rules

In this step, we focus on aggregating the validated rules into concise annotation guidelines that allow the model to accurately predict entities and their types.

For each correctly verified rule r_j, we maintain a record of its frequency of occurrence, denoted as f_j. The frequency f_j is incremented by one each time the rule is validated as correct. This results in a set of pairs $\{(r_j, f_j)\}_{j=1}^{U}$, where each r_j represents a rule and f_j its corresponding frequency, and U is the number of all rules.

Next, we process this set to ensure that only the most general rules are retained:

1. **Frequency Filtering**: For each entity type, we examine all rules associated with that type. If different entity types share identical rules, we retain the rule with the higher frequency and discard those with lower frequencies.
2. **Top k Rules Selection**: For each entity type, we select the top k rules based on their frequency. This selection forms the **Guidelines** for that entity type, ensuring that the most frequent and relevant rules are prioritized, preventing too many low-frequency rules from affecting the model's predictions.

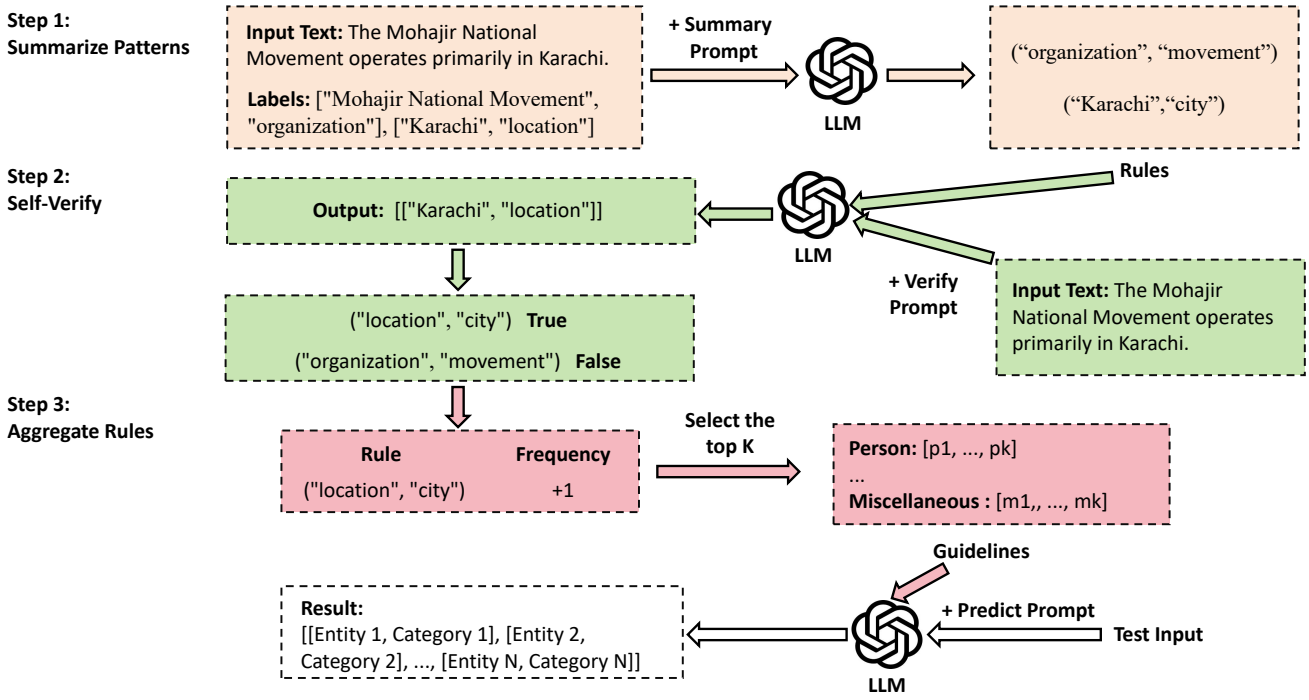

Figure 2: Overall framework of GuideNER.

To apply these Guidelines in practice, we use a *Predict Prompt* $\mathcal{P}_3$, which contains four segments:

- **Task description**: "Task: Please identify Entity from the given text and guidelines. The guidelines provide an entity category followed by a list of rules.".
- **Guidelines**: The content of the guide obtained above.
- **examples**: This segment begins with "Examples:" followed by examples selected from the training set that includes three components: "Input Text: {*input text in example*}", "Instructions for using the guidelines: *Given the Input Text and Guidelines, the Output is:*" and "{*labels in example*}".
- **Input that require inference**: This segment begins with "Identify Entities for:" followed by the input X_i to "Input text" and "Instructions for using the guidelines".

Next, we fill the input text $\hat{X}$ to be inferred into the $\mathcal{P}_3$ to create a complete prompt and enter it into LLM for entity recognition:

$$\{(\hat{e}_j, \hat{t}_j)\}_{j=1}^{O} = LLM(\mathcal{P}_3(\hat{X})) \quad (3)$$

where $\{(\hat{e}_j, \hat{t}_j)\}_{j=1}^{O}$ are the predicted entities and types derived from the input text $\hat{X}$, O is the number of predicted labels. **Note**: The cost of extracting annotation guidelines is a one-time expense, and our method achieves lower inference costs due to shorter prompt lengths. Consequently, our approach offers a cost advantage when conducting a large number of inference tasks.

Experiments

In this section, we conduct various experiments to comprehensively evaluate the performance of our method GuideNER on four widely used NER datasets. Following many previous works (Lu et al. 2022; Li et al. 2023; Sainz et al. 2024), we use the F1 score (F1) as evaluation metrics.

Datasets

We conduct experiments on four widely used NER datasets: (1) CoNLL03 (Tjong Kim Sang and De Meulder 2003) is a NER dataset for the news domain and contains four types of named entities: Person, Organization, Location and Miscellaneous; (2) ACE04 (Doddington et al. 2004) and (3) ACE05 (Walker et al. 2006) are a NER dataset for the news domain and contain seven types of entities: Person, Organization, Vehicle, Geopolitical entity, Location, Weapon and Facility; (4) GENIA is a nested NER dataset in the molecular biology domain containing five entity types: DNA, RNA, Protein, Cell_type, Cell_line.

Experiments Setting

We ran all experiments using the deep learning framework PyTorch 2.3.1 and vllm 0.5.3 on an Ubuntu 22.04.2 system equipped with NVIDIA 4090 GPUs.

We adopt Llama3-8B-Instruct and text-davinci-003 as the LLMs used in the experiments. By default, the same LLM is used in the three steps of this paper. We set the maximum output length to 256 tokens. For the models, Llama3-8B-Instruct and text-davinci-003, we configure the top k values to $k = 10$ and $k = 20$, respectively. In experiments, we use

Model	Backbone	CoNLL03	ACE04	ACE05	GENIA	Prompt Length
Llama3-8B-Instruct	llama3-8B-Instruct	47.67	34.67	36.49	31.79	1,012
text-davinci-002	text-davinci-002	68.84	45.51	48.93	-	786
text-davinci-003	text-davinci-003	72.36	47.78	51.29	46.28	873
CodeIE	code-davinci-002	82.32	55.29	54.82	-	1,172
Code4UIE	text-davinci-003	83.60	60.10	60.90	-	-
GPT-NER	text-davinci-003	85.90	62.52	60.02	52.13	4,475
C-ICL	codeLlama-34B	87.36	54.47	55.65	-	-
GuideNER (ours)	llama3-8B-Instruct	66.71	47.67	48.23	44.07	**506**
GuideNER (ours)	text-davinci-003	**90.27**†	**67.21**†	**68.58**†	**64.76**†	694

Table 1: The experiment performances on four NER datasets. Some methods are not reported for their prompt length, due to not disclosing their code or full prompts. The reason that the length of GPT-NER is much higher than that of the other methods is that it models NER as multiple binary categorization tasks, and each category requires a separate prompt. The marker † refers to significant test p-value < 0.05 when compared with C-ICL, GPT-NER and CodeIE.

the grid search to find the temperature within $[0.7, 0.8, 0.9]$, the top_p within $[0.85, 0.90, 0.95]$.

Baselines

We use the currently widely adopted ICL for NER as a baseline, which does not involve any fine-tuned model and utilizes the training set (retrieving examples from the training set), including:

- Llama3-8B-Instruct, text-davinci-002 (Li et al. 2023) and text-davinci-003 are natural language-based LLMs that obtain structured results by inputting instructions in textual form.
- CodeIE (Li et al. 2023) proposes to recast the structured output in the form of code instead of natural language and utilize generative LLMs of code to perform NER.
- Code4UIE (Guo et al. 2023) is a retrieval-augmented code generation framework, which formulates NER task as a Python instance code completion task.
- GPT-NER (Wang et al. 2023a) bridges the gap between the sequence labeling task and the text generation task by instructing the LLM to generate a labeled sequence by surrounding entities with special tokens. Since it uses a fine-tuned model, for a fair comparison, we report on its performance using a general retrieval model.
- C-ICL (Mo et al. 2024) proposes using both correct and incorrect sample constructions to create in-context learning demonstrations.

Main Results

Table 1 presents the performance of various models across four NER datasets. Firstly, we compare the performance between the LLMs, and we observe that model performance generally correlates with the model's inherent capabilities. For instance, text-davinci-003 consistently outperforms text-davinci-002, indicating that more advanced models deliver better results for this task.

Secondly, comparing models that use annotation guidelines to their counterparts that use the examples, such as GuideNER (llama3-8B-Instruct) versus Llama3-8B-Instruct. Specifically, GuideNER (llama3-8B-Instruct) surpasses the corresponding model llama3-8B-Instruct significantly and achieves comparable performance to text-davinci-002 (175B), demonstrating the effectiveness of annotation guidelines in improving NER performance. This improvement highlights the benefit of annotation guidelines over simple examples. Additionally, GuideNER (text-davinci-003) shows greater improvement over GuideNER (llama3-8B-Instruct) with the help of annotation guidelines, which we attribute to its superior ability to understand and apply complex instructions, allowing it to better utilize the annotation guidelines for NER.

Finally, our method achieves state-of-the-art performance on all four datasets. Notably, our model outperforms the previous state-of-the-art model, GPT-NER, by 12.63 F1 points on the GENIA dataset, which demonstrates that the use of annotation guidelines can provide more effective contextual information to help the model perform the NER task. In addition, we find that the prompt length of our method is significantly less than other methods, which is because current methods use a large number of examples to enhance model performance, e.g., CodeIE uses 25 examples on CoNLL03. Our method's efficiency in using annotation guidelines demonstrates that fewer tokens can achieve high performance, providing a clear advantage over methods that rely on extensive examples.

Ablation Study on Self-Verify

As shown in Table 2, the performance of both GuideNER (text-davinci-003) and GuideNER (llama3-8B-Instruct) decreases significantly after removing the self-validation step, indicating the effectiveness of the rules summarized by the validation model. Among them, GuideNER (llama3-8B-Instruct) experiences a more significant drop, as Llama3-8B-Instruct has weaker instruction comprehension and reasoning capabilities compared to text-davinci, making it more prone to generating incorrect rules that contaminate the final annotation guide.

Model	Backbone	CoNLL03	ACE04	ACE05	GENIA
GuideNER	llama3-8B-Instruct	66.71	47.67	48.23	44.07
GuideNER	text-davinci-003	**90.27**	**67.21**	**68.58**	**64.76**
GuideNER w/o Step2	llama3-8B-Instruct	60.12	39.98	41.11	39.92
GuideNER w/o Step2	text-davinci-003	87.55	64.76	64.67	62.80

Table 2: Ablation study on Self-Verify.

Summary Model	Inference Model	CONLL03	ACE04	ACE05	GENIA
-	llama3-8B-Instruct	47.67	34.67	36.49	31.79
-	text-davinci-003	72.36	47.78	51.29	46.28
llama3-8B-Instruct	llama3-8B-Instruct	66.71	47.67	48.23	44.07
llama3-8B-Instruct	text-davinci-003	81.24	56.76	58.91	47.27
text-davinci-003	llama3-8B-Instruct	70.73	49.11	53.08	46.27
text-davinci-003	text-davinci-003	**90.27**	**67.21**	**68.58**	**64.76**

Table 3: Comparison of performance using different LLMs in different steps, where the method of no summary model indicates the use of examples. text-davinci-003 still uses the top 20 rules and llama3-8B-Instruct uses the top 10 rules.

Evaluation of Performance with Different Models Across Steps

In the GuideNER framework, LLM is used in three distinct steps: the first two steps are dedicated to summarizing rules, while the third step focuses on inference. Different models can be used for summarization and inference, respectively.

Table 3 illustrates the performance of GuideNER when various models are employed during the summarization (first two steps) and inference (third step) processes. We find that regardless of whether the summary model and inference model are consistent, the method utilizing summarized rules always outperforms its corresponding example-based method, demonstrating the effectiveness of using annotation guidelines. In addition, we observe that using text-davinci-003 as the summary model leads to improved performance in the inference phase compared to using llama3-8B-Instruct. This suggests that a more advanced LLM for summarization significantly enhances the effectiveness of the inference model. Therefore, employing a stronger summarization model with a larger number of parameters, paired with an inference model with relatively fewer parameters, is a reasonable and effective strategy.

Evaluation of GuideNER with Few Rules

In the second step of the GuideNER, labels are predicted based on previously summarized patterns. The goal of this step is to verify the accuracy of the summarized patterns, but it also essentially utilizes rules as contextual information for NER.

As shown in Table 4, we present the performance of GuideNER performance in this second step, referred to as GuideNER (Step 2). Although GuideNER (Step 2) is tested on the training set, its input only includes text and patterns, and we believe its performance is still of reference value. Despite some errors in the patterns during Step 2, GuideNER (Step 2) shows a notable improvement because the model only receives patterns related to entity types, allowing it to effectively follow the rules to predict entities.

Therefore, we consider using only a few rules as contextual information rather than the entire annotation guidelines. We convert the rules into natural language form; for example, *("location", "city")* is translated into "the entity category for the city is location". We then use SimCSE (Gao, Yao, and Chen 2021) to compute the similarity between the input text and the rules, retrieving the top 5 rules as context, referred to as GuideNER (Retrieval). However, we find that the performance of GuideNER (Retrieval) is poor, even worse than using example-based methods. This is due to the significant semantic gap between the input text and the rules, which makes it challenging to retrieve the relevant rules through similarity calculations.

Overall, we believe that providing only relevant rules can better help the model to make predictions and can further reduce the length of the prompts. However, there are still challenges in realizing this with current semantic-based retrieval approaches. We believe this finding will be useful for future work.

Effect of Top-k Rules on GuideNER Performance

To explore the effect of different numbers of top-k rules on the model performance, we evaluate GuideNER (text-davinci-003) and GuideNER (llama3-8B-Instruct) using different k values and observe their performance across three datasets in Table 5.

For GuideNER (text-davinci-003), performance initially improves with increasing k, but declines as k continues to grow. This trend suggests that a higher number of rules does not always lead to better performance. We attribute this phenomenon to two main factors: (1) As k increases, the context length becomes longer, which may hinder the model's ability to follow instructions (annotation guidelines) effectively; (2) With more rules, the inclusion of less common

Model	Backbone	CoNLL03	ACE04	ACE05	GENIA
Llama3-8B-Instruct	llama3-8B-Instruct	47.67	34.67	36.49	31.79
text-davinci-003	text-davinci-003	72.36	47.78	51.29	46.28
GuideNER	llama3-8B-Instruct	66.71	47.67	48.23	44.07
GuideNER	text-davinci-003	90.27	67.21	68.58	64.76
GuideNER (Step 2)	llama3-8B-Instruct	82.71	59.20	61.20	57.61
GuideNER (Step 2)	text-davinci-003	**93.44**	**80.21**	**82.39**	**76.42**
GuideNER (Retrieval)	llama3-8B-Instruct	44.21	32.16	35.78	30.69
GuideNER (Retrieval)	text-davinci-003	70.52	44.21	46.70	43.56

Table 4: Performance comparison under different rule configurations.

Model	Top-k	ACE04	ACE05	GENIA
llama3-8B	0	34.67	36.49	31.79
llama3-8B	5	45.21	47.51	42.13
llama3-8B	10	**47.67**	**48.23**	44.07
llama3-8B	20	46.11	47.01	**44.32**
llama3-8B	30	45.21	46.54	43.28
text-davinci	0	47.78	51.29	46.28
text-davinci	5	58.46	60.72	54.76
text-davinci	10	64.43	64.23	58.42
text-davinci	20	**67.21**	**68.58**	64.76
text-davinci	30	66.21	67.77	**65.12**

Table 5: Effect of top-k rules on GuideNER performance, where k=0 indicates the use of examples to provide contextual information.

Model	Amount	ACE04	ACE05	GENIA
llama3-8B	0%	40.67	39.49	35.79
llama3-8B	20%	43.21	43.51	42.13
llama3-8B	30%	45.67	45.23	43.07
llama3-8B	40%	46.11	46.01	43.32
llama3-8B	70%	46.51	47.54	**48.23**
llama3-8B	100%	**47.67**	**48.23**	44.07
text-davinci	0%	53.21	55.29	55.28
text-davinci	20%	58.76	63.72	62.72
text-davinci	30%	62.43	65.23	63.92
text-davinci	40%	64.98	67.21	64.34
text-davinci	70%	67.01	68.23	64.61
text-davinci	100%	**67.21**	**68.58**	**64.76**

Table 6: Effect of amount of training data on GuideNER performance.

rules might dilute the effectiveness of more frequent rules, as the model may treat all rules with equal importance.

Similarly, this phenomenon is also observed in GuideNER (llama3-8B-Instruct), but with the difference that its performance declines when k exceeds 10. We attribute this decline to its comparatively weaker ability to process and adhere to a larger number of annotation guidelines, as compared to GuideNER (text-davinci-003), which can handle a greater number of annotation guidelines more effectively.

Effect of Amount of Training Data on GuideNER Performance

As shown in Table 6, we demonstrate the impact of using different amounts of the dataset on the performance of GuideNER. Our results for both GuideNER (text-davinci-003) and GuideNER (llama3-8B-Instruct) indicate that when the training set usage reaches 50% to 70%, the performance is comparable to that achieved with the full dataset. This is because we only select the top-k rules, and when the inductive process covers more than half of the training data, the relative order of these top-k rules generally remains stable. This suggests that our method demonstrates the potential and efficiency, which may be particularly beneficial in more constrained environments. Moreover, it also suggests that summarizing rules does not require using the entire dataset and can stop once the top-k rules converge.

Conclusion

In this paper, we present GuideNER. Unlike methods that dynamically retrieve examples from the training set, our approach condenses training data into annotation guidelines, explicitly providing contextual information to LLMs to enhance predictions. Extensive experiments demonstrate that our method significantly outperforms previous state-of-the-art approaches with shorter prompts, highlighting its effectiveness in delivering contextual information compared to example-based methods. Furthermore, we conducted a detailed analysis of GuideNER, validating the effectiveness and limitations of dynamic rule retrieval, offering insights for future research.

Acknowledgments

This work is supported by National Science and Technology Major Project (2021ZD0111000/2021ZD0111004), the Science and Technology Commission of Shanghai Municipality Grant (No. 21511100101, 22511105901,22DZ2229004), the Fundamental Research Funds for the Central Universities 2232023D-19, the Open Research Fund of Key Laboratory of Advanced Theory and Application in Statistics and

Data Science (East China Normal University), Ministry of Education. Xin Alex Lin is the corresponding author. Xin Alex Lin is also a member of Key Laboratory of Advanced Theory and Application in Statistics and Data Science (East China Normal University), Ministry of Education.

References

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 1877–1901.

Chiu, J. P.; and Nichols, E. 2016. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4: 357–370.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70): 1–53.

Cui, L.; Wu, Y.; Liu, J.; Yang, S.; and Zhang, Y. 2021. Template-Based Named Entity Recognition Using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1835–1845.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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)*, 4171–4186.

Doddington, G. R.; Mitchell, A.; Przybocki, M.; Ramshaw, L.; Strassel, S.; and Weischedel, R. 2004. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6894–6910.

Guo, Y.; Li, Z.; Jin, X.; Liu, Y.; Zeng, Y.; Liu, W.; Li, X.; Yang, P.; Bai, L.; Guo, J.; et al. 2023. Retrieval-augmented code generation for universal information extraction. *arXiv preprint arXiv:2311.02962*.

Li, P.; Sun, T.; Tang, Q.; Yan, H.; Wu, Y.; Huang, X.-J.; and Qiu, X. 2023. CodeIE: Large Code Generation Models are Better Few-Shot Information Extractors. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15339–15353.

Linguistic Data Consortium. 2008. *ACE (Automatic Content Extraction) English Annotation Guidelines for Entities*.

Liu, J.; Shen, D.; Zhang, Y.; Dolan, W. B.; Carin, L.; and Chen, W. 2022. What Makes Good In-Context Examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, 100–114.

Lu, Y.; Liu, Q.; Dai, D.; Xiao, X.; Lin, H.; Han, X.; Sun, L.; and Wu, H. 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)*, 5755–5772.

Min, S.; Lyu, X.; Holtzman, A.; Artetxe, M.; Lewis, M.; Hajishirzi, H.; and Zettlemoyer, L. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 11048–11064. Association for Computational Linguistics.

Mo, Y.; Yang, J.; Liu, J.; Zhang, S.; Wang, J.; and Li, Z. 2024. C-ICL: Contrastive In-context Learning for Information Extraction. *arXiv e-prints*, arXiv–2402.

Pang, C.; Cao, Y.; Ding, Q.; and Luo, P. 2023. Guideline Learning for In-Context Information Extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 15372–15389.

Paolini, G.; Athiwaratkun, B.; Krone, J.; Jie, M.; Achille, A.; Anubhai, R.; dos Santos, C. N.; Xiang, B.; Soatto, S.; et al. 2021. Structured prediction as translation between augmented natural languages. In *ICLR 2021-9th International Conference on Learning Representations*, 1–26. International Conference on Learning Representations, ICLR.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Sainz, O.; García-Ferrero, I.; Agerri, R.; de Lacalle, O. L.; Rigau, G.; and Agirre, E. 2024. GoLLIE: Annotation Guidelines improve Zero-Shot Information-Extraction. In *The Twelfth International Conference on Learning Representations*.

Tjong Kim Sang, E. F.; and De Meulder, F. 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*, 142–147.

Walker, C.; Strassel, S.; Medero, J.; and Maeda, K. 2006. ACE 2005 Multilingual Training Corpus. Linguistic Data Consortium, Philadelphia.

Wang, S.; Sun, X.; Li, X.; Ouyang, R.; Wu, F.; Zhang, T.; Li, J.; and Wang, G. 2023a. Gpt-ner: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*.

Wang, X.; Zhou, W.; Zu, C.; Xia, H.; Chen, T.; Zhang, Y.; Zheng, R.; Ye, J.; Zhang, Q.; Gui, T.; et al. 2023b. InstructUIE: Multi-task Instruction Tuning for Unified Information Extraction. *arXiv preprint arXiv:2304.08085*.