EvoPrompt: Evolving Prompts for Enhanced Zero-Shot Named Entity Recognition with Large Language Models

Zeliang Tong[♠1,2] , **Zhuojun Ding**[♠1,2] and **Wei Wei** [✉1,2]
[1]Cognitive Computing and Intelligent Information Processing (CCIIP) Laboratory,
School of Computer Science and Technology, Huazhong University of Science and Technology
[2]Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL)
tongzeliang744@gmail.com, {dingzj, weiw}@hust.edu.cn

Abstract

Large language models (LLMs) possess extensive prior knowledge and powerful in-context learning (ICL) capabilities, presenting significant opportunities for low-resource tasks. Though effective, several key issues still have not been well-addressed when focusing on zero-shot named entity recognition (NER), including the *misalignment between model and human definitions of entity types*, and *confusion of similar types*. This paper proposes an Evolving Prompts framework that guides the model to better address these issues through continuous prompt refinement. Specifically, we leverage the model to summarize the definition of each entity type and the distinctions between similar types (*i.e.*, entity type guidelines). An iterative process is introduced to continually adjust and improve these guidelines. Additionally, since high-quality demonstrations are crucial for effective learning yet challenging to obtain in zero-shot scenarios, we design a strategy motivated by self-consistency and prototype learning to extract reliable and diverse pseudo samples from the model's predictions. Experiments on four benchmarks demonstrate the effectiveness of our framework, showing consistent performance improvements. Code is available at https://github.com/tongzeliang/EvoPrompt.

1 Introduction

Named Entity Recognition (NER) aims to identify entities belonging to predefined types in texts, which is a fundamental information extraction task (Wei et al., 2021; Gu et al., 2022; Fan et al., 2024; Liu et al., 2023, 2024). While neural networks perform well in supervised settings, practical scenarios often lack annotated data. This has led to increased attention on NER in low-resource scenarios (Chen et al., 2022; Qu et al., 2023a). Early methods include distant supervision (Qu et al., 2023b), transfer learning (Jia et al., 2019; Ding et al., 2024), few/zero-shot learning (Wang et al., 2022), etc. Recent works show that large language models (LLMs) perform well across various tasks using only a few in-context learning (ICL) demonstrations, without parameter tuning (Wei et al., 2022; Yao et al., 2024). This has motivated research into designing prompting methods for low-resource NER using LLMs (Wei et al., 2023; Xie et al., 2024; Wang et al., 2023a).

In this work, we use LLMs for zero-shot NER, a strict scenario where no annotated data is available. We focus on two challenges: (1) the misalignment between the model's understanding of entity types and humans, and (2) the confusion of similar types, as shown in Figure 1. While a significant number of methods utilize high-quality ICL demonstra-

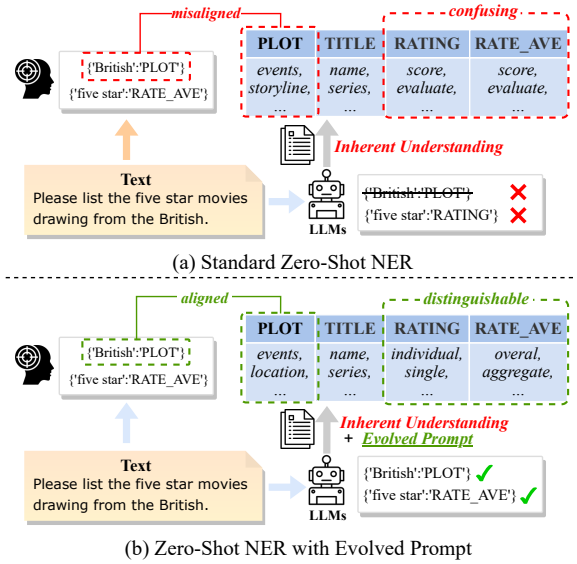


Figure 1: (a) Traditional prompting methods struggle with misalignment between model and human definitions, as well as confusion among similar types, especially in resource-scarce zero-shot settings. (b) We introduce the evolved prompt through iterative improvement to explicitly address these challenges.

♠ Contribute equal to this work.
✉ Corresponding author.

Proceedings of the 31st International Conference on Computational Linguistics, pages 5136–5153
January 19–24, 2025.

tions to improve performance, they do not explicitly tackle these challenges, which hinders further improvements. Furthermore, they often require an annotated sample library (e.g., retrieving similar examples from the library for each test example). Some studies (Li et al., 2023; Sainz et al., 2023) attempt to introduce guidelines for entity types to instruct the model on which entities to recognize explicitly. While this approach can be beneficial, their guidelines are often developed in isolation for each entity type, without accounting for the distinctions between similar types. Moreover, crafting guidelines that are comprehensive and easy for the model to understand is also a challenging task.

Considering these issues, we propose an Evolving Prompts framework, which offers more precise guidance for the model through continuously refined prompts. Specifically, we first prompt the model to generate initial entity predictions, referred to as pseudo-samples. Then, we leverage these pseudo-samples to construct entity type guidelines from two perspectives: (**i**) *Definitions of each entity type*. We leverage the model to summarize definitions for entity types based on the pseudo-samples. (**ii**) *Distinctions between similar types*. We identify entities that are assigned to multiple types across different contexts. When sufficient examples are found, we infer that the corresponding types are confusable and use the model to summarize their distinctions. In addition, we construct ICL demonstrations from these pseudo-samples. Finally, the guidelines and pseudo-ICL demonstrations prompt the model in subsequent iterations, progressively improving its predictions.

Notably, we rely solely on pseudo-samples to derive guidelines and pseudo-ICL demonstrations in zero-shot scenarios. Ensuring the reliability and diversity of these samples is crucial to avoid negative optimization. To address this, we design a pseudo-sample selection strategy inspired by self-consistency (Xie et al., 2024) and prototype learning. First, we filter samples based on consistency across predictions from previous iterations. Then, we apply clustering to identify prototypical samples for each entity type, using them as representatives. This approach mitigates noise in pseudo-samples and enhances the diversity of selected samples.

Our automated prompts refinement framework offers several advantages. Besides its convenience, it provides more comprehensive guidance: as iterations progress, the model encounters an increasing number of reliable and representative samples, resulting in progressively more precise guidelines and higher-quality pseudo-ICL demonstrations. Another potential benefit is that the self-summarized and refined guidelines are more adaptable to the model, similar to human learning, where mastery is achieved through personal understanding rather than solely relying on externally provided information.

To sum up, our contributions are as follows: **(1)** We propose an Evolving Prompts framework for zero-shot NER with LLMs, which provides dynamic, comprehensive, and model-adaptive guidance that alleviates the model's misunderstanding of entity types and confusion about similar types. **(2)** We design a pseudo-sample selection strategy to improve sample reliability and diversity for zero-shot settings. **(3)** Experimental results on four benchmark datasets demonstrate the effectiveness of our proposed framework.

2 Related Works

LLMs for Named Entity Recognition The exploration of applying LLMs to various tasks continues unabated, including IE tasks like NER (Ma et al., 2023; Zhang et al., 2024). Existing works can be mainly categorized into two types based on *whether model training is required*.

One category of works constructs extensive instruction-tuning data from open-source datasets and then performs instruction tuning on LLMs (Lu et al., 2023; Wang et al., 2023b; Zhou et al., 2023a; Xiao et al., 2023). Li et al. (2024) include pre-training, further enhancing the model's task adaptability. Despite their promising results, the significant demand for annotated data and computational resources hinders their widespread adoption.

Another category of works designs prompting methods. Some works consider improving task prompts, mainly focusing on *prompt format* and *ICL demonstrations*. For example, Li et al. (2023) employs code-style prompts and Code-LLMs to better meet the structured output needs of IE tasks. Xie et al. (2024) select reliable samples from model predictions as ICL demonstrations for zero-shot NER. Other works propose more intricate reasoning processes instead of a single interaction (Xie et al., 2023; Wei et al., 2023). They do not directly address the above-mentioned fundamental issues, limiting potential performance gains. Some studies incorporate crafted label guidelines into prompts (Sainz et al., 2023; Zhang et al., 2023). Although

beneficial, manually written guidelines often lack comprehensiveness and suitability. We propose an Evolving Prompts framework to better guide the model through continuously refined prompts.

Demonstration Retrieval in ICL Existing studies have demonstrated that high-quality demonstrations can significantly enhance the performance of LLMs in ICL (Rubin et al., 2022; Wang et al., 2023a; Cai et al., 2023). However, this becomes more challenging in low-resource settings. Xie et al. (2024) retrieves reliable self-annotated demonstration by self-consistency. Building on this, we propose a novel framework that further enhances reliability and considers the diversity of pseudo-labeled demonstrations.

3 Methodology

Preliminary Let x represent the input text and $\mathcal{L} = \{l_i\}_n$ denotes the entity label set, where n indicates the total number of labels. The objective of NER is to extract a set of entities $\mathcal{T} = \{(e_1, l_{e_1}), (e_2, l_{e_2}), \ldots, (e_N, l_{e_N})\}$, where e_i represents the entity span in x and $l_{e_i} \in \mathcal{L}$ is the corresponding entity type for e_i.

Our goal is to perform zero-shot NER with LLMs. Given a test sample x, we prompt the LLM $\mathcal{M}$ to generate corresponding predictions for x:

$$y \sim P_{\mathcal{M}}(y \mid T, S, x), \tag{1}$$

where T is the task description, and S denotes ICL demonstrations. For zero-shot settings, we initially have no golden annotated data to construct S.

Overal workflow As shown in Figure 2: **(1)** We start by prompting the model to generate entity predictions. **(2)** Then, we consolidate these predictions with any from previous steps to form a new set, denoted as pseudo-sample set $\mathcal{D}$. This set guides the model in generating guidelines for entity types (Section 3.2) and selecting pseudo-ICL demonstrations (Section 3.3). **(3)** Finally, we integrate these guidelines and demonstrations to create a new prompt, which guides the model in predicting entities in the next iteration.

3.1 Initial Self-Annotating

We start from an unlabeled corpus $\mathcal{U} = \{\mathbf{x}_i\}$. We first generate predictions with the model via zero-shot prompting:

$$y \sim P_{\mathcal{M}}(y \mid T_{ner}, x). \tag{2}$$

We instruct the model to generate predictions in JSON format to facilitate processing. Based on dataset introductions, we incorporate brief seed guidelines for entity types in the task description T_{ner} (Appendix E). Since these entity predictions contain noise, we refer to them as pseudo-samples.

3.2 Guidelines Self-Summarization

After obtaining the pseudo-samples, we leverage them to construct comprehensive and adaptive guidelines for entity types, thereby enhancing the model's understanding of label semantics. As shown in the second step of Figure 2, we employ the model to summarize two aspects of the guidelines: the definition of entity types and the distinctions between similar types.

Sample Library Construction Based on the pseudo-samples $\mathcal{D}$, we initially construct sample libraries to generate guidelines.

For the first part of the guidelines (i.e., the definition of entity types), we partition $\mathcal{D}$ by entity types. For an entity type $l \in \mathcal{L}$, we construct a *definition-related sample library* $\mathcal{D}_l = \{s_1, s_2, \ldots, s_n\}$. Each element s_i in $\mathcal{D}_l$ consists of a text x_i and all entities e_i^j of type l within the text: $s_i = \{x_i : [e_i^1, e_i^2, \ldots, e_i^m]\}$.

For the second part of the guidelines (i.e., the distinctions between similar types l_1 and l_2), we construct a *distinction-related library* $\mathcal{D}_{l_1 l_2}$. The element s_i of this library is defined as: $s_i = \{e_i,\ [x_1^{l_1}, x_2^{l_1}, \ldots, x_m^{l_1}],\ [x_1^{l_2}, x_2^{l_2}, \ldots, x_n^{l_2}]\}$, where e_i denotes an entity, $x_j^{l_1}$ denotes a text in which e_i appears as an entity of type l_1, and $x_j^{l_2}$ denotes a text where e_i appears as an entity of type l_2.

Representative Sample Selection After constructing sample libraries, we further select representative subsets for them, considering the reliability and diversity.

For the *definition-related* library $\mathcal{D}_l$, we select a set of prototypes (Snell et al., 2017) as representative samples $\mathcal{D}_l^{sub}$. Specifically, we perform clustering and choose the cluster centroids. For the element $s_i \in \mathcal{D}_l$, we consolidate its information (i.e., the text and entities) into a new text $s_i^{'}$. For example, $s_i^{'}$ ="*In the text* x_i, e_i^1 *and* e_i^2 *are entities of type* l". These new texts incorporate both contextual and entity information. Then we perform clustering on them and get K clustering centroids to construct $\mathcal{D}_l^{sub}$. By focusing on cluster centroids, we mitigate noise from boundary and outlier points,

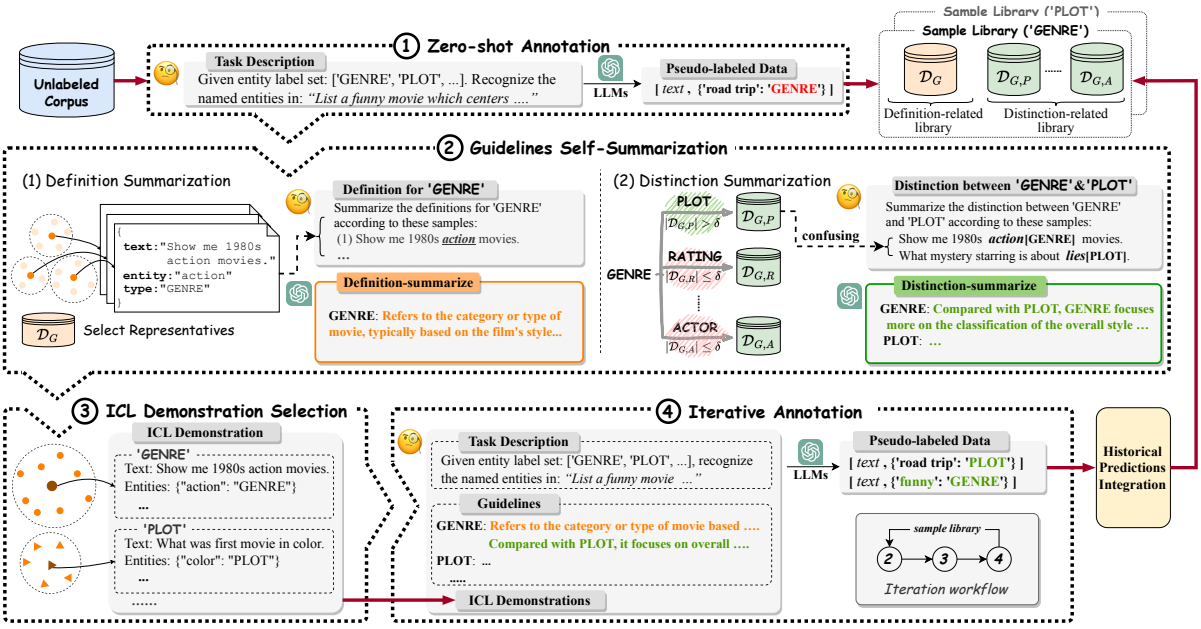


Figure 2: Overview of our framework. **(1)** We start with an unlabeled corpus and use zero-shot prompting to generate entity predictions (pseudo-labeled data). We then build two types of sample libraries using this data. **(2)** Reliable and representative samples are selected from these libraries to prompt the model to summarize guidelines (i.e. *definitions for each entity type* and *distinctions between similar types*. **(3)** We further select in-context learning (ICL) demonstrations from these sample libraries. **(4)** With the guidelines and ICL demonstrations, we prompt the model to generate new entity predictions. These predictions can then be used in a new iteration.

leading to more accurate guideline generation. Furthermore, these cluster centroids effectively capture the diverse features of the sample set, enabling the model to generate more comprehensive guidelines.

For the *distinction-related library* $\mathcal{D}_{l_1 l_2}$ of type l_1 and l_2, we apply the *majority rule* to construct a representative sample subset $\mathcal{D}^{sub}_{l_1 l_2}$. Consider the content of its elements: $s_i = \{e_i,\ [x_1^{l_1}, x_2^{l_1}, \ldots, x_m^{l_1}],\ [x_1^{l_2}, x_2^{l_2}, \ldots, x_n^{l_2}]\}$. If m is very small, it suggests that entity e_i rarely appears as type l_1. This could mean that e_i is an uncommon instance of type l_1, potentially limiting its significance. Moreover, with fewer instances, the risk of prediction error increases (i.e. e_i is not of type l_1). The same logic applies to n. Therefore, we only include a sample in $\mathcal{D}^{sub}_{l_1 l_2}$ when both m and n exceed a certain threshold.

Guidelines Generation We prompt the model to generate guidelines based on the representative samples $\mathcal{D}^{sub}_{l_i}$ and $\mathcal{D}^{sub}_{l_i l_j}$, where $l_i,\ l_j \in \mathcal{L}$.

To generate the first part of guidelines g_{i_1} (i.e. type definition) for the entity type l_i, we prompt the model with the task description T_g^1 and $\mathcal{D}^{sub}_{l_i}$:

$$g_{i_1} \sim P_{\mathcal{M}}(g_{i_1} | T_g^1, g_{i_1}^{pre}, \mathcal{D}^{sub}_{l_i}), \tag{3}$$

where $g_{i_1}^{pre}$ denotes the guidelines of type l_i from the previous iteration.

To generate the second part of guidelines g_{i_2} (i.e. distinctions between similar types) for l_i, we first identify which types are confused with l_i. For a given type l_j $(j \neq i)$, if the number of samples in $\mathcal{D}^{sub}_{l_i l_j}$ exceeds a certain threshold, we determine that l_i and l_j are easily confused and prompt the model to summarize their distinctions:

$$g_{i_2}^{(j)} \sim P_{\mathcal{M}}(g_{i_2}^{(j)} | T_g^2, \mathcal{D}^{sub}_{l_i l_j}). \tag{4}$$

The second part of the guidelines for l_i is then constructed as: $g_{i_2} = [g_{i_2}^{(j_1)}; \ldots; g_{i_2}^{(j_m)}]$.

3.3 ICL Demonstration Selection

There is no annotated data for ICL demonstrations in zero-shot scenarios. In the previous process, we constructed a subset $\mathcal{D}^{sub}_{l}$ for each entity type l, consisting of samples with high reliability and diversity. They can be directly used as ICL demonstrations. In our experiments, we control the number of demonstrations for each type to be around two. We call them pseudo-ICL demonstrations.

Additionally, KNN sampling (i.e. retrieving the K-nearest neighbors based on similarity for each test instance) is a widely used strategy for ICL demonstration selection. However, it typically relies on an annotated sample library. In our scenario, we also test this strategy.

3.4 Iterative Annotation

Our framework gradually improves the predictions over several iterations. Previous sections have described the main processes within an iteration. This section supplements two additional processes.

Historical Predictions Integration As described in Section 3.2, we start with a pseudo-sample set $\mathcal{D}$ and proceed with the *sample library construction*. In the initial step, only the prediction $\mathcal{D}_0$ from this step is available. However, as the iterative process advances, we obtain multiple predictions $\mathcal{D}_t$ from different steps. To enhance the reliability of the pseudo-sample set $\mathcal{D}$, we consider integrating these predictions, similar to self-consistency (SC) (Wang et al., 2023c). This integration further enhances the overall reliability of the process.

Unlike SC, which generates different predictions by adjusting the model's temperature parameter, our historical predictions $\mathcal{D}_t$ are derived from distinct guidelines and pseudo-ICL demonstrations. We propose two integration strategies: (1) *Quantity-based Selection*: This follows the main approach of SC. An entity prediction (e, l_e) is selected if it appears in a sufficient number of predictions. We dynamically adjust this threshold based on the difference between the number of entities predicted in the current step and those in the previous steps. This ensures that the number of predicted entities does not fluctuate excessively, enhancing the stability of the iterative process. (2) *Weight-based Selection*: Given our efforts for iterative improvement, predictions $\mathcal{D}_i$ at steps i are theoretically more accurate than $\mathcal{D}_j$ $(j < i)$. Therefore, we assign weights to each entity prediction (e, l_e). When (e, l_e) appears in $\mathcal{D}_i$, it receives a weight w_i, with $w_i < w_j$ if $i < j$ (we simply set $w_i = i$). We accumulate the weights assigned to the prediction at each step. An entity prediction (e, l_e) is selected if its weight exceeds a certain threshold.

Termination Conditions A straightforward approach is empirically setting the maximum step to terminate the iteration. In our experiments, approximately five iterations are sufficient. We also adopt a simple termination strategy to enhance the framework's generalization and autonomy. Specifically, after several initial steps, if the difference between predictions of two consecutive steps, $|\mathcal{D}_i \setminus \mathcal{D}_{i+1}|/|\mathcal{D}_i|$, falls below a threshold (e.g. 5%), further iterations are not performed.

4 Experiments

4.1 Setup

Datasets and Metrics We conduct the experiments on four widely used NER benchmark datasets: CONLL03 (Sang and De Meulder, 2003), ACE05 (Walker et al., 2006), MIT-Movie and MIT-Restaurant (Ushio and Camacho-Collados, 2021). We use their test sets for experiments. We use the entity-level micro-F1 score as the metric, where both the entity boundary and entity type must be correctly predicted.

LLMs We evaluate the effectiveness of our framework using gpt-3.5-turbo, and the open-source models Qwen2-72B (Yang et al., 2024a) and Llama3.1-70B (Dubey et al., 2024). We utilize the instruction versions of these models for better instruction following ability. For most of our analytical experiments, we employed the locally deployed Qwen2-72B model. For more implementation details, please refer to our Appendix A.

4.2 Main Results

Firstly, we evaluate the overall effectiveness of our framework, as shown in Table 1. We compare our work with recent studies in similar scenarios, including **InstructUIE** (Wang et al., 2023b), **GoLLIE** (Sainz et al., 2023), **CodeIE** (Li et al., 2023), **Code4UIE** (Guo et al., 2023) and **Self-Improving** (Xie et al., 2024), to demonstrate the significance of our work. See appendix A for details. Additionally, we compare backbone LLMs without our framework (refer to as the **vanilla**) to those with our framework, demonstrating the effective improvements brought by our framework. We also attempt to extend our framework to sentiment information extraction tasks (appendix 4.9).

We further analyze our framework, as shown in Table 2. Specifically: (1) **Analysis of Guidelines**. Experimental results show that the worst performance mostly occurs without any guidelines (*w/o guidelines*). Either *guidelines_definition* or *guidelines_distinction* brings improvement, and using both yields the best results (i.e. the complete framework, EvoPrompt). A noteworthy comparison is between the *Vanilla* (zero-shot with crafted seed guidelines) and *w/o pseudo demonstrations* (zero-shot with model-summarized guidelines), which highlights the advantages of our approach. While manually crafted guidelines could potentially be superior, our framework strikes a better balance

between convenience and performance. (2) **Analysis of Pseudo Demonstrations**. The performance drop (*w/o pseudo demonstrations*) indicates that even noisy pseudo-ICL demonstrations can be beneficial. However, this depends on the quality of these demonstrations. Retrieving similar ones for each test query (*w/ nearest demonstrations*) may be less effective than not using any pseudo-samples, as directly retrieved neighbors from pseudo-samples often contain more noise.For fine-grained tasks like NER, the accuracy of ICL samples significantly impacts performance. (3) **Analysis of Pseudo-sample Selection**. This part affects both guidelines and pseudo-ICL demonstrations. We test the initial selection based on historical predictions integration (HPI) and the secondary selection based on clustering. Overall, the former has a slightly greater impact. However, the latter may have a greater effect on data with more noise (such as ACE05).

Additionally, we introduce annotated data as ICL demonstrations (**Scenarios with Annotated Data**). We use the training set to test two common ICL demonstration selection strategies: random and nearest neighbor-based selection. Results show that guidelines remain beneficial. However, with nearest neighbor selection, the impact of guidelines is minimal. A possible reason is that similar samples are more conducive to promoting model learning. Therefore, the quality of similar samples significantly influences the model's performance (comparing '*w/ nearest demonstrations*' and '*nearest + guidelines*'). Nevertheless, these experiments highlight the potential of our framework to generalize to scenarios with limited annotated data, closer to practical application scenarios

	Movie	Res	CONLL03	ACE05
InstructUIE (2023b)	63.00	20.99	–	–
GoLLIE-34B (2023)	62.40	52.70	–	–
CodeIE (2023)	–	–	72.66	45.99
Code4UIE (2023)	–	–	79.70	**57.00**
Self-Improving (2024)	–	–	74.99	32.29
gpt-3.5-turbo				
Vanilla	54.90	46.36	66.92	32.78
EvoPrompt	68.10	68.44	79.24	47.34
Qwen2-72B				
Vanilla	62.86	52.45	72.62	47.03
EvoPrompt	**70.93**	**69.26**	**81.33**	51.23
Llama3.1-70B				
Vanilla	49.17	48.36	69.16	36.22
EvoPrompt	65.45	58.41	73.57	42.92

Table 1: Experimental results. We compare with recent works in similar scenarios and vanilla performance (i.e. general zero-shot settings without our framework).

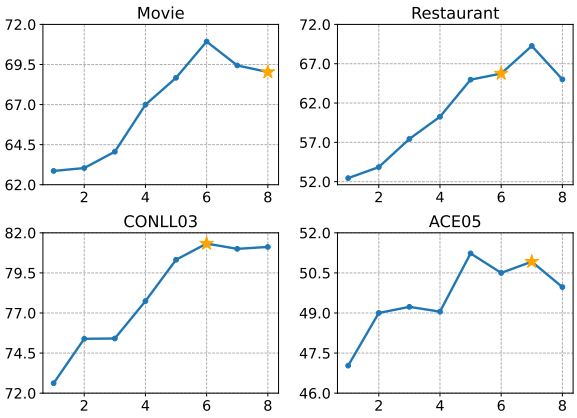


Figure 3: Iterative improvements on different datasets. The horizontal axis is the iteration number and the vertical denotes entity-level F1 scores. Star points indicate the last step based on the termination strategy.

4.3 Analysis of Iterative Improvements

This section analyzes performance changes during the iteration process. Figure 3 shows the performance for the four datasets across different iterations, using Qwen2-72B as the backbone LLM. Generally, the model's performance improves gradually in the iterations, confirming the effectiveness of our framework. After several iterations, performance tends to stabilize, and further iterations may cause a decline or fluctuation. This is expected, as both the generation of guidelines and the acquisition of ICL demonstrations rely on noisy pseudo-samples, which have optimization limits.

We further evaluate our framework on models of different sizes, as shown in Figure 4. The framework proves effective for both smaller-scale models and larger-scale models, demonstrating its robustness. For more details, refer to our appendix B.1.

4.4 Cognition Evolution of LLM

We show the extraction accuracy of several labels in Figure 5, where the darkened regions denote correct extractions, reflecting the consensus between the model and manual annotations. Our framework substantially enhances cognitive consistency between LLMs and manual annotations. For instance, in the CONLL dataset, the entity type "MISC" often suffers from ambiguous definitions in manual guidelines, leading to challenges in accurate extraction by the LLMs. Conversely, the entity type "PLOT" in the Movie dataset has a broad interpretation (*e.g.*, events, categories, countries, *etc.*), and

	Movie	Restaurant	CONLL03	ACE05	Avg
Vanilla	62.86	52.45	72.62	47.03	58.74
Analysis of Guidelines					
w/o guidelines	61.39	65.04	75.90	44.38	61.68
w/ guidelines_definition	65.06	68.70	78.67	48.31	65.19
w/ guidelines_distinction	64.47	67.98	78.94	49.51	65.23
Analysis of Pseudo Demonstrations					
w/o pseudo demonstrations	65.70	58.51	76.66	51.02	62.97
w/ nearest demonstrations	66.43	59.37	77.22	46.18	62.30
Analysis of Pseudo-sample Selection					
w/o HPI	66.19	64.63	76.10	50.56	64.37
w/o cluster	70.16	67.19	78.93	47.20	65.87
EvoPrompt	**70.93**	**69.26**	**81.33**	**51.23**	**68.19**
Scenarios with Annotated Data					
random	76.08	70.12	81.30	53.34	70.21
random + guidelines	77.63	71.36	82.84	54.85	71.67
nearest	84.38	75.08	89.58	61.07	77.53
nearest + guidelines	84.69	74.96	89.36	61.37	77.60

Table 2: Detailed analysis of our framework. We mainly analyze the impact of three factors: (1) guidelines, (2) pseudo-ICL demonstrations, and (3) representative pseudo-sample selection strategies. Additionally, we briefly explored the combination of guidelines and annotated data.

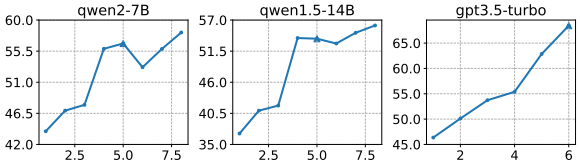
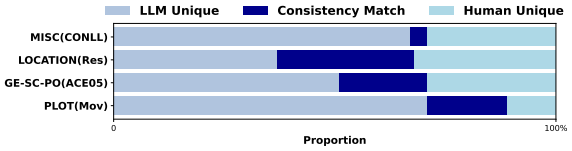
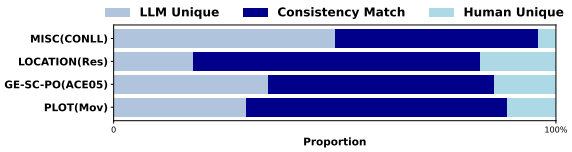


Figure 4: Iterative improvements in different scale and closed source models using the Restaurant dataset.



(a) Vanilla



(b) EvoPrompt

Figure 5: Effect of our framework on the evolution cognitive of the model. The darker areas represent the consensus between LLM and human understanding, while the lighter areas represent their unique cognitions.

models often struggle to capture its full conceptual breadth. By incorporating continuously refined guidelines, we observe a significant reduction in cognitive misalignment, demonstrating the effectiveness of our approach. For more details, refer to our appendix B.2.

4.5 Confusion Mitigation Visualization

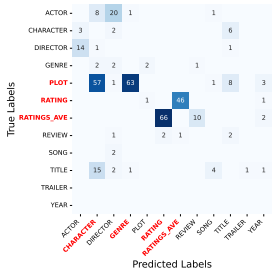
As shown in Figure 6, the confusion matrices on two datasets demonstrate that our framework effectively reduces the model's confusion between similar types compared with the original ICL setting, such as "GENRE" and "PLOT", "RATINGS_AVE" and "RATING" in the MIT-Movie dataset, and "CUISINE" and "DISH" in the MIT-Restaurant dataset. This improvement stems from our framework, which provides clear label definitions and distinctions, along with reliable and representative samples. This enhances the model's understanding of entity types and reduces noise interference. For more details, refer to our appendix B.3.

4.6 Case Study

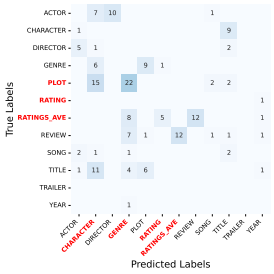
This section provides two cases to illustrate the effectiveness of our proposed framework, as shown in Table 3. In case (a), the traditional ICL-based solution fails to offer a detailed and comprehensive explanation of the labels "LOCATION" and "AMENTIY", leading the model to overly rely on

(a)	***Sentence:***	Are there any places to eat in the area that offer a two for one special?		
	Golden	(in the area, LOCATION)	(two for one special, AMENITY)	
	Traditional ICL	×	(***two for one***, ***PRICE***) ×	(***any***, ***AMENITY***) ×
	EvoPrompt (Ours)	√	√	
(b)	***Sentence:***	I would like to watch a romance about soccer.		
	Golden	(romance, GENRE)	(soccer, PLOT)	
	Traditional ICL	√	(soccer, ***GENRE***) ×	
	EvoPrompt (Ours)	√	√	

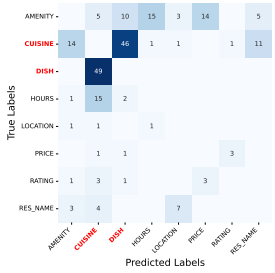
Table 3: Example cases with golden labels and predictions from traditional ICL and our method. √ indicates a correct prediction, while × represents a failure to generate. Incorrect components are highlighted in ***bold***.



(a) Vanilla-Movie



(b) EvoPrompt-Movie



(c) Vanilla-Restaurant

(d) EvoPrompt-Restaurant

Figure 6: The confusion matrix for our framework on the Movie and Restaurant datasets. Labels highlighted in red denote types prone to confusion, while the heatmap's color intensity indicates the degree of confusion, with darker shades representing higher levels of ambiguity.

its inherent understanding. This prevents it from consistently aligning with task requirements, ultimately failing to extract the relevant entities "*in the area*" and "*two for one special*". In case (b), the traditional ICL approach struggles to help the model distinguish between the two ambiguous labels "GENRE" and "PLOT". Our EvoPrompt framework successfully resolves these two problematic cases by incorporating dynamic, comprehensive, and model-adaptive guidance.

4.7 Efficiency Analysis

Our approach demonstrates promising results. However, the requirement for multiple iterations to refine guidelines and enhance accuracy inevitably leads to increased calls to the LLM. To assess the practicality of our method, we compared its inference performance and associated costs with those of other approaches, as summarized in Table 4. The results show that our EvoPrompt-based reasoning paradigm significantly reduces both time and financial costs compared to other iterative annotation-based methods (e.g., self-improve (Xie et al., 2024)). While our framework incurs higher costs than the Vanilla method, it substantially improves performance. We consider this trade-off acceptable, as the gains in accuracy outweigh the additional computational expense.

Model	**gpt-3.5-turbo**		**Qwen2-72B-Chat**	
	Cost/100 items	F1	Time/item	F1
Vanilla	0.35$	50.24	0.31sec	58.74
Self-Improving	2.34$	59.24	2.43sec	62.01
EvoPrompt(Ours)	1.47$	65.78	1.29sec	68.19

Table 4: Comparison of inference time or API call costs. All experimental metrics we report are averages of the four datasets used in the main experiment.

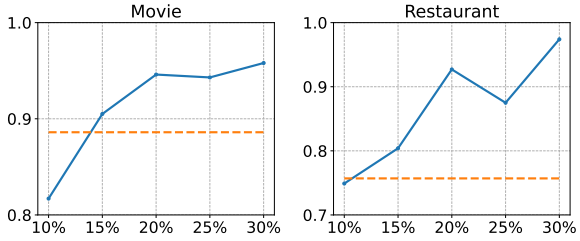


Figure 7: The performance with prompts derived from partial data. The horizontal axis represents the percentage of data used for prompt construction, and the vertical axis represents the performance ratio compared to full data settings. The dashed line indicates the vanilla performance (i.e. zero-shot without our framework).

4.8 Efficiency Improvements

The iterative process inherently increases time costs. Its primary goal is to generate multiple versions of predictions, thereby acquiring more reliable data for constructing better prompts. From

this perspective, reducing the data in the iterative process is feasible. To this end, we attempt to utilize a portion of the unlabeled data in the iterative process to construct prompts and then apply them for the final prediction on all test data. As shown in Figure 7, using approximately 20% of the data for prompt construction yields satisfactory performance. We hypothesize that increasing the data can enhance the comprehensiveness of guidelines and pseudo-samples but may also introduce more noise. Therefore, using an appropriate data volume during iteration can improve efficiency and potentially enhance performance as well.

4.9 Framework Generalization

In this section, we extend the proposed EvoPrompt framework to the Aspect-based Sentiment Analysis (ABSA) (Zhang et al., 2021; Hu et al., 2022; Tong et al., 2024) task, which focuses on extracting sentiment-related information from given sentences, including aspect terms, opinion terms, categories, and sentiment polarities. Similar to NER, we define the extraction target as a triplet comprising <*aspect*, *category*, *sentiment* >, where the aspect and category correspond to the entity and entity type, respectively, in NER. However, unlike NER, the extraction target also incorporates sentiment polarity associated with the aspect, categorized as positive, negative, or neutral. Moreover, due to the diversity of sentiment expression, a single aspect in a sentence may be associated with multiple emotional attributes and assigning multiple category labels, which presents more significant challenges and complexities.

We conduct experiments on the test sets of the Rest15, Rest16 (Zhang et al., 2021), Lap14 (Cai et al., 2021), and FSQP (Bai et al., 2024) datasets, which are widely recognized for their comprehensive annotations of sentiment quadruples, encompassing the aforementioned four sentiment elements. We only changed the output format and initial prompt of the task, and all other settings remain the same, as stated in appendix A. The micro F1 score (F1) is employed as the evaluation metric. Experimental results in Table 5 demonstrate that our model achieves robust performance, affirming the strong generalization capability of EvoPrompt.

4.10 Effectiveness of Guidelines

To further validate the effectiveness of the guidelines generated by EvoPrompt, we generalized them to GoLLIE (Sainz et al., 2023), which focuses on enhancing the model's ability to follow guidelines through fine-tuning. Specifically, we deployed GoLLIE with the guidelines generated by EvoPrompt, replacing the manually designed, fixed guidelines previously used. The experimental results in Table 6 show a significant performance improvement when GoLLIE applies EvoPrompt-generated guidelines, thereby confirming their effectiveness. This also suggests that EvoPrompt can be orthogonally integrated with parameter tuning-based methods, acting as a powerful complement.

	Rest15	**Rest16**	**FSQP**
gpt-3.5-turbo			
Vanilla	37.64	40.99	44.29
EvoPrompt	49.06	52.04	52.18
Qwen2-72B			
Vanilla	46.64	44.38	40.58
EvoPrompt	**55.70**	53.02	47.96
Llama3.1-70B			
Vanilla	46.86	55.73	50.49
EvoPrompt	55.65	**64.80**	**53.99**

Table 5: Experimental results. We compare EvoPrompt with vanilla performance (i.e. general zero-shot settings without our framework).

Method	**Movie**	**Restaurant**
GoLLIE-7B	63.00	43.40
GoLLIE-7B *w* EvoPrompt	**71.18**	**54.01**

Table 6: Effectiveness of the guidelines generated by Evoprompt for GoLLIE. The notation "w" represents the guidelines in GoLLIE that are initialized by the guidelines generated at the termination of the EvoPrompt iteration.

5 Conclusion

We propose the EvoPrompt framework for zero-shot NER, consisting of two carefully designed components: (1) the iteratively refined guidelines that inject dynamic, comprehensive, and model-adaptive entity type information into LLMs, mitigating their misunderstanding of types and reducing confusion between similar types, and (2) a sample selection strategy, grounded in self-consistency and prototype learning, which enhances the reliability and diversity of pseudo samples. Experiments on benchmark datasets demonstrate the effectiveness. We hope our contributions will offer meaningful insights for further advancements in the field.

Limitations

We acknowledge the following limitations of our work: (1) We primarily focus on the zero-shot NER task. Further experiments are needed for other IE tasks. (2) Some settings could be further optimized. For example, we can select more suitable sentence embedding models or further adapt them to the data domain. (3) The bias introduced by the model's prior knowledge is a challenge that deserves further exploration in future work.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No.62276110, No.62172039 and in part by the fund of Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL). The authors would also like to thank anonymous reviewers for their comments on improving the quality of this paper.

References

Yinhao Bai, Yalan Xie, Xiaoyi Liu, Yuhua Zhao, Zhixin Han, Mengting Hu, Hang Gao, and Renhong Cheng. 2024. BvSP: Broad-view soft prompting for few-shot aspect sentiment quad prediction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8465–8482.

Chenran Cai, Qianlong Wang, Bin Liang, Bing Qin, Min Yang, Kam-Fai Wong, and Ruifeng Xu. 2023. In-context learning for few-shot multimodal named entity recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2969–2979.

Hongjie Cai, Rui Xia, and Jianfei Yu. 2021. Aspect-category-opinion-sentiment quadruple extraction with implicit aspects and opinions. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 340–350.

Xiang Chen, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, Huajun Chen, and Ningyu Zhang. 2022. Lightner: A lightweight tuning paradigm for low-resource ner via pluggable prompting. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2374–2387.

Zhuojun Ding, Wei Wei, Xiaoye Qu, and Dangyang Chen. 2024. Improving pseudo labels with global-local denoising framework for cross-lingual named entity recognition. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 6252–6260.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Chenghao Fan, Wei Wei, Xiaoye Qu, Zhenyi Lu, Wenfeng Xie, Yu Cheng, and Dangyang Chen. 2024. Enhancing low-resource relation representations through multi-view decoupling. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, pages 17968–17976.

Yingjie Gu, Xiaoye Qu, Zhefeng Wang, Yi Zheng, Baoxing Huai, and Nicholas Jing Yuan. 2022. Delving deep into regularity: a simple but effective method for chinese named entity recognition. *arXiv preprint arXiv:2204.05544*.

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

Mengting Hu, Yike Wu, Hang Gao, Yinhao Bai, and Shiwan Zhao. 2022. Improving aspect sentiment quad prediction via template-order data augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7900.

Chen Jia, Xiaobo Liang, and Yue Zhang. 2019. Cross-domain ner using cross-domain language modeling. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2464–2474.

Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuanjing Huang, and Xipeng Qiu. 2023. Codeie: Large code generation models are better few-shot information extractors. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Xiang Li, Zhilei Hu, et al. 2024. Knowcoder: Coding structured knowledge into llms for universal information extraction. *arXiv preprint arXiv:2403.07969*.

Wanlong Liu, Shaohuan Cheng, Dingyi Zeng, and Qu Hong. 2023. Enhancing document-level event argument extraction with contextual clues and role relevance. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12908–12922.

Wanlong Liu, Li Zhou, Dingyi Zeng, Yichen Xiao, Shaohuan Cheng, Chen Zhang, Grandee Lee, Malu Zhang, and Wenyu Chen. 2024. Beyond single-event extraction: Towards efficient document-level multi-event argument extraction. *arXiv preprint arXiv:2405.01884*.

Keming Lu, Xiaoman Pan, Kaiqiang Song, Hongming Zhang, Dong Yu, and Jianshu Chen. 2023. Pivoine: Instruction tuning for open-world information extraction. *arXiv preprint arXiv:2305.14898*.

Yubo Ma, Yixin Cao, Yong Hong, and Aixin Sun. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10572–10601.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968.

Xiaoye Qu, Yingjie Gu, Qingrong Xia, Zechang Li, Zhefeng Wang, and Baoxing Huai. 2023a. A survey on arabic named entity recognition: Past, recent advances, and future trends. *IEEE Transactions on Knowledge and Data Engineering*.

Xiaoye Qu, Jun Zeng, Daizong Liu, Zhefeng Wang, Baoxing Huai, and Pan Zhou. 2023b. Distantly-supervised named entity recognition with adaptive teacher learning and fine-grained student ensemble. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13501–13509.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2023. Gollie: Annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations*.

Erik 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.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Zeliang Tong, Wei Wei, and . 2024. Cciiplab at sighan-2024 dimabsa task: Contrastive learning-enhanced span-based framework for chinese dimensional aspect-based sentiment analysis. In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pages 102–111.

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.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus ldc2006t06, 2006. *URL https://catalog. ldc. upenn. edu/LDC2006T06*.

Rui Wang, Tong Yu, Handong Zhao, Sungchul Kim, Subrata Mitra, Ruiyi Zhang, and Ricardo Henao. 2022. Few-shot class-incremental learning for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 571–582.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023a. Gpt-ner: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wei Wei, Zanbo Wang, Xianling Mao, Guangyou Zhou, Pan Zhou, and Sheng Jiang. 2021. Position-aware self-attention based neural sequence labeling. *Pattern Recognition*, 110:107636.

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

Xinglin Xiao, Yijie Wang, Nan Xu, Yuqi Wang, Hanxuan Yang, Minzheng Wang, Yin Luo, Lei Wang, Wenji Mao, and Daniel Zeng. 2023. Yayi-uie: A chat-enhanced instruction tuning framework for universal information extraction. *arXiv preprint arXiv:2312.15548*.

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.

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.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan

Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024b. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Mozhi Zhang, Hang Yan, Yaqian Zhou, and Xipeng Qiu. 2023. Promptner: A prompting method for few-shot named entity recognition via k nearest neighbor search. *arXiv preprint arXiv:2305.12217*.

Wenxuan Zhang, Yang Deng, Xin Li, Yifei Yuan, Lidong Bing, and Wai Lam. 2021. Aspect sentiment quad prediction as paraphrase generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9209–9219.

Zhen Zhang, Yuhua Zhao, Hang Gao, and Mengting Hu. 2024. Linkner: Linking local named entity recognition models to large language models using uncertainty. In *Proceedings of the ACM on Web Conference 2024*, pages 4047–4058.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2023a. Universalner: Targeted distillation from large language models for open named entity recognition. In *The Twelfth International Conference on Learning Representations*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023b. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

A Experimental Setup Details

A.1 Implementation Details

We employ gpt-3.5-turbo and two open-source models, Qwen2-72B, and Llama3.1-70B, as our foundational LLMs. We access the former via the gpt-3.5-turbo API; for the latter two, we utilize their GPTQ-Int4 quantized versions[1] [2], enabling local deployment. We leverage the vLLM[3] framework to boost the inference speed.

We set the temperature to 0 for LLMs for all experiments. We set the maximum length of each part of the guidelines to 32 tokens and the number of pseudo-ICL demonstrations for each type to 2. In the first two iterations, we use only guidelines without pseudo-ICL demonstrations. At this stage, we select the first sample from the dataset as a demonstration to show the input-output format (simply selecting the first one without any additional design). We employ LaBSE[4] as the sentence embedding model, which is a recently updated model in the HuggingFace repository at the time. We employ the classic K-means clustering method.

A.2 Recent Works

We compared several recent works in our experiments. Here is a brief overview: **InstructUIE** and **GoLLIE** enhance the model's general capability for information extraction (IE) tasks through instruction tuning, achieving strong zero-shot generalization to unseen data. **CodeIE** improves the model's ability to output structured content by using code-style prompts and code-based LLMs, making it more effective for IE tasks. **Code4UIE** utilizes annotated data to select ICL demonstrations based on sample similarities. **Self-Improving** stimulates the self-learning ability of LLMs to prompt them for zero-shot NER tasks.

As previously mentioned, the main goal of comparing our work with existing studies is to highlight the practical value of our results. While all methods focus on low-resource information extraction, they differ in specific configurations such as data and base model requirements. Therefore, we do not recommend using experimental results as the absolute criterion for method superiority.

[1] https://huggingface.co/Qwen/Qwen2-72B-Instruct-GPTQ-Int4

[2] https://huggingface.co/hugging-quants/Meta-Llama-3.1-70B-Instruct-GPTQ-INT4

[3] https://github.com/vllm-project/vllm

[4] https://huggingface.co/sentence-transformers/LaBSE

A.3 Details of Overall Workflow

Algorithm 1 EvoPrompt

Require: Unlabeled corpus $\mathcal{U}$, predifined label set $\mathcal{L}$, seed guidelines $\bar{g}$, task instruction for NER T_{ner}, task instruction for guidelines summarize T_g^1, T_g^2, language model P_θ, max iteration steps M and termination threshold δ.

$\mathcal{D}_0 \sim P_\theta(\mathcal{D}_0 \mid T_{ner}, \bar{g}, \mathcal{U})$
Initialize $t = 0$, $\hat{\mathcal{D}} = \mathcal{D}_0$
while $t \leq M$ and $|\mathcal{D}_t \setminus \mathcal{D}_{t+1}|/|\mathcal{D}_t| > \delta$ **do**
    Initialize $S^{icl} = \Phi$, $G = \Phi$
    for $l_i \in \mathcal{L}$ **do**
        Get sample libraries $\mathcal{D}_{l_i}^{sub}$, $\mathcal{D}_{l_i,l_j}^{sub}$ from $\hat{\mathcal{D}}$
        $g_{i_1} \sim P_\theta(g_{i_1} \mid T_g^1, \mathcal{D}_{l_i}^{sub})$
        $g_{i_2} \sim P_\theta(g_{i_2} \mid T_g^2, \mathcal{D}_{l_i,l_j}^{sub})$
        $g_i \leftarrow \{g_{i_1}, g_{i_2}\}$
        Sample pseudo ICL-demonstrations $S_{l_i}^{icl} \in \mathcal{D}_{l_i}^{sub}$
        $S^{icl} \leftarrow S^{icl} \cup S_{l_i}^{icl}$, $G \leftarrow g_i \cup G$
    end for
    $\mathcal{D}_t \sim P_\theta(\mathcal{D}_t \mid T_{ner}, G, S^{icl}, \mathcal{U})$
    $\hat{D} \leftarrow$ Integration$(\hat{D} \cup \mathcal{D}_t)$
    $t \leftarrow t + 1$
end while
return D_t

B Complete Experimental Analysis

B.1 Iterative Improvements

In section 4.3, we briefly analyze the iterative improvements. This section presents the iterative performance of the three primary vanilla LLMs on four datasets, as illustrated in Figure 8. Our framework consistently enhances the performance of the three vanilla LLMs, demonstrating strong generalizability. Notably, the improvement trends vary among models. Qwen2 and gpt-3.5-turbo typically show a stable increase or a slow initial improvement followed by a significant boost. Conversely, Llama often exhibits a substantial initial improvement followed by a gradual slowdown. The distinct characteristics of different models can also be observed in the subsequent experimental analysis.

B.2 Cognition Evolution

In section 4.4, we briefly analyze the cognition evolution of LLMs. As shown Figure 9, we observe that across various LLM sizes and architectures (including Llama3.1, Qwen2, and gpt-3.5-turbo), there is a consistent divergence in model-human cognitive alignment on specific labels, such as "PLOT" in the Movie dataset and "MISC" in the CONLL dataset. Furthermore, different LLMs exhibit distinct interpretations of the same labels, even when following identical manual guidelines. For instance, both Qwen2 and gpt-3.5-turbo align

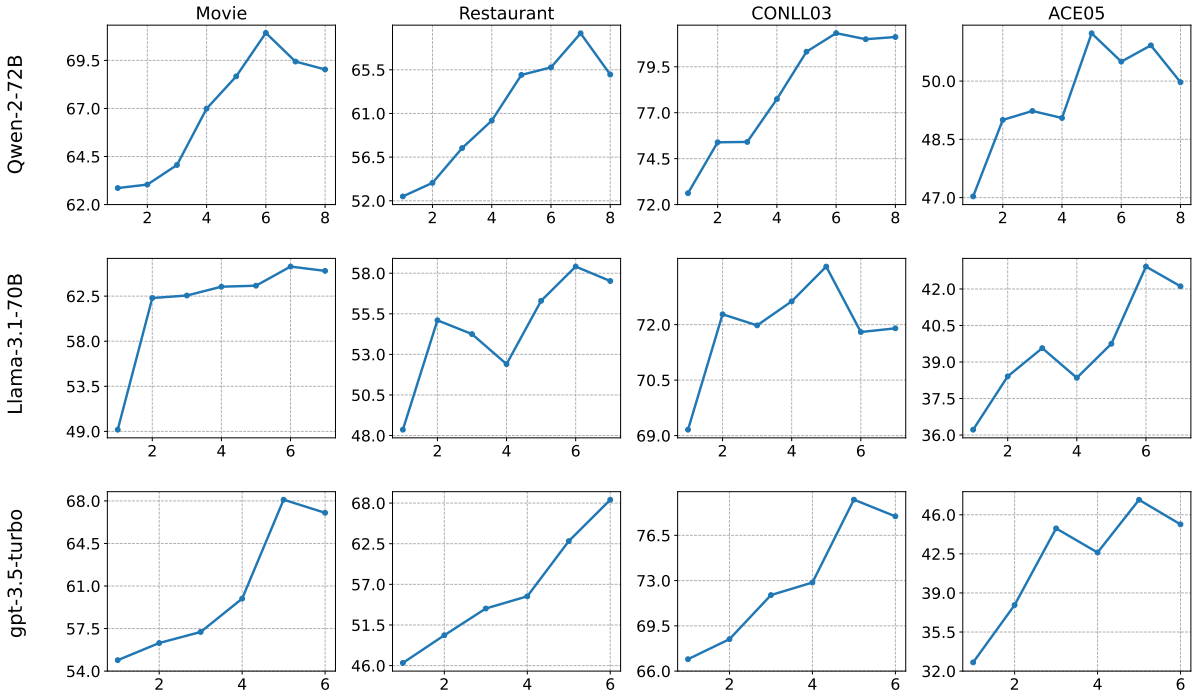


Figure 8: Iterative improvements on four datasets for Qwen-2-72B, Llama-3.1-70B and gpt-3.5-turbo. The horizontal axis is the iteration number and the vertical axis is the entity-level F1 scores.

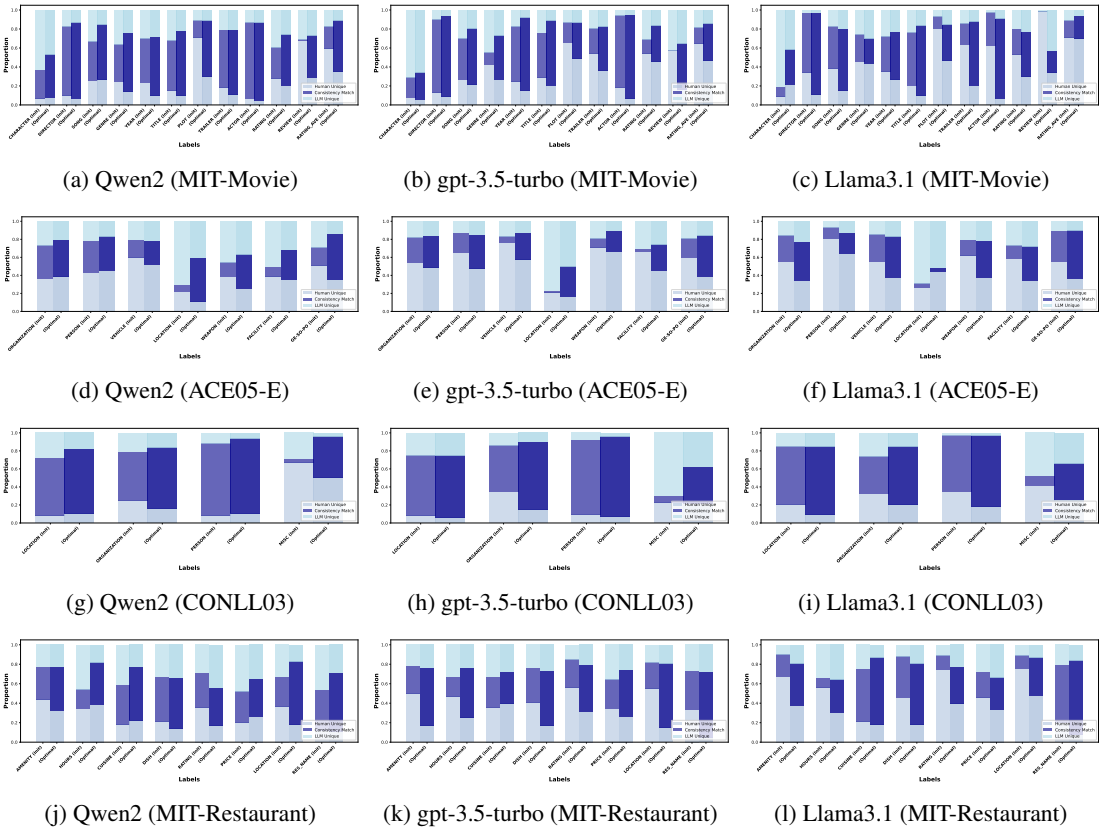
with the annotator's understanding of the label "ACTOR" in the Movie dataset, whereas Llama3.1 deviates in its interpretation. This discrepancy can be attributed to the inherent ambiguity or broad applicability of certain labels. As Sainz et al. (2023) noted, some labels are used to annotate a wide array of elements (e.g., "PLOT" refers to events, characters, countries, etc.), and handwritten guidelines often fall short in fully or adaptively correcting a model's innate understanding of these labels, thereby complicating the task of maintaining cognitive alignment between the model and annotators. Fortunately, our EvoPrompt framework introduces dynamic, comprehensive, and model-adaptive guidelines, which mitigate the model's misinterpretation of entity types. Notably, our framework yields significant improvements in aligning labels that historically exhibit substantial cognitive dissonance, such as "MISC" in the CONLL dataset, "RATING_AVE" in the MIT-Movie dataset, and "LOCATION" in the MIT-Restaurant dataset, among others.

B.3 Confusion Mitigation

In section 4.5, we briefly analyze the confusion mitigation. As shown in the upper part of Figure 10, we observe that various LLMs consistently confuse certain labels, especially those with high semantic similarity, such as "CUISINE" and "DISH" in the Restaurant dataset and "RATING" and "RATINGS_AVE" in the Movie dataset. These labels share similar characteristics, making it challenging to classify them accurately based solely on label names or straightforward interpretations. Additionally, different LLMs exhibit varied confusion patterns. For instance, Llama3.1 struggles to distinguish between "CHARACTER" and "ACTOR" in the Movie dataset, while gpt-3.5-turbo and Qwen2 correctly classify them. Conversely, the two models confuse "CHARACTER" with "PLOT", a distinction Llama3.1 manages successfully. This variation suggests that designing a universal prompt paradigm or guidelines is difficult, as different LLMs require tailored prompts. Our EvoPrompt framework addresses this issue by allowing models to self-induce optimal guidelines through reasoning, significantly reducing label confusion, as shown in the lower part of Figure 10.

C Compared with Automatic Prompt Engineering in Low-Resource Scenario

To further elucidate the value and advantages of our work in low-resource scenarios, this section provides a detailed analysis of the differences between EvoPrompt and other existing automatic prompt engineering-based optimization methods, accompanied by comprehensive experimental comparisons.



(a) Qwen2 (MIT-Movie) (b) gpt-3.5-turbo (MIT-Movie) (c) Llama3.1 (MIT-Movie)

(d) Qwen2 (ACE05-E) (e) gpt-3.5-turbo (ACE05-E) (f) Llama3.1 (ACE05-E)

(g) Qwen2 (CONLL03) (h) gpt-3.5-turbo (CONLL03) (i) Llama3.1 (CONLL03)

(j) Qwen2 (MIT-Restaurant) (k) gpt-3.5-turbo (MIT-Restaurant) (l) Llama3.1 (MIT-Restaurant)

Figure 9: The impact of our framework on the cognitive evolution of Qwen, Llama, and gpt-3.5-turbo. The x-axis denotes labels, and each label's left and right columns represent the initial and optimized results. The darker areas represent the consensus between LLM and human understanding, while the lighter represent their unique cognitions.

C.1 Automatic Prompt Engineering

Automatic prompt optimization aims to refine the prompts to achieve optimal performance. The current approach involves generating an initial set of prompts manually or automatically and performing preliminary validation on the training set, followed by iterative refinement through the integration of specifically designed optimization rules and tailored feedback utilization methods, thereby progressively enhancing prompt quality. For example, Zhou et al. (2023b) introduced **APE**, which first prompts the LLM to generate candidate prompts, then utilizes their performance scores on the training set as feedback signals and simulates a Monte Carlo search process to obtain high-quality prompts. Pryzant et al. (2023) proposed **ProTeGi**, which primarily utilizes error cases as feedback signals and refines the prompts through the LLM's analysis of the underlying causes of these errors during the iterative optimization process. Yang et al. (2024b) introduced **OPRO**, which leverages the entirety of past iteration records as feedback signals, including the performance of each prompt in every round and regularity in the iterative process, thereby refining the prompts through the LLM's logical reasoning capabilities.

	Movie	Restaurant	CONLL03	ACE05
EvoPrompt	**68.10**	**68.44**	**79.24**	**47.37**
$\text{APE}_{\text{forward}}$	62.39	63.93	69.34	41.02
$\text{APE}_{\text{reverse}}$	61.77	65.00	67.58	41.36
ProTeGi	64.27	61.98	73.43	42.92
OPRO	63.18	65.09	72.80	44.16

Table 7: Experimental comparison of EvoPrompt with other automatic prompt optimization methods. For APE, we deploy it in two modes: (1) the forward mode provides task examples first, allowing the LLM to generate the prompt at the final stage, and (2) the reverse mode provides the prompt's position initially, enabling the LLM to generate it in a fill-in-the-blank manner.

While these methods have enabled LLMs to self-optimize prompts, they still require a certain amount of annotated data to provide real-time feedback on prompt performance. In resource-scarce scenarios, the scale of downstream annotated data is often insufficient to support this process, lead-

(a)	***Sentence:***	I'm trying to find a family friendly restaurant with a gift shop within 10 miles of sunswept hotel in orange beach.
	Golden:	(within 10 miles of sunswept hotel in orange beach, LOCATION)
	EvoPrompt:	(within 10 miles of sunswept hotel, LOCATION)
(b)	***Sentence:***	Show me the movie where cary grant has two aunts that poison and bury old men.
	Golden:	(two aunts that poison and bury old men, PLOT)
	EvoPrompt:	(cary grant has two aunts that poison and bury old men, PLOT)
(c)	***Sentence:***	Find the melissa leo movie about smuggling illegals across the border.
	Golden:	(about smuggling illegals across the border, PLOT)
	EvoPrompt:	(smuggling illegals across the border, PLOT)

Table 8: Example cases with predictions from EvoPrompt for complex long named entities

ing to suboptimal results. In contrast, EvoPrompt introduces a novel automated framework that optimizes prompts without annotated data, demonstrating strong adaptability in low-resource settings.

C.2 Experiments and Analysis

To empirically compare the performance of various automated prompt optimization methods, we conducted experimental comparisons between EvoPrompt and the works mentioned above. The implementations of these methods adhered to their officially released code and configurations. We provide five real annotated examples for each label as feedback signals for the iterative optimization of the methods mentioned above and report the average results across five runs with different random seeds. We use gpt-3.5-turbo as the backbone, and the results are presented in Table 7.

We infer that the suboptimal performance of automated prompt optimization methods can be attributed to two primary factors: (**1**) under low-resource conditions, models struggle to derive detailed and comprehensive guideline interpretations from limited annotated data; (**2**) These methods often face challenges in pinpointing and optimizing the weaker components of the prompt. This lack of specificity in the optimization target impedes the development of consistent guidelines for tagging elements with ambiguous or easily confused labels.

In short, thanks to the refinement and targeted optimization of various components in prompt, as well as the screening strategy of high-quality pseudo-samples, EvoPrompt performs well under resource-poor conditions with a lack of labeled data and has great usage value and broad application scenarios.

D Complex Long Named Entities

To obtain a comprehensive understanding of our EvoPrompt framework, we also analyze its failure cases. While EvoPrompt achieves state-of-the-art performance across various situations, it occasionally fails when handling complex long-named entities, as illustrated in Table 8. However, this failure is not due to a fundamental technical issue with the model's understanding of labels but rather arises from differences in extraction granularity, such as the retention level of entity modifiers, prepositions, and other contextual information. We believe this issue can be addressed through more detailed entity boundary definitions and contextual information. Given the rarity of such cases, we leave the perfection of EvoPrompt as the future work[5].

E Human Effort to Build Seed Guidelines

In the initial zero-shot process, we designed seed guidelines for each entity type to enhance the performance of the vanilla model. We directly used the official guidelines for datasets that already provided entity types as seed guidelines. For datasets without publicly available guidelines, we organized a group of domain experts to design guidelines for each entity type and conducted a voting process to select the best ones as seed guidelines.

It is important to note that since EvoPrompt has the ability to summarize, generalize, and iteratively optimize guidelines, the role of seed guidelines is primarily to assist the model in the initial understanding of label interpretation. This requirement is also common across most IE tasks.

F Prompts

F.1 Prompts for NER

The prompts used for NER are as follows, including three scenarios: (1) zero-shot, (2) guidelines-only, and (3) the combination of guidelines and pseudo-ICL demonstrations.

[5]We want to point out that existing LLM-based zero-shot schemes cannot handle those situations perfectly as well.

Prompt for Zero-shot NER

Given entity types:
{"type1","type2",. . . }.
[seed guidelines (optional)]
Please recognize the named entities in the text belonging to the given types. Don't fabricate entities that don't belong to the text.
Provide answers in the following JSON format: {"entity": "type"}.
Text: {text}
Answer:

Prompt with Guidelines

Given entity types:
{"type1","type2",. . . }.
The definitions of each type are:
{"type1":"guideliens1", . . . }
Please recognize the named entities in the text belonging to the given types. Don't fabricate entities that don't belong to the text.
Provide answers in the following JSON format: {"entity": "type"}.
Text: {text}
Answer:

Prompt with Guidelines & pseudo-ICL demonstrations

Given entity types:
{"type1","type2",. . . }.
The definitions of each type are:
{"type1":"guideliens1", . . . }
Please recognize the named entities in the text belonging to the given types. Don't fabricate entities that don't belong to the text.
Provide answers in the following JSON format: {"entity": "type"}.
Here are some examples:
Text: {text1}
Answer: {answer1}
. . .
Text: {text}
Answer:

F.2 Prompts for Guidelines Generation

The prompts used for NER are as follows, including two parts: (1) definition summarization and (2) distinction summarization.

Prompt for Definition Summarization

Currently, the "type1" is defined as: "{previous guidelines}".

The following are some texts which contain entities of the type. According to these examples, supplement or modify the definition to make it more complete.

In text "text", the "type1" entities are: {entity1,entity2,. . . }

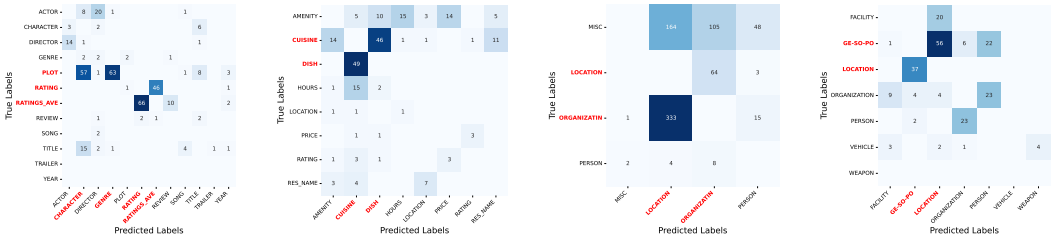
According to these examples, the "type1 refers to:

Prompt for Distinction Summarization

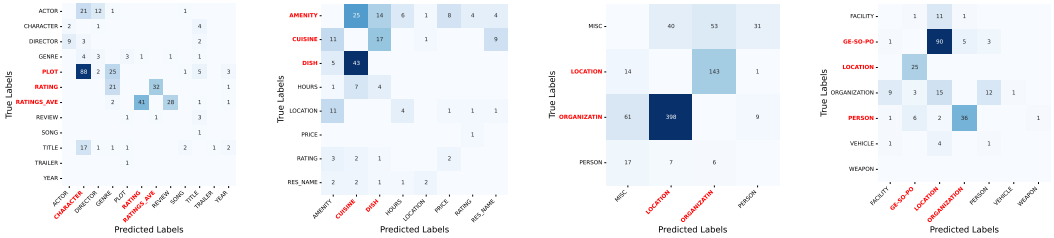
The following are some examples of different types of the same entity:

"entity1" is a "type_a" entity in text: "text_a"; and a "type_b" entity in text: "text_b"
. . .

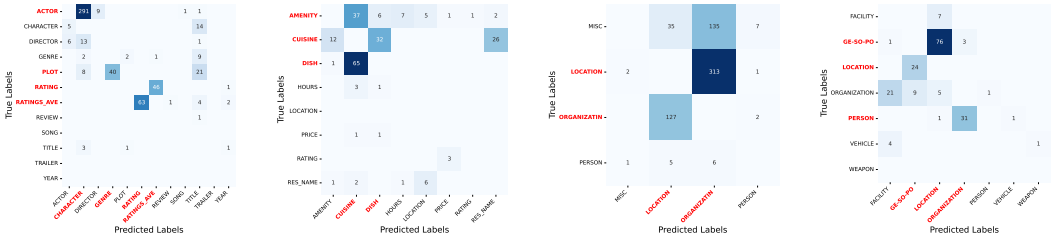
According to these confusing examples, compared to the "type_b", the "type_a" refers to:



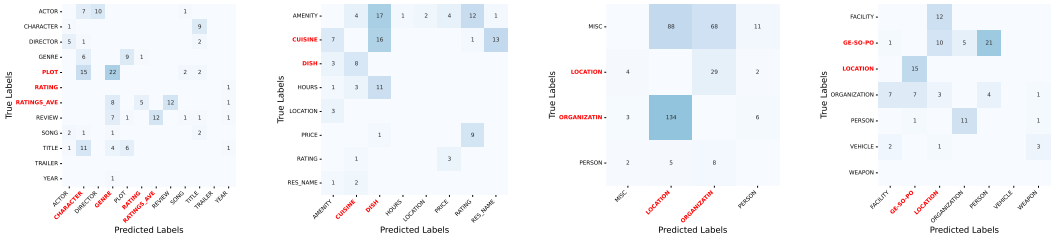
(a) Qwen2-Movie (b) Qwen2-Restaurant (c) Qwen2-CONLL (d) Qwen2-ACE05



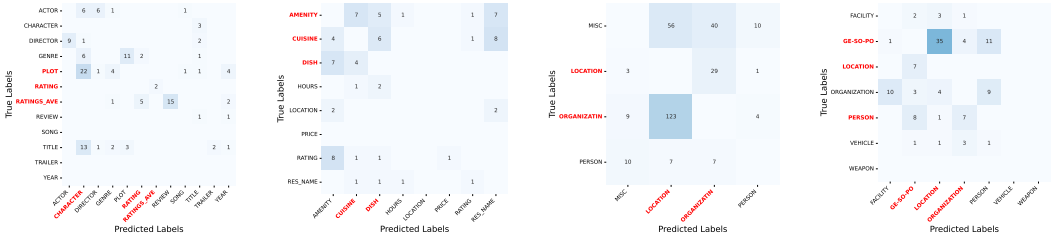
(e) gpt-3.5-Movie (f) gpt-3.5-Restaurant (g) gpt-3.5-CONLL (h) gpt-3.5-ACE05



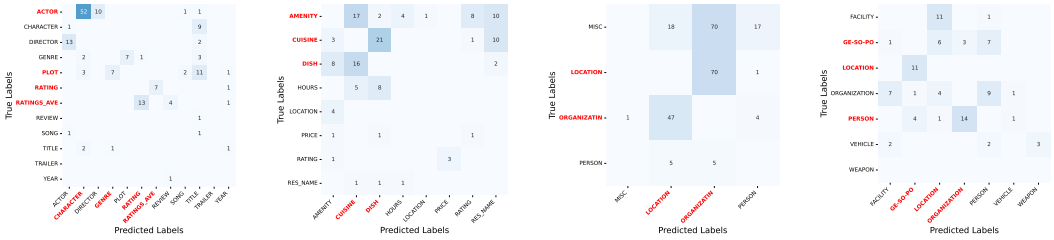
(i) Llama3.1-Movie (j) Llama3.1-Restaurant (k) Llama3.1-CONLL (l) Llama3.1-ACE05



(m) Qwen2-Movie (n) Qwen2-Restaurant (o) Qwen2-CONLL (p) Qwen2-ACE05



(q) gpt-3.5-Movie (r) gpt-3.5-Restaurant (s) gpt-3.5-CONLL (t) gpt-3.5-ACE05



(u) Llama3.1-Movie (v) Llama3.1-Restaurant (w) Llama3.1-CONLL (x) Llama3.1-ACE05

Figure 10: Confusion matrix of our framework on all datasets across various LLMs (Qwen2, gpt-3.5-turbo and Llama3.1). The part above the dotted line represents the test results of the vanilla, and the part below the dotted line represents the results of applying EvoPrompt .