Handling Missing Entities in Zero-Shot Named Entity Recognition: Integrated Recall and Retrieval Augmentation

Ruichu Cai[1,2], **Junhao Lu**[1], **Zhongjie Chen**[1], **Boyan Xu**[1*], **Zhifeng Hao**[1,3]
[1]School of Computer Science, Guangdong University of Technology
[2]Peng Cheng Laboratory
[3]College of Science, Shantou University
{cairuichu, lujunhao.code, bariancgg, hpakyim}@gmail.com
haozhifeng@stu.edu.cn

Abstract

Zero-shot Named Entity Recognition (ZS-NER) aims to recognize entities in unseen domains without specific annotated data. A key challenge is handling missing entities while ensuring accurate type recognition, hindered by: 1) the pre-training assumption that each entity has a single type, overlooking diversity, and 2) insufficient contextual knowledge for type reasoning. To address this, we propose IRRA (Integrated Recall and Retrieval Augmentation), a novel two-stage framework leveraging large language model techniques. In the *Recall Augmented Entity Extracting* stage, we built a perturbed dataset to induce the model to exhibit missing or erroneous extracted entities. Based on this, we trained an enhanced model to correct these errors. This approach can improve the ZS-NER's recall rate. In the *Retrieval Augmented Type Correcting* stage, we employ Retrieval-Augmented Generation techniques to locate entity-related unannotated contexts, with the additional contextual information significantly improving the accuracy of type correcting. Extensive evaluations demonstrate the state-of-the-art performance of our IRRA, with significant improvements in zero-shot cross-domain settings validated through both auto-evaluated metrics and analysis. Our implementation will be open-sourced at https://github.com/DMIRLAB-Group/IRRA.

1 Introduction

Zero-shot Named Entity Recognition (ZS-NER) aims to achieve efficient named entity recognition in unseen domains without requiring training on specific annotated datasets. ZS-NER allows for effective entity detection and classification in various domains(Xie et al., 2024a; Picco et al., 2023). The existing leading ZS-NER approaches involve end-to-end training of a robust language model on large NER corpora to enhance its cross-domain generalization capabilities.

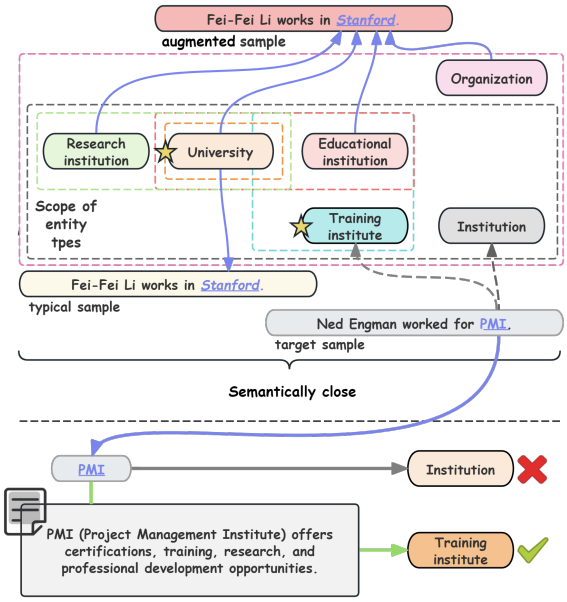

Figure 1: The upper part shows the difference between typical dataset and perturbed dataset. The typical dataset usually assigns one single gold type to each entity, while The perturbed dataset assigns multiple types to each entity. The dashed boxes represent the scope of entity types sets corresponding to these entity types. The lower part illustrates that the entity types of the retrieved entities may be not precise enough.

Although promising results were reported, we observe that end-to-end solutions have two notable limitations that adversely impact their generalization performance: (i) the lack of diversity in the data annotation process. Existing NER corpora typically assign only one type to each entity, whereas entities in real world may correspond to multiple types. This restricts the model's ability to learn the diversity of entities and understand the connections between different types during end-to-end training. As shown in Figure 1, given a sample similar in form to the training sample but with entities of different types, the semantic distance between the

*Corresponding author, hpakyim@gmail.com

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 10790–10802
April 29 - May 4, 2025

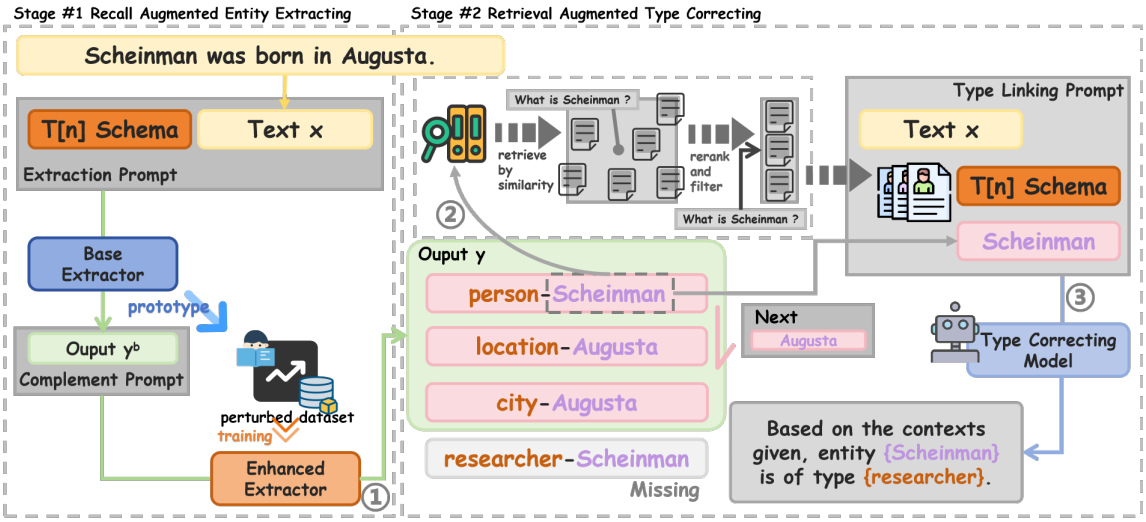

Figure 2: **The overall framework**. In Stage #2, The extracted entities are used to retrieve contexts, and these retrieved entity-related contexts are used to prompt the LLM to correct the type of the entity. ("Missing" indicates that the entity is not extracted in Stage #1)

types "Train institute", "Institution" and "University" could lead to "PMI" being neither extracted nor recognized, or (ii) the lack of sufficient contextual information for entity recognition. As shown in Figure 1, Unlike in the given training set sample, the context surrounding the entity "PMI" may be insufficient to determine its specific type. The key to overcome these limitations lies in decoupling the end-to-end processes to introduce specific augmentation mechanisms.

Instead of utilizing end-to-end methods, we propose the **IRRA** (**I**ntegrated **R**ecall and **R**etrieval **A**ugmentation), a two-stage ZS-NER framework that strategically leverages large language models (LLMs) techniques. Specifically, **IRRA** is featured with a *Recall Augmented Entity Extracting* stage and *Retrieval Augmented Type Correcting* stage. In *Recall Augmented Entity Extracting* stage, we introduce a data perturbation method by enriching each entity's type with synonyms and hypernyms, thereby constructing a perturbed dataset which model will make inference errors on it. We then use these errors to build an enhanced dataset to train an enhanced extractor, which can recall more entities. Through the above process, as shown in Figure 1, the entity "Stanford" in the training sample matched to multiple types, also increases the probability that the test sample entity "PMI" to be extracted, due to the introduction of the types "Educational institution" and "Institution", which are more closely related to "Training institute". In *Retrieval Augmented Type Correcting* stage, we employ Retrieval-Augmented Generation (RAG) techniques to retrieve contexts in the database related to the entities extracted in the previous stage. Then we instruct a frozen LLM to correct the entity's type based on the contextual information brought by the retrieved contexts. The main contributions of the paper are summarized as follows:

- A two-stage ZS-NER framework, **IRRA**, composed of an entity extracting stage and a type correcting stage, integrates LLMs techniques to address the issue of lack of information in end-to-end methods.
- A *Recall Augmented Entity Extracting* stage constructs a perturbed dataset that improves type diversity. Based on this, we propose two different extraction approaches, achieving nearly a 20% improvement in recall rate compared to the base extractor.
- A *Retrieval Augmented Type Correcting* stage incorporates RAG techniques to address the issue of performance degradation in accurately classifying entities without necessary extensive contextual information.
- We conduct extensive experiments to evaluate the effectiveness of **IRRA**, which notably outperforms the state-of-the-art LLM-based methods.

2 Related Work

2.1 LLM Self-correction

The evolution of LLMs has brought the concept of self-correction into focus. The discussion around self-correction centers on whether these advanced models can recognize the accuracy of their outputs and provide refined answers (Bai et al., 2022; Madaan et al., 2023; Welleck et al., 2022; Huang et al., 2024). Previous research has explored self-correction methods based on feedback (Pan et al., 2023), where feedback can come from the LLM itself or from external inputs. Internal feedback relies on the model's inherent knowledge and parameters. In contrast, external feedback involves inputs from humans, other models (Wang et al., 2023a; Paul et al., 2024), external tools, external knowledge(Gou et al., 2024; Chen et al., 2024; Olausson et al., 2024; Gao et al., 2023), etc. Our approach is inspired by this technology.

2.2 Retrieval Augmented NER

The RAG method enhances the performance of LLMs by incorporating external knowledge (Lewis et al., 2021). The standard RAG approach follows the "retrieve-then-read" paradigm. First, a retrieval query is constructed, which is then used to search for relevant information from an external knowledge source such as BookCorpus (Kobayashi, 2018). The retrieved information is then provided as context to the LLM, allowing the model to generate the final response based on it. Retrieval is based on similarity scores between the query and knowledge text and is mainly divided into two categories: sparse retrievers, like BM25 (Robertson, 2009); and dense retrievers, such as DPR (Karpukhin et al., 2020), Contriever (Izacard et al., 2022), and BCEmbedding. Additionally, there is work involving graph-based retrieval (Baek et al., 2023).

In the NER field, there is research using RAG technique to retrieve examples from annotated corpora as few-shots to improve the model's extraction performance (Xie et al., 2024b). While in medical and biological NER tasks, there is research retrieving relevant entity descriptions by entities matching from the specific databases to correct entity type (Kim et al., 2024).

However, prior work introducing the RAG technique into the NER field has not attempted to correct entity type in open-domain NER tasks. The work on retrieving few-shot examples does not focus on introducing entity-related external contexts for the model to reference, while retrieval methods in specific domains require manually constructed and maintained databases. To the best of our knowledge, we are the first to introduce the RAG method in the open-domain NER field, and using raw contexts as the retrieval data source. **IRRA** retrieves entity-related contexts to guide the LLM in correcting entity type. This retrieval process can be directly performed by a search engine, and the knowledge base used for retrieval does not require manual annotation, which enhances the robustness and generalizability of the method.

3 Methodology

The overall **IRRA** framework is shown in Figure 2. The entire process of this framework is divided into two stages: *Recall Augmented Entity Extracting* stage and *Retrieval Augmented Type Correcting*. In *Recall Augmented Entity Extracting* stage, we first deploy an LLM for entity extraction, then construct a perturbed dataset and train an LLM on it to enhance the initial extraction results. This stage is responsible for providing entity extraction results for the next stage while also enabling multi-label recognition. In *Retrieval Augmented Type Correcting* stage, based on the extracted entities, corresponding queries are generated to retrieve and rerank contexts from the database. These contexts are then provided as additional contextual information to the type correcting model.

3.1 Recall Augmented Entity Extracting

We develop two specialized models to extract entities. First, a base extractor $Extractor_{base}$ is used for initial entity extraction. Then, a perturbed dataset is constructed, and an enhanced extractor $Extractor_{enhanced}$ is trained to identify potentially missed entities, thus improve the recall rate of this stage. Afterward, the extraction results of both extractors are merged, providing the results for stage 1, which consists of multiple entity-type pairs.

3.1.1 Base Extractor

We first train a vanilla model on the original dataset D to serve as the $Extractor_{base}$ to directly extract the initial entities y_i^{base}:

$$y_i^{base} = Extractor_{base}(x_i, T_i).$$

Where x_i denotes the sentence to be extracted; T_i denotes the schema which is a set of entity types

that extraction needs to be performed based on. To facilitate the analysis of the generated results, we stipulate that y_i^{base} must be a parsable JSON string, where each key represents an entity type and its corresponding value is a list of entities.

3.1.2 Enhanced Extractor

To obtain the $Extractor_{enhanced}$ capable of supplementing the extraction results of the $Extractor_{base}$, we aim to discover behaviors where the $Extractor_{base}$ overlooks some potential entities. To achieve this, we introduce a series of perturbation measures to disrupt the training dataset of the $Extractor_{base}$. When the $Extractor_{base}$ performs inference on this perturbed dataset, it exhibits behaviors of extraction omission or error. The perturbation measures are as follows. Formally, given a dataset

$$D = \{(x_i, T_i, y_i) | i \in [0, |D|)\}.$$

$y_i = \{(e_j, t_j) | j \in [0, |y_i|)\}$ denotes the entity-type pairs for the i-th sample in D. We first define a function $\mathbf{SH}(\cdot)$ based on the LLM API, which map an entity type t to the union of the synonym set Syn_t and hypernym Hyn_t set for that entity type:

$$\mathbf{SH}(t) = Syn_t \cup Hyp_t.$$

Given the i-th sample in the original dataset D, we apply the following three perturbation measures to get the perturbed dataset D':

- **Random Syno-Hypernym Selection:** For each $t \in T_i$, we randomly select a $t' \in \mathbf{SH}(t)$ and replace t with t' in T_i^p. This perturbation introduces variability and disrupts the Base Extractor's reliance on a fixed vocabulary for understanding entity types, helping to uncover instances where it overlooks latent entities during inference.

- **Type Order Shuffle:** We randomly shuffle T_i^p. This perturbation makes it more difficult for the model to correctly extract entities, as it disrupts its reliance on memorizing task definitions based on entity type order, reflecting the reality that entity type order is not fixed in real-world scenarios and encouraging the model to understand the underlying relationships instead.

- **Syno-Hypernym Types Insertion:** This is the central aspect of our perturbation strategy. For each type $t \in T_i^p$, we first draw a random value $r \sim U(0, 1)$ from a uniform distribution. If $r < p$, we randomly select a $t' \in \mathbf{SH}(t)$ and insert it into T_i^p. This process is repeated, updating the value of r each time, until $r \geq p$. This method compels the model to learn that an entity can correspond to different types and helps the model better comprehend the relationships between various entity types, thereby enhancing its ability to generalize in diverse contexts.

Algorithm 1: Dataset perturbation with Noise Introduction

Input: D, p
$D' = \{\}$
for $(x_i, T_i, y_i) \in D$ **do**
  $T_i^p = \{\}$
  for $t \in T_i$ **do**
    $t' \leftarrow$ randomly sample in $\mathbf{SH}(t)$
    $T_i^p \leftarrow T_i^p \cup \{t'\}$
  end for
  $T_i^p \leftarrow$ randomly shuffle T_i^p
  $r \sim U(0, 1)$
  for $t \in T_i^p$ **do**
    while $r < p$ **do**
      $t' \leftarrow$ randomly sample in $\mathbf{SH}(t)$
      $T_i^p \leftarrow T_i^p \cup \{t'\}$
      $r \sim U(0, 1)$
    end while
  end for
  $y_i^p = \{\}$
  for $(e_j, t_j) \in y_i$ **do**
    for $t \in T_i^p \cap \mathbf{SH}(t_j)$ **do**
      $y_i^p \leftarrow y_i^p \cup \{(e_j, t)\}$
    end for
  end for
  $D' = D' \cup \{(x_i, T_i^p, y_i^p)\}$
end for
Output: D'

Subsequently, we modify the output of each sample to align the schema T_i^p. Consider the j-th entity-type pair (e_j, t_j) in y_i of the i-th sample, we take out all types in $\mathbf{SH}(t_j)$ that have been placed into T_i^p, and assign each of them to that entity to obtain the output $y_i^p = \{(e_j, t_{jk}) | j \in [0, |y_i|) \wedge k \in [0, |\mathbf{SH}(t_j) \cap T_i^*|)\}$. Finally, we obtain the perturbed dataset

$$D' = \{(x_i, T_i^p, y_i^p) | i \in [0, |D|)\}.$$

Then we use the $Extractor_{base}$ to inference on

D', obtain

$$y_i' = Extractor_{base}(x_i, T_i^p),$$

Then, we place y_i' into D', resulting in the enhanced dataset D'' for training $Extractor_{enhanced}$:

$$D'' = \{(x_i, T_i^p, y_i', y_i^p) | i \in [0, |D|)\}.$$

Then we train the $Extractor_{base}$ on D'' to obtain the $Extractor_{enhanced}$, Then we use the $Extractor_{enhanced}$ to expand the initial recall results. Finally, we have the result of stage 1:

$$y_i^{final} = Extractor_{enhanced}(x_i, T_i, y_i^{base}).$$

Similarly, y_i^{final} is a parsable JSON string in the same format. We use y_i^{final} for correction in the subsequent steps.

Utilizing these methods on the IEPile dataset, we observed a nearly 20% increase in recall. This significant improvement highlights the effectiveness of our perturbation strategy. By using the perturbed dataset, we aim to develop a more adaptable and resilient model capable of handling diverse and complex real-world scenarios.

3.2 Retrieval Augmented Type Correcting

The previous stage can recall more entities, but there is no guarantee that the corresponding type of each entity is inferred correctly. thus after extracting entities, we need to correctly correct their types. We use LLMs to perform the type correcting task. However, since x_i may lack relevant contextual information to help the LLM identify the correct entity type, We believe that external contexts is needed. We pull out all entities from y_i^{final} and deduplicate them to obtain the set of all entities to be processed, denoted as $E_i^* = \{e_j^* | j \in [0, |E_i^*|)\}$. For each $e_j^* \in E_i^*$, we apply retrieval augmented type correcting approaches to correctly match the entity with its type. In the correcting stage, we retrieve a small set of entity-related external contexts to guide the correcting model.

3.2.1 Vector Database Construction

The corpora used for retrieval can come from sources such as Wikipedia, domain-specific databases, websites, and other data that do not require manual annotation. In **IRRA**, the retrieval method is independent of the module, we can use search engine retrieval, vector database retrieval, or other methods. However, since using search engine APIs can be costly, we pre-collect the relevant data and use vector database retrieval in this paper. After collecting a large amount of original text, due to the length of the original text often exceeds the context window length of the LLM, we chunk the original text into multiple shorter contexts. Let the set of all contexts be $\mathcal{D}_{all} = \{d_l | l \in [0, |\mathcal{D}_{all}|)\}$. Given the embedding model, we define the embedding function $\mathbf{Emb} : \mathcal{T} \mapsto \mathbb{R}^D$, where $\mathcal{T}$ represents the text space and $\mathbb{R}^D$ represents the D-dimensional embedding space. For each context $d_l \in \mathcal{D}_{all}$, we obtain its vector representation $\mathbf{d_l} = \mathbf{Emb}(d_l)$. Then we store all context vectors in a open-source vector database along with its domain in the corresponding metadata for subsequent retrieval.

3.2.2 Contexts Retrieval

Given an entity to be processsed $e_j^* \in E_i^*$, we construct a retrieval query $q_j =$ `"What is` e_j`"`, and then obtain the vector representation of the query $\mathbf{q_j} = \mathbf{Emb}(q_j)$. Subsequently, we compute the cosine similarity between the query and the context:

$$CosineScore(q_j, d_l) = \frac{\mathbf{q_j} \cdot \mathbf{d_l}}{||\mathbf{q_j}|| \cdot ||\mathbf{d_l}||}.$$

Based on the $CosineScore$ of all contexts, we recall the top ncontexts with the highest score, denoted as $\hat{\mathcal{D}} = \{\hat{d}_l | l \in [0, n)\}$.

Due to the limitations of cosine similarity, we additionally use a reranker model to further filter the $\hat{d}_l \in \hat{\mathcal{D}}$. The reranker model is based on the cross-encoder architecture. Unlike other encoders that require the query and context to be fed into separate encoder models, the cross-encoder requires the query and context to be concatenated and input into a single encoder model. This allows the encoder to better capture the relationship between the query and the context. After obtaining the encoder output, the representation of the `[CLS]` token is taken and passed through a fully connected layer to get the relevance score between the query and the context. Similarly, given the cross-encoder based reranker model, we define the reranker function $\mathbf{Reranker} : \mathcal{T} \mapsto \mathbb{R}$. We directly concatenate the same query q_j and the retrieved context $\hat{d}_l$, then obtain the score:

$$RerankScore = \mathbf{Reranker}([q_j; \hat{d}_l]).$$

We then select the top $\mathbf{k}$ ($\mathbf{k} \leq \mathbf{n}$) contexts with the highest $RerankScore$, denoted as $\mathcal{D}^* = \{d_l^* | l \in [0, k)\}$.

3.2.3 Type Correcting

After retrieving the entity-related contexts, we can use these contexts to correctly correct the entity e_j^* to its entity type. We treat this as an ICL-based QA task. We use a frozen LLM $Corrector_{contexts}$ as the type correcting model and construct a prompt using x_i, e_j^*, D^*, and T_i to instruct the LLM in performing the type correcting task and obtain the final type t_j^* to the entity e_j^*:

$$t_j^* = Corrector_{contexts}(x_i, e_j^*, \mathcal{D}^*, T_i).$$

For each $e_j^* \in E_i^*$, we apply the above process in batches to correct the type of each extracted entity, thereby obtaining the final output y_i^{final} of our framework.

4 Experiment

This section presents experiments conducted under zero-shot settings to validate the effectiveness of **IRRA**. The details of the experimental settings are described in the following parts.

Tokens	AI	Literature	Music	Politics	Science
512	73964	2353876	2427757	2444698	1334738
1024	33772	1158747	1144115	1144115	624434

Table 1: The number of contexts in each domain of the database under different maximum token length processing.

4.1 Implement of IRRA

- **Backbones:** We use Llama3-8B as the backbone model for the entity extraction stage and the type correcting model. Llama3-8B aligns well with human intent and is better at understanding and responding to user requests.

- **Training details:** We directly use the trained Llama3-8B provided by the IEPile (Gui et al., 2024) as the base extractor. The IEPile dataset collects a large number of NER datasets and includes some modifications to the original NER datasets. Training for all extractors is performed using supervised instruction-based LoRA (Hu et al., 2021) fine-tuning with the LlamaFactory library (Zheng et al., 2024) on an Nvidia A800 GPU.

- **Database:** We use Chroma (Chroma, 2024) as a vector database for retrieval. Due to the high costs associated with using search engine interfaces as a retrieval tool, we collected a large amount of raw text from sources similar to those in the CrossNER dataset to serve as the data source for our vector database. Each text is sliced into multiple smaller contexts with a maximum length of not more than **512** tokens. The end of each context is a period or new-line character to ensure that the context is not split in the middle of a sentence. All the contexts are indexed and stored in the vector database. The number of documents is shown in Table 1.

- **Retrieval:** For retrieval, we adopt BCEmbedding (NetEase Youdao, 2023), a state-of-the-art model for context embedding, to obtain context embeddings. We first select the top **n = 20** contexts, then we use BCEReranker (NetEase Youdao, 2023), which is usually compatible with BCEmbedding, to score and filter the retrieved n contexts, selecting the top **k = 1** contexts.

4.2 Evaluation Metrics

We mainly focus on the zero-shot evaluation. We adopt CrossNER (Liu et al., 2020) benchmark following IEPile's settings. CrossNER provides datasets that spans multiple domains, including AI, Literature, Music, Politics, and Science which is suited for evaluating zero-shot NER tasks. We mainly report the micro-F1 score as it's a widely used metric in Named Entity Recognition.

4.3 Comparsion Baselines

we use the following methods as baselines:

- **InstructUIE** (Wang et al., 2023b) InstructUIE improves performance by learning multiple similar tasks. We use the best settings in the corresponding paper as the comparison.

- **ChatGPT** (OpenAI, 2024) We use the zero-shot settings on ChatGPT directly.

- **UniNER** (Zhou et al., 2024) This method investigates targeted distillation combined with mission-focused instruction tuning to train student models. These models not only excel in a wide range of applications but also significantly outperform general large language models. We directly use the best scores provided in the paper for comparison.

Method	Extra Training Data	AI	Literature	Music	Politics	Science	Average
InstructUIE(2023)	✓	49.0	47.2	53.2	48.2	49.3	49.4
ChatGPT(2023)	✗	54.4	54.0	61.2	59.1	63.0	58.3
UniNER(2023)	✓	54.2	60.9	64.5	61.4	63.5	60.9
GoLLIE(2024)	✓	**61.6**	**62.7**	68.4	60.2	56.3	61.8
IEPile-Baichuan2-13B(2024)	**IEPile**	48.5	43.6	57.0	51.4	49.8	50.1
IEPile-Llama3-8B(2024)	**IEPile**	49.7	43.0	53.9	56.9	50.4	50.8
IEPile-Qwen1.5-14B(2024)	**IEPile**	55.6	44.3	58.7	54.6	51.4	52.9
OneKE(2024)	**IEPile**	-	-	-	-	-	60.9
IRRA(ours)	**IEPile**	57.5	59.3	**69.4**	**74.0**	**68.3**	**65.7**

Table 2: The micro-F1 scores on zero-shot cross-domain setting. All settings for baselines are selected by the highest score in the corresponding papers. InstructUIE, UniNER and GoLLIE use extra training data from different sources given in their paper. Since ChatGPT does not provide the training APIs, no extra training data is used. IEPile, OneKE and our method use IEPile's training set as extra training data.

- **GoLLIE** (Sainz et al., 2024) GoLLIE provided some Large Language Models trained to follow annotation guidelines. We follow all the settings in this paper completely, and for the results obtained from different models, we directly choose the highest-scoring result, even if the model used is larger than ours.

- **IEPile** This method builds a large dataset to train the model's information extraction capabilities. Since our method uses this dataset for training, its settings are close to ours.

- **OneKE** (Gui et al., 2024) OneKE uses "Schema-based Polling Instruction Construction" method in instruction fine-tuning. We directly use the best scores provided in the paper for comparison.

4.4 Main Results

We report the micro-F1 scores for each domain in CrossNER, as well as the average micro-F1 score, as shown in Table 2. Except for ChatGPT, all methods require extra training data to train the models. Among all the generative LLM-based methods, **IRRA** achieves the highest average micro-F1 score, showing an nearly 4% improvement over the previous SOTA. Additionally, we report the performance of using three different LLMs as backbones for the IEPile method. Consistent with **IRRA**'s results, IEPile also scores lower in the AI and Literature domains, which may be due to internal biases in the extra training data. We also note that using Qwen1.5-14B as the backbone for IEPile results in a relatively higher score in the AI domain, likely due to its broader internal knowledge in AI. The SOTA scores for the AI and Literature domains come from GoLLIE. Although our method is slightly weaker than GoLLIE, the model used by GoLLIE is significantly larger than ours (34B *vs.* 8B). However, even with such a large difference in model size, **IRRA** still achieves higher scores in other domains and a higher average score compared to GoLLIE, which may be due to the compensatory effect of incorporating external knowledge.

5 Analytical Experiment

5.1 Ablation Analysis

To analyze the specific parts of the **IRRA** framework, we design several different pipeline settings. For each setting, we conduct the analysis from three dimensions: recall, precision and F1 score.

We report the performance of the base extractor trained solely on IEPile (**Base**) for direct entity extraction. Additionally, we compare two methods in the type correcting stage: one based on inserting annotation guidelines to the prompt **(Guidelines)** and another based on inserting retrieved contexts to the prompt **(Contexts)**.

Entity Extracting	Type Correcting	Recall	Precision	F1
Base	✗	49.1	54.0	50.8
Base & Enhanced	✗	**69.4**	49.5	57.4
Base	Contexts	53.6	**72.8**	61.3
Base & Enhanced	Guidelines	59.0	63.8	61.2
Base & Enhanced	Contexts	63.4	68.5	**65.7**

Table 3: Performance of different entity extracting methods and type correcting methods across all domains. This comparative analysis experiments are all based on Llama3-8B. All other experimental settings are consistent with the main experiments.

As shown in Table 3, incorporating external knowledge to correct the base model's extraction results improves performance as expected. However, applying retrieval-based correction to the enhanced model's extraction results leads to a decrease in recall. This is likely because the recall-enhanced model identifies more challenging entities, which are prone to being incorrectly modified during the correction stage. These errors result in an increase in false negatives (FN) and a corresponding decrease in true positives (TP).

5.2 Effect of Vanilla Models

To verify that our method works with different LLMs, we use Qwen1.5-14B (Bai et al., 2023) and Baichuan2-13B (Baichuan, 2023) as additional backbone models and type correcting model in our framework. All of these models are trained on the IEPile dataset and our perturbed dataset and used as base extractor and enhanced extractor. The results from different models using the IEPile method are shown in Table 2.

Entity Extracting	Type Correcting	Recall	Precision	F1
Baichuan2-13B	Baichuan2-13B	51.9	52.1	51.9
Baichuan2-13B	Llama3-8B	58.5	61.8	60.0
Qwen1.5-14B	Qwen1.5-14B	62.0	65.0	63.4
Qwen1.5-14B	Llama3-8B	62.6	66.0	64.2
Llama3-8B	Llama3-8B	63.4	68.5	65.7

Table 4: Performance of different combinations of models in two stages of **IRRA** over all domains. During the retrieval and type correcting stages, we set the maximum length of the retrieved context to **512** tokens and the number of contexts utilized by the model to **1**.

We use different combinations of vanilla models within our framework, with the results presented in Table 4. The results indicate that our method improves performance across different vanilla models, with the most significant improvement observed on Llama3-8B, while the improvement is less pronounced on the larger Baichuan2-13B model. From Table 2, we can observe that even without using our method, Baichuan2-13B is still slightly weaker than other models. Based on our observations, Baichuan2-13B is weaker in following instruction, whereas Llama3-8B shows superior performance.

5.3 Effect of Top Documents

We conducted an ablation study by evaluating the performance of our model when using the top-1, top-2, and top-3 highest-scored documents as additional knowledge during inference. The results are shown in the table 5:

Top Documents	Recall	Precision	F1
3	62.8	67.2	64.8
2	63.1	68.0	65.3
1	63.4	68.5	65.7

Table 5: The number of documents retrieved is controlled by the parameter **k**.

In line with human intuition, the performance of the model shows a gradual decline with the increase of **k**. This indicates to some extent that the quality and quantity of retrieved documents have an impact on the performance of the model.

5.4 Generality of Framework

To further demonstrate the validity of our approach, we propose a 10-ways-0-shots benchmark on the Few-NERD (Ding et al., 2021) dataset, a commonly used dataset for Few-shot NER that contains a large number of fine-grained entity types. Under this setting, there are 10 random fine-grained entity types in the schema of each sample for extraction, and annotated samples are not allowed to be used as prompts. At the same time, we test several different base models and our methods on this benchmark:

Method	Recall	Precision	F1
IEPile-Baichuan2-13B(2024)	31.3	67.3	42.7
IEPile-Qwen1.5-14B(2024)	47.9	70.3	57.0
IEPile-Llama3-8B(2024)	29.6	68.3	41.3
OneKE(2024)	54.1	66.5	59.6
IRRA(ours)	56.0	68.0	61.4

Table 6: We use the test set of supervised partition (a few-nerd default partition) for our experiments.

We followed exactly the same configuration as the main experiment, which means that we choose Llama3-8B as the vanilla model. From the results, we can see that our approach can significantly improve the performance of the vanilla model, while also outperforming other basic models. Since this dataset is not divided by domains, we choose to use Wikipedia as an additional source of knowledge, which may limits the performance of our method.

5.5 Correction Analysis

As shown in Figure 3, we observe that when the type of the entity extracted in the previous stage is unique and correct, the performance of both methods is similar. This similarity may be due to that the correcting model is more confident with easily

judged samples. However, both methods introduce some noise, leading to recognition error, especially the one based on retrieved contexts. Consider the entity "Apple" in a text about fruit. Using retrieved contexts to correct the entity type may retrieve information from technology-related contexts, leading the model to incorrectly match "Apple" to company rather than fruit.

Similar to human intuition, when the extraction model struggles to determine the specific type or completely mismatches the type, introducing the related external knowledge can achieve better improvement. This indicates that the model may lack the necessary contexts to judge the type when extracting the entity.

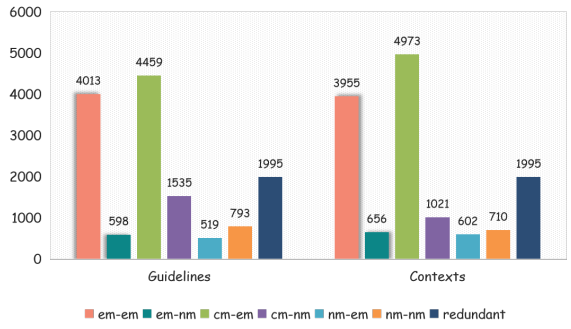

Figure 3: Comparison of correction based on annotation guidelines and correction based on retrieved contexts. **em** indicates that the extracted entity's type exactly matches the ground truth type. **cm** indicates that among the multiple types for the extracted entity, at least one matches the ground truth type. **nm** indicates that none of the types for the extracted entity match any ground truth types. **redundant** indicates that the entity should not be extracted.

5.6 Redundant Entities

From Figure 4, we can see that while our method recalls more correct entities, it also retrieves more redundant entities. For the domain-specific types, our method recalls fewer redundant entities, indicating that it is more cautious with less common types. For the common types, the number of redundant entities retrieved by our method increases, suggesting a more lenient approach toward common types. As for the miscellaneous types, our method recalls a large number of redundant entities, which may be due to the unclear definition of redundant entities. This kind of types also exhibits annotation bias in human labeling. For zero-shot tasks, handling the miscellaneous types remains a challenge.

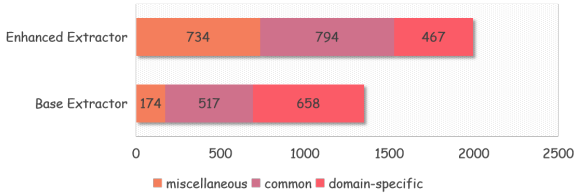

Figure 4: Type cases in redundant recalled entities. **common** represents the type common to all domains such as country, location, organisation and person. **domain-specific** represents the type specific to a domain, and **miscellaneous** represents the type that does not belong to the other two types. We counted the types of entities that were eventually recognized by the model.

6 Conclusion

In this paper, we propose a two-stage **IRRA** framework for zero-shot Named Entity Recognition (NER), strategically leveraging large language model techniques. The core idea of IRRA is to supplement missing information in entity extraction and recognition through the *Recall Augmented Entity Extracting* and *Retrieval Augmented Type Correcting* stages. Experimental results show significant improvements over state-of-the-art approaches, highlighting the efficacy of multi-stage processing in large model applications.

Limitations

This work has two main limitations:

(1) Due to the prohibitive expense of commercial search APIs, our document retrieval relies on cost-efficient alternatives (e.g., local indexes), which may limit result quality compared to industry-grade systems.

(2)The pipeline's sequential structure prevents dynamic cross-module interactions (e.g., feedback loops), potentially capping performance gains. Future work could explore interactive module designs.

Acknowledgments

This research was supported in part by National Science and Technology Major Project (2021ZD0111501), National Science Fund for Excellent Young Scholars (62122022), Natural Science Foundation of China (62406078, 62476163, U24A20233), Guangdong Basic and Applied Basic Research Foundation (2023B1515120020).

References

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *Preprint*, arXiv:2306.04136.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional ai: Harmlessness from ai feedback. *Preprint*, arXiv:2212.08073.

Baichuan. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

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

Chroma. 2024. Chroma: An open-source ai application database. https://www.trychroma.com. Accessed: 2024-07-13.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Hai-Tao Zheng, and Zhiyuan Liu. 2021. Few-nerd: A few-shot named entity recognition dataset. *Preprint*, arXiv:2105.07464.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. Critic: Large language models can self-correct with tool-interactive critiquing. *Preprint*, arXiv:2305.11738.

Honghao Gui, Lin Yuan, Hongbin Ye, Ningyu Zhang, Mengshu Sun, Lei Liang, and Huajun Chen. 2024. Iepile: Unearthing large-scale schema-based information extraction corpus. *CoRR*, abs/2402.14710.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. *Preprint*, arXiv:2310.01798.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Preprint*, arXiv:2112.09118.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

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

Sosuke Kobayashi. 2018. Homemade bookcorpus. https://github.com/soskek/bookcorpus.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2020. Crossner: Evaluating cross-domain named entity recognition.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,

Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Inc. NetEase Youdao. 2023. Bcembedding: Bilingual and crosslingual embedding for rag. https://github.com/netease-youdao/BCEmbedding.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2024. Is self-repair a silver bullet for code generation? *Preprint*, arXiv:2306.09896.

OpenAI. 2024. Chatgpt. https://www.openai.com/chatgpt. Accessed: 2024-07-13.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *Preprint*, arXiv:2308.03188.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. REFINER: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126, St. Julian's, Malta. Association for Computational Linguistics.

Gabriele Picco, Marcos Martinez Galindo, Alberto Purpura, Leopold Fuchs, Vanessa Lopez, and Thanh Lam Hoang. 2023. Zshot: An open-source framework for zero-shot named entity recognition and relation extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 357–368, Toronto, Canada. Association for Computational Linguistics.

S. Robertson. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. GoLLIE: Annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations*.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023a. Shepherd: A critic for language model generation. *Preprint*, arXiv:2308.04592.

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*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. *Preprint*, arXiv:2211.00053.

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

Tingyu Xie, Jian Zhang, Yan Zhang, Yuanyuan Liang, Qi Li, and Hongwei Wang. 2024b. Retrieval augmented instruction tuning for open ner with large language models. *Preprint*, arXiv:2406.17305.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

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

A Appendix

A.1 Prompts

Syno-Hypernyms generation prompt is specifically designed and employed to generate a comprehensive set of synonyms and hypernyms for each entity type. This process ensures that for every entity type, a robust and contextually relevant collection of alternative expressions is created.

Prompt 1: Synonym generation prompt

Complete each word's list with multiple possible synonyms and hypernyms.

```
{
 "entity type 1": ["entity type 1"],
 "entity type 2": ["entity type 2"],
 ...
}
```

Extraction prompt is utilized to guide the base extractor. It directs the models to accurately extract entities from the provided text in accordance with the predefined schema. This prompt ensures that the extraction process is aligned with the specific

requirements of the schema, thereby improving the quality of the extracted results.

Prompt 2: Extraction prompt

```
{
 "instruction": "Please extract
     entities that match the schema
     definition from the input. Return
      an empty list if the entity type
      does not exist. Please respond
     in the format of a JSON string.",
 "schema": [list of entity types],
 "input": "input sentence string"
}
```

Complement prompt is used to guide the enhanced extractor. It directs the model to expand and complement the initial recall results.

Prompt 3: Complement prompt

```
{
 "instruction": "Modify the extracted
     entities based on input and
     schema. If you think there may be
      some entities missing or that
     you don't understand something,
     please list them as entities.",
 "schema": [list of entity types],
 "input": "input sentence string",
 "entities": {
  "eneity type 1": [list of entities],
  "eneity type 2": [list of entities],
  ...
  }
}
```

Correcting prompt based on retrieved contexts is employed to guide the model in correcting entities and their types based on the retrieved contexts. This prompt ensures that the model accurately corrects each entity and its corresponding type.

Prompt 4: Correcting prompt based on retrieved contexts

 # Instruction
You need to answer the **type** of the corresponding entity in the text according to the **Types** and **the relevant contexts retrieved**.
 # Text
{text}
 # Entity
{entity}
 # Relevant contexts
{contexts}
 # Types
[list of entity types]
 # Rules
Guidance information needed when extracting entities.
 # Answer format

```
{
 "entity": "the entity in Entity",
 "reason": "The reason you judge the
     type which from Types",
 "entity type": "type from Types"
}
```

Prompt 5: Correcting prompt based on annotation guidelines

 # Instruction
You need to answer the **type** of the corresponding entity in the text according to the **Types** and **guidelines**.
 # Text
{text}
 # Entity
{entity}
 # Guidelines
Annotation guidelines about the data.
 # Types
[list of entity types]
 # Rules
Guidance information needed when extracting entities.
 # Answer format

```
{
 "entity": "the entity in Entity",
 "reason": "The reason you judge the
     type which from Types",
 "entity type": "type from Types"
}
```

Correcting prompt based on annotation

guidelines is employed to guide the model in correcting entities and their types based on predefined annotation guidelines. This prompt also ensures that the model accurately corrects each entity and its corresponding type.