A Redundancy-Enhanced Framework for Error Correction in Named Entity Recognition

Anonymous ACL submission

Abstract

We present a redundancy-enhanced framework for error correction of NER by incorporating related sentences from the internet. Our key contribution is a Transformer-based refiner that integrates additional information into pre-trained language model with minimal effort. We begin by forming a redundancy set composed of (i) related sentences to the target sentence, using a proposed retrieval pipeline, and (ii) their NER predictions from an external Named Entity tagger. We then construct this refiner by combining a pre-trained Transformer-based model with an NE-tag embedding layer, both of which are fine-tuned on the target sentences and their corresponding redundancy sets. Methodologically, we propose a branch-and-conquer learning paradigm, termed *Incremental Learning*, for accurate error correction. In particular, it delivers an error reduction of 4.48% and a new state-of-the-art performance of 61.43 micro-f1 score on realistic WNUT17 dataset.

Table 1: **Implicit and Explicit Redundancy.** Both implicit and explicit redundancy refer to the information that "Joe Biden is a person NE", with the explicit one associated with explicit NE annotation ([span]$_{\text{type}}$, meaning the span and type of that NE).

Type	Example
Implicit Redundancy	1. Biden made an announcement. (news article) 2. We are happy to have President (transcript of the press conference) Biden here. (comments in the webpage) 3. Joe Biden is the U.S. president. (Wikipedia hyperlinked)
Explicit Redundancy	1. [Biden]$_{\text{person}}$ made an announcement. 2. We are happy to have [President Biden]$_{\text{person}}$ here. 3. [Joe Biden]$_{\text{person}}$ is the U.S. president.

1 Introduction

In this study, we present a novel approach to error correction in Named Entity Recognition (NER) by introducing a redundancy-enhanced framework that incorporates relevant sentences from the internet. The term 'redundancy' within the NER context refers to the phenomenon where the same Named Entity (NE) appears multiple times within a single paragraph, across different sections of a document, or within a collection of texts. This concept is exemplified in Table 1, where we illustrate redundancy through a real-world scenario. For instance, in a news article reporting an announcement by President Joe Biden, the NE 'Joe Biden' is not only mentioned in the main article but also redundantly appears in associated texts such as the transcript of the related press conference, comments on a Twitter post about the event, and a Wikipedia article linked within the news piece. These various instances across different sources demonstrate the redundancy of the NE 'Joe Biden', underscoring the central theme of our study."

In this paper, we divide previous work that uses redundant information explained above into two categories: *One-Stage approach* and *Two-Stage approach*, which *implicitly* and *explicitly* incorporate redundancy into their model, respectively. The One-Stage approach, typified by BERT-NER, processes multiple sentences in a single model to enhance context awareness as well as other methods such as hand-crafted features, non-local feature extractors, and virtual memory usage. On the other hand, the Two-Stage approach first extracts the *explicit* NE information, i.e., the NE tags of the given sentence, extracted by a vanilla NER at the first stage. The second stage—an *NE refiner*—then performs NE error correction, using broader and non-local information, both text and NE predicted labels (cf. Appendix D for more details).

This information-scarce problem manifests itself as three challenges in practice for NER:

Context Dependency: Despite advancements in language modeling, systems still struggle with sentences where little contextual information is provided to identify associated Named Entities (NEs) (Wu et al., 2020; Li et al., 2020).

Uncommon Language Usage: These systems have difficulty handling sentences that contain uncommon language uses, such as abbreviations, rare, or unseen NEs, making recognition challenging

(Derczynski et al., 2017).

Data Sparsity: The performance of machine learning models is highly dependent on the quality and quantity of the training data. In the case of uncommon or rare NEs, the models might not have enough examples during training to generalize well to these entities in unseen data.

This paper presents a transformer-based refiner that can effectively incorporate supplementary information into any pre-trained language model. The initial stage of this process involves the establishment of a redundancy set, which includes sentences related to the target sentence and their corresponding NER predictions. These predictions are generated using an external NER tagger. The refiner is constructed by combining a pre-trained transformer-based model with an additional NE-tag embedding layer, which can take as input not only the tokens but also the tag predictions.

Furthermore, we introduce a novel approach to learning called *Incremental Learning*. This approach segments the task of error correction into distinct sub-tasks, focusing on two abilities to learn: (1) recognizing NEs (A_1) and (2) making refinement by selecting related NEs and copying the predictions (A_2). By employing various text data augmentation methods for *Incremental Learning*, the model is enabled to progressively learn these sub-tasks through these two pre-training stages. Our empirical evaluations demonstrate the effectiveness of our approach. We observed a significant reduction in errors and an improvement in the micro-f1 score, surpassing the performance of the state-of-the-art models. These evaluations were conducted on the realistic WNUT17 dataset (Derczynski et al., 2017). The source code is shared in this GitHub repository [1].

Contributions are as follows:

- We propose to isolate the ability of error recovery (A_2) and the ability of NER (A_1) to facilitate effective training, where an explicit majority-voting approach and a transformer-based approach are proposed, respectively.
- We propose *Incremental Learning* and utilize *Data Augmentation* techniques to allow gradual learning with specific smaller and easier tasks for the refinement model.
- We propose a pipeline that retrieves related sentences that are more beneficial for NER.
- We conduct experiments to show the superiority of our proposed approach delivering the 61.43 F1 score and the 4.48% error recovery rate, showing a 1.62% improvement compared to the previous state-of-the-art.

[1] https://github.com/***

2 Proposed Operational Flow

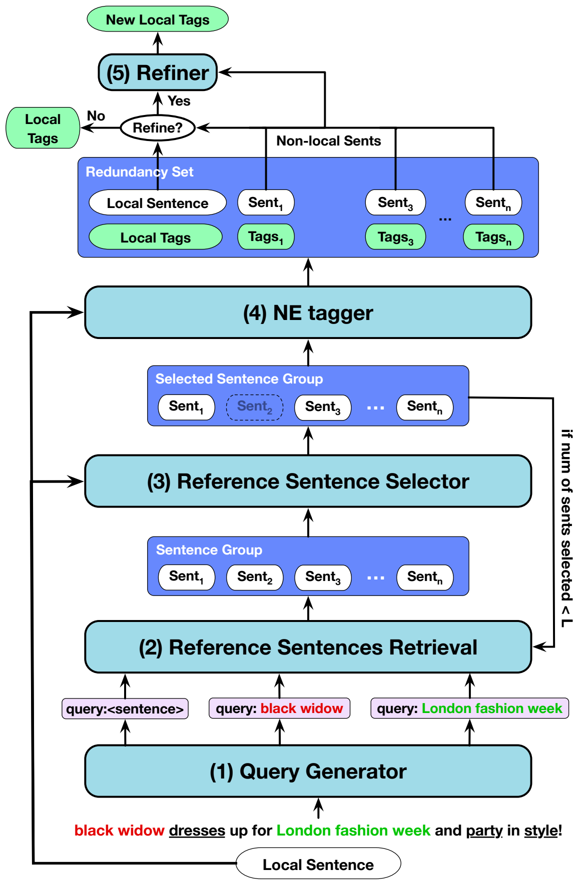

Figure 1: **Proposed Operational Flow.** It consists of six stages: **(1)** *Query Generator*, which takes in one sentence and generates query texts, **(2)** *Reference Sentences Retrieval*, which retrieves a number of sentences for each generated query, **(3)** *Reference Sentence Selector*, which selects the related reference sentences based on the local sentence, **(4)** *NE Tagger*, which generates the original NE predictions, and **(5)** *Refiner*, which will be triggered if at least one non-local sentence is retrieved, and will take in the token and tag sequence and outputs the refined tags on the local sentence.

As shown in Figure 1, we propose a framework with five modules: (1) *Query Generator*, generating queries from the sentence; (2) *Reference Sentences Retrieval*, retrieving related sentences from the given documents, Wikipedia, and the Internet, separately; (3) *Reference Sentence Selector*, scoring and selecting the valuable sentences; (4) *NE Tagger*; and (5) *Refiner*, conducting refinement with the help of the related reference sentences. Each module will be further elaborated as follows.

2.1 Query Generator

This tagger converts the local sentence into a list of queries, and then pass it to the next module. For example, two spans 'black widow' and 'london fashion week' are identified in Figure 1. We will train a mention detector to identify such spans. After detecting all spans in the given/local sentence, we additionally invoke a query with each span in addition to the original local sentence as a specific query. Therefore, a sentence with M spans will issue total M+1 different queries (including the query with the whole sentence). For example, the sentence in Figure 1 will issue three queries: 'black widow', 'london fashion week' and the one from the entire sentence. Formally,

$$Q = \mathrm{QG}(s) = \{s\} \cup \mathrm{MD}(s) \tag{2.1}$$

where QG represents the module of Query Generation, s means the local sentence, MD represents a mention detector giving a set of mentions, and Q represents the set of all queries generated. The implementation is based on transformer framework, and for more details please refer to Appendix I.

2.2 Reference Sentences Retrieval

In each query turn, the given query will be sent to Google Search Engine[2] to crawl over all websites on the Internet. We take the title and the snippet of the first 100 (if any) returned results, which provides abundant reference sentences related to the local sentence. Formally,

$$\mathrm{NS} = \mathrm{RSR}(q_1) \cup \cdots \cup \mathrm{RSR}(q_{M+1}), \forall q_i \in Q \tag{2.2}$$

where RSR represents the module of *Reference Sentences Retrieval*, NS represents the set of all related sentences, (i.e., non-local sentences), retrieved for all queries q_i in Q. In addition, we adopt a recursive retrieval[3] whic aims to retrieve at least one sentence to maintain an adequate level of redundancy. This strategy is particularly useful when the next module, Section 2.3, is overly selective. The implementation details are included in Appendix I.

2.3 Reference Sentence Selector

As the third module in the entire system, it serves a pivotal role, as illustrated in Figure 1. This module operates as a selective filter, determining which sentences from the preceding module are retained and forwarded to the subsequent modules Sections 2.4 and 2.5 for recovery. More specifically, it takes as input the local sentence and each non-local sentence, and determines which non-local sentences should be retained. Formally, a specification of this module would be:

$$\begin{aligned} &\mathrm{RSS}(s, ns) \in \{\text{keep}, \text{discard}\} \\ &\mathrm{NS}' = \{ns \in \mathrm{NS} \mid \mathrm{RSS}(s, ns) = \text{keep}\}, \end{aligned} \tag{2.3}$$

where RSS represents a model that performs such a function to output either the decision "keep" or "discard" based on the local sentence s, and one non-local sentence ns. Only the non-local sentences with "keep" decision will be kept and passed (denoted as NS′) to the next module. Note that we specifically build this module with high precision to have high-quality non-local sentences. In addition, combined with the recursive retrieval mentioned in the last module, Section 2.2, a number of high-quality non-local sentences could be expected. The module is implemented in transformer framework, and please refer to Appendix I for more details.

2.4 Name Entity Tagger

In order to collect redundant NE information, an NE tagger is required to identify NEs for both the local and non-local sentences so that correlated NEs from the non-local sentences may help to recover potential errors in the local NEs. Note that this module corresponds to the vanilla NER ability, A_1, as mentioned in Section 1. Formally,

$$\mathrm{TS} = \{\mathrm{NT}(s)\}, \forall s \in \mathrm{S} \tag{2.4}$$

$$\mathrm{TNS}' = \{\mathrm{NT}(ns')\}, \forall ns' \in \mathrm{NS}' \tag{2.5}$$

where NT is the NE tagger, which outputs the tags of each token, TS and TNS′, of the local sentence S and filtered non-local sentences NS′, respectively. The implementation details are included in Appendix I.

2.5 Refiner

Refiner is a key module in the whole system to perform recovery action based on the local sentence and the reference sentences. This module will be triggered if at least one non-local sentence is selected from *Reference Sentence Retrieval* or no action will be performed otherwise. Formally, it can be represented as

$$\mathrm{RD} = \{\mathrm{S}, \mathrm{TS}, \mathrm{NS}', \mathrm{TNS}'\} \tag{2.6}$$

$$\mathrm{TS}' = \mathrm{RF}(\mathrm{RD}) \tag{2.7}$$

where RD is the formed redundancy set by the local sentence S and the selected non-local sentences NS' and their corresponding tag predictions TS

[2]https://www.google.com

[3]By default, Search Engine can only provide a fixed number of Top-N results for each query in one "page". To ensure enough level of redundancy, we will navigate to the following pages to obtain more results, which is termed as "recursive retrieval" in our work.

and TNS′ respectively. RF is a model which predicts new refinement tags, TS′, for the tokens of the local sentence. These refinement predictions are based on the information given in the whole redundancy set RD. Specifically, this is related to the model learning of the recovery ability, A_2 as mentioned in Section 1. More details are included in Sections 3.2 and 4.

3 Proposed Framework for Refinement

This section explains the implementation of the module *Refiner* introduced in Section 2.5. Specifically, we explain the two types: (1) Majority-Voting Approach, and (2) transformer-Based Approach. For other implementation details of modules in Sections 2.1 to 2.3, please refer to Appendix I.

3.1 Majority-Voting Refiner (ReMV)

In our work, we have developed a rule-based refiner employing a majority voting algorithm, drawing inspiration from the approach by Yangarber and Jokipii (2005), yet tailored for NER tasks. The core mechanism of this refiner involves creating a 'redundancy set' by clustering NEs from a target sentence along with its related sentences. This clustering hinges on the detection of shared content words.

The crucial aspect of this refiner is its labeling methodology. For each unique NE identified within a cluster, labels are assigned through a majority voting process. We explore two variations of this refiner: the standard **ReMV**, which assigns unique NE types as labels, and the enhanced **ReMV (+span)**, where labels consist of unique combinations of spans and NE types.

Furthermore, we introduce an 'oracle method' (**Oracle**), which represents the theoretical maximum performance one could achieve with this approach. This method is based on the **ReMV (+span)** variant but differs in its use of 'pseudo gold' reference sentences. These sentences are designated as 'keep' during the benchmark generation process, aligning with a weakly supervised learning approach.

For an in-depth understanding of this process, including its algorithmic foundations, please refer to Algorithm 1 for detailed pseudo code.

3.2 Transformer-Based Refiner (ReTRF)

The transformer-Based Refiner, **ReTRF**, is a masked sequence tagger inspired by the architecture proposed by Wang et al. (2021). It consists of a transformer encoder layer, a classification layer, and a CRF layer. The transformer encoder itself comprises a pre-trained embedding layer (including word and positional embeddings) and a multi-head self-attention block. Unique to our approach is the addition of a custom, trainable embedding layer for NE tag embeddings. These embeddings, which represent NE tags as d-dimensional vectors, are combined with the original embeddings to enhance the model's understanding of NE contexts.

ReTRF processes two primary inputs. The first is a concatenated token sequence, structured as $\mathrm{Concat}([s; \langle \text{SEP} \rangle; ns_1; ns_2; \ldots])$, where '$\langle \text{SEP} \rangle$' marks the end of the local token sequence, and $ns_i \in NS'$ represents non-local sentences. The second input is a corresponding tag sequence, formatted similarly, and also using the '$\langle \text{SEP} \rangle$' symbol to separate the concatenated sentences as $\mathrm{Concat}([ts; \langle \text{SEP} \rangle; tns_1; tns_2; \ldots])$. Both inputs are converted to embeddings via a lookup matrix. The embeddings (including both transformer and tag embeddings) are then summed at each sequence position and fed into the transformer Encoder.

The refinement process culminates with the generation of refined tag predictions by the last layer of the module. During fine-tuning, a loss is calculated based on benchmarks in the refinement dataset (refer to Appendix G for more details), and gradients are propagated accordingly.

To facilitate efficient training and fine-tuning, we ensure a robust initialization of the tag embeddings. Rather than starting from a random initialization, we pre-train the tag embeddings on a simple NE tagging task (as depicted in Figure 2). The weight matrix of the last classification linear layer of the NE tagger, which aligns with the shape of the NE tag embedding lookup matrix, serves as the basis for this pre-training (both shaped as N tag embeddings with dimension d, where N represents the total number of possible NE tags, and d signifies the embedding dimension).

4 Proposed Data Augmentation and Incremental Learning for Refinement

This paper presents an approach to creating an NER error recovery dataset, crucial for training and evaluating our refinement model. Based on the N-fold cross-validation method by Krishnan and Manning (2006), we generate a dataset that includes NER tags predicted by a first-stage NE tagger and ground truth tags, as shown in Appendix G.

An example of the generated data by Krishnan and Manning (2006) is shown in the

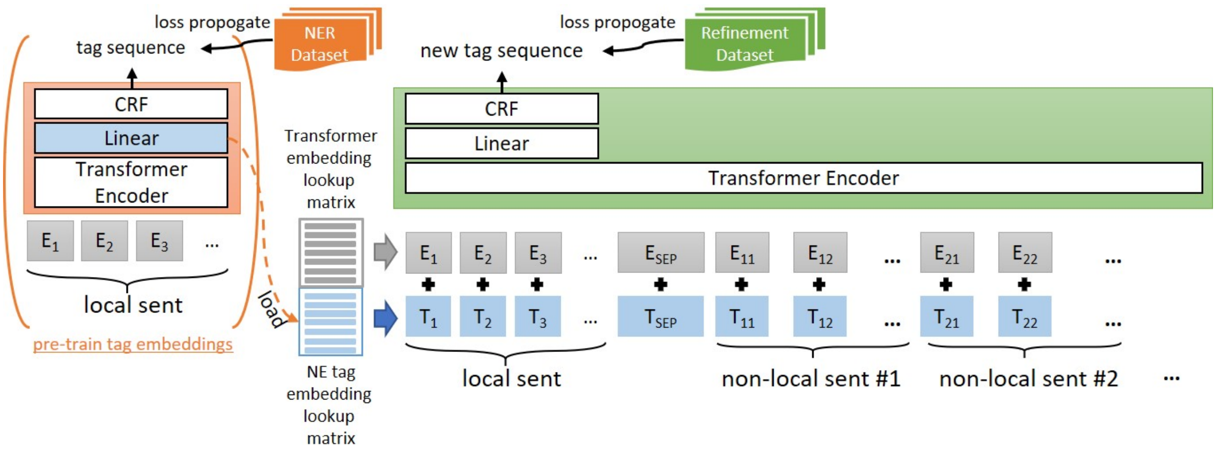

Figure 2: **Proposed transformer-Based Refiner.** This refiner features a pre-trained transformer embedding layer, an encoder layer, a classification layer, and a CRF layer (Lafferty et al., 2001). It processes token and tag sequences derived from concatenated local and selected non-local sentences. The architecture initializes tag embeddings using the last classification layer of a trained NE tagger, enhancing parameter initialization. The transformer and tag embedding lookup matrices transform the token and tag sequence into corresponding embeddings, respectively. Notably, E_i and T_i represent the i-th token and tag embedding in the local sentence, while E_{jk} and T_{jk} denote the k-th token and tag embedding in the j-th non-local sentence. Summed token and tag embeddings are input to the transformer Encoder for final sentence refinement, with non-local sentence positions masked during loss calculation.

⟨original_example⟩ row of Table 2, which shows the local sentence "let's go!! black widow!" with original NE predictions of "black widow" as a non-entity and "black widow" as a person in WNUT17 benchmark, and three retrieved non-local sentences and their NE predictions.

While Krishnan and Manning (2006) provides a foundational method for generating training data, our study enhances this with data augmentation and *Incremental Learning*. We break down the recovery ability into three sub-abilities: copying local predictions (A_{21}), copying non-local predictions (A_{22}), and matching/selecting related non-local sentences (A_{23}). Training data for each sub-ability are generated and used sequentially in the curriculum ($A_{21} \rightarrow A_{22} \rightarrow A_{23}$).

The integration of Incremental Learning is pivotal, addressing the challenge of combining richly informed pre-trained token embeddings with newly trained tag embeddings. This phased approach prevents the model from over-relying on token embeddings, ensuring a balanced utilization of both embeddings. Without Incremental Learning, our experiments show that tag embeddings remain underutilized, leading the model to function more as an additional NE tagger rather than a refinement tool.

Local Predictions Copying (A_{21}) The model learns to retain the original predictions from the local scope when no non-local references are available, essentially mirroring the existing local predictions without modification. Data for this sub-ability are generated by excluding non-local sentences to focus on retaining original predictions. To prevent over-reliance on textual embeddings and diversify learning, examples with modified NE types are also created. This is foundational for refinement as it prevents the model from arbitrarily generating new tags, which would signify a re-implementation of NER rather than refinement or majority voting.

Non-local Predictions Copying (A_{22}) The objective of A_{22} is to train the model to effectively copy predictions from the non-local scope, under the assumption that these non-local sentences are accurate. This phase is crucial as it teaches the model to utilize external references, a skill not addressed in A_{21}. For A_{22}, we specifically select 'helpful' non-local sentences that contain gold NEs in their predicted NE tags (e.g., the sentence "[black widow]$_{\text{person}}$ is a female character", since it has the gold NE "[black widow]$_{\text{person}}$"). By focusing on these reliable sentences, the model learns to copy non-local predictions, a step that is distinctly different from A_{21}'s focus on local predictions. This stage prepares the model for the subsequent and more complex task of matching and selecting the appropriate non-local sentences for final prediction in A_{23}. Similarly, more examples can be created by altering the type of the predicted local NEs.

Matching and Selecting (A_{23}) A_{23} represents a critical advancement in our Incremental Learn-

Table 2: **Stages of Incremental Learning in NER Refinement.** This table presents the stages A_{21}, A_{22}, and A_{23} using the example "black widow dressed up for london fashion week and party in style". A_{21} is centered on learning to copy the local predictions by removing the non-local context, thereby focusing the model's attention on the local sentence's NE predictions. A_{22} shifts the focus to learning from 'helpful' non-local predictions, training the model to effectively utilize accurate non-local NE information. Finally, A_{23} advances to the complex task of matching and selecting relevant ones from both 'helpful' and 'unhelpful' non-local sentences for a comprehensive and refined error recovery.

Goal	Data Augmentation	Local NE predictions	Non-local sentences and their predictions	Generated Benchmarks
A_{21}	Remove All NLCs Change Local Type	[black widow]$_{\text{non-entity}}$ [black widow]$_{\text{group}}$	⟨`all_removed`⟩ ⟨`all_removed`⟩	[black widow]$_{\text{non-entity}}$ [black widow]$_{\text{group}}$
A_{22}	Sample One Helpful NLC Change Local Type	[black widow]$_{\text{non-entity}}$ [black widow]$_{\text{group}}$	(1) [black widow]$_{\text{person}}$ is a female character (1) [black widow]$_{\text{person}}$ is a female character	[black widow]$_{\text{person}}$ [black widow]$_{\text{person}}$
A_{23}	⟨`original_example`⟩	[black widow]$_{\text{non-entity}}$	(1) [black widow]$_{\text{person}}$ is a female character (2) [black widow]$_{\text{creative-work}}$ is a film (3) [black widow]$_{\text{person}}$ fights with hulk	[black widow]$_{\text{person}}$

ing framework, building upon the foundations laid by A_{21} and A_{22}. While A_{22} focuses on learning to replicate predictions from exclusively 'helpful' non-local sentences (by removing unhelpful ones), A_{23} requires the model to discern and select the 'helpful' sentences from a mix that includes both helpful and unhelpful non-local sentences. This step is crucial as it mirrors real-world scenarios where the model must navigate and evaluate a variety of non-local information to refine its predictions.

5 Experimental Studies

Datasets. In the field of NER, there are many different datasets which differ in domains, languages, the types of entity and the granularity of entity types. We will evaluate our proposed approaches on the WNUT17 dataset (Derczynski et al., 2017) [4]. WNUT17 is adopted because the SOTA system (Wang et al., 2021) still gives poor performances due to noisy texts, rare entities and the intentional data mismatch created by different creation time (see Appendix H). We would like to see how well our NER error recovery performs on this noisier dataset, especially focusing on the problem of rare entities. This dataset consists of 5,678 sentences in total, with a roughly 3:1:1 split for the training, development, and test set. For more detailed statistics of the dataset, please refer to Appendix H.

Baselines and Oracle Method. To validate the effectiveness of our model, we show baseline methods and our model in three categories: the ordinary NER ($\mathbf{M_O}$), method incorporating implicit ($\mathbf{M_I}$) and explicit redundancy ($\mathbf{M_E}$). The result of **no context** in the work of Wang et al. (2021) are included for readers' reference, corresponding to the vanilla model without any external context in $\mathbf{M_O}$.

Two other results in the paper, **CLNER** and **CLNER (+CL)** are also presented in $\mathbf{M_I}$, the latter of which is the best setting in their work by using cooperative learning and also the SOTA performance on WNUT17. Note that Wang et al. (2021) set up their experiment by adding the development set to the training set to improve the performance. However, in order to have a fair ground and a development set to tune hyperparameters in our experiment, we only use the training set. The experiment **no context** and **CLNER** in Wang et al. (2021) are also reproduced under this setting instead.

Evaluation Metrics. The micro-f1 score: $\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$ is used to evaluate the performance of NER. To further understand how our error recovery methods perform compared to the other methods, the error recovery ratio **ER** (%): $\frac{f_M - f_B}{1 - f_B}$, where f_M is the f-1 score of the method M, and f_B is the f1 score of the baseline method, i.e., **no context**.

Setup In the *Query Generator*'s mention detector, recall is prioritized over precision to generate a sufficient number of queries for retrieval. This involves adjusting the threshold for binary decisions based on the best f_2-score in the development set.

For training *Reference Sentence Selector*, the benchmark for each non-local sentence in each redundancy set is required. This is achieved by using the same clustering algorithm as in Majority-Voting Refiner (see Section 2.5). Subsequently, a benchmark of "keep" will be assigned if the sentence has any gold NE (i.e., the NE benchmarks of the target sentence) and "discard" otherwise. Note that since

[4] https://huggingface.co/datasets/wnut_17

Table 3: **Performance of the Development and the Test Set of WNUT17.** Three categories are the ordinary NER ($\mathbf{M_O}$), method incorporating implicit ($\mathbf{M_I}$) and explicit redundancy ($\mathbf{M_E}$). Three experiment results, **no context**, **CLNER**, **CLNER (+CL)**, from the paper of (Wang et al., 2021) are shown, which represents an ordinary NER, the model with external sentences, and the model with the best setting respectively. The micro f1 score (**F1**) and the error recovery rate (**ER**) to the original predictions (**no context**) is computed, respectively. **CLNER (w/ Our Sents)** refers to the method of (Wang et al., 2021) along with taking as input the selected and retrieved sentences from our system. **ReMV** and **ReMV (+span)** refers to the majority-voting based method without and with span correction function (cf. Algorithm 1). **ReTRF** refers to the transformer-based model. **Oracle** refers to the top-line performance obtained by using **ReMV (+span)** with only the "pseudo gold" reference sentences (i.e., annotated as "keep" automatically in our benchmark generation algorithm). **ReTRF (-$\mathbf{A_{21}}$)**, and **ReTRF (-$\mathbf{A_{21}}$-$\mathbf{A_{22}}$)** denotes the ablation experiment of removing the first and both stages of $\mathbf{A_{21}}$ and $\mathbf{A_{22}}$ from the proposed Incremental Learning technique, respectively. **ReMV (-RSS)** represents the ablation experiment of removing the module *RSS* from **ReMV** and thus keeping all sentences. An average performance of five experiments was reported.

Type	Method	Development Set			Test Set		
		F1 (%)	ER (%)	$\frac{\text{ER}}{\text{ER}_{\text{Oracle}}}$ (%)	F1 (%)	ER (%)	$\frac{\text{ER}}{\text{ER}_{\text{Oracle}}}$ (%)
M_O	**no context** by Wang et al. (2021)	N/A	—	—	57.86	—	—
	no context	69.09	0.00	0.00	59.62	0.00	0.00
M_I	**CLNER** by Wang et al. (2021)	N/A	—	—	60.20	1.44	4.87
	CLNER (+CL) by Wang et al. (2021)	N/A	—	—	60.45	2.06	6.97
	CLNER (w/ Our Sents)	69.90	2.62	6.40	60.96	3.32	11.26
M_E	**ReMV**	70.27	3.82	9.33	60.42	1.98	6.72
	ReMV (-RSS)	61.53	-24.46	-59.72	57.32	-5.7	-19.34
	ReMV (+span)	68.80	-0.94	-2.29	60.70	2.67	9.06
	ReTRF	**71.82**	**8.83**	**21.56**	**61.43**	**4.48**	**15.20**
	ReTRF (-A_{22})	71.35	7.31	17.85	60.99	3.39	11.50
	ReTRF (-A_{21}-A_{22})	70.50	4.56	11.13	60.04	1.04	3.53
	Oracle	81.75	40.96	100	71.52	29.47	100

the precision of this module is more significant than recall (see Section 2.3), we put more weights on the precision in the loss calculation and the model selection during training. Please see Appendix I for more details.

The N-fold cross validation in the training data generation for Refiner uses a canonical N of 5.

In order to prevent overfitting, early stopping is conducted for all training experiments of all modules in the system by selecting the model with the best micro-f1 score on the development set. See Appendix J for more training details.

5.1 Results

This section details the experimental outcomes presented in Table 3. The baseline model **no context** achieved f1 scores of 69.09 and 59.62 on the development and test sets, respectively. Conversely, the **CLNER (w/ Our Sents)** experiment, which inputs sentences selected by the first three modules into **CLNER**, showcases enhanced performance with an F1 score of 60.96 and an error recovery rate of 3.32%. This improvement underscores the superior quality of sentence selection by our system, attributed to the efficacy of the *Query Generator* and *Reference Sentence Selector*. For a detailed comparison between our retrieval system and that of Wang et al. (2021), refer to Appendix K.

In our study, we included the **Oracle** performance metrics in our comparative analysis. The experiment demonstrated that **Oracle** achieved scores of 81.75 and 71.52 on the development and test sets, respectively, with corresponding recovery ratios of 40.96% and 29.47%. This serves as a benchmark for the potential upper-bound performance in the refinement stage and indicates the proportion of recoverable errors. For context, **CLNER (w/ Our Sents)** managed to recover 11.26% of the errors deemed recoverable by the **Oracle** standard. The presence of unrecoverable errors is largely attributed to the occasional inadequacy of Google Search in providing complete knowledge of the NEs in target sentences. Moreover, the noise inherent in the redundancy set collected from the Internet and during the retrieval process also contributes to these limitations.

For M_E, the two majority-voting based algorithms **ReMV**, and **ReMV (+span)** achieves an error recovery rate of 3.82% and 1.98% respectively. On the other hand, **ReTRF** achieves a recovery rate of 8.83% and 4.48%, making 71.82 on the development set and a new state-of-the-art performance of 61.43 on the test set.

To delve deeper into the efficacy of various techniques and components, the outcomes of our ablation study are delineated below. In the experiment titled **ReMV (-RSS)**, which involves the omission of the selectivity function of the *Reference Sentence Selector* module from **ReMV** (achieved by retaining all sentences without exclusion), we observed a diminished recovery ratio of 24.46% and an error increment of 5.7% in the development and test sets, respectively. These findings underscore the pivotal role of the *Reference Sentence Selector* module in enhancing overall performance.

The following ablation study focusing on the *Incremental Learning* aspect of our model reveals notable findings. When the model is not pre-trained on sub-tasks, particularly in the case of the transformer-based model excluding A_{21} and A_{22} (termed as **ReTRF (-A_{21}-A_{22})**), there is a marked decrease in performance. Specifically, this configuration yields a recovery ratio of 4.56% and 1.04% in the development and test sets, respectively. This represents a reduction of 4.27% and 3.44% when compared to the fully equipped method, **ReTRF**.

6 Discussion

Challenges and Insights in Embedding Integration and Incremental Learning This research highlights a critical challenge in the integration of well-trained token embeddings and newly trained tag embeddings within our NER refinement model. Our experiments reveal a tendency for the model to overly depend on token embeddings, which are richly informed from extensive pre-training on language modeling tasks. This reliance becomes particularly pronounced when both embeddings are trained simultaneously in an end-to-end procedure without a phased approach.

Overcoming Embedding Bias through Incremental Learning We discovered that without Incremental Learning, the tag embeddings remain largely unadjusted during training, implying their underutilization. This suggests that the model, in such a scenario, functions more as an additional NE tagger rather than a true refinement model. The intended purpose of a refinement model is to selectively adjust and validate first-stage NE predictions, yet in the absence of effective tag embedding utilization, it reverts to relying solely on text for NER, undermining its refinement role.

Significance of Two-Stage Architecture and Explicit Redundancy The integration of a two-stage architecture, leveraging explicit redundancy, is pivotal in breaking down the complex task of NER error recovery into manageable sub-tasks. This approach contrasts starkly with models that utilize implicit redundancy, where the learning process is less structured and controllable. The explicit redundancy, as introduced in the earlier sections, allows for a step-by-step learning process, making training more efficient and targeted. Without this bifurcation, models relying on implicit redundancy tend to engage in self-directed learning, which lacks the precision and targeted efficiency of our proposed method.

Real-time Adaptation in NER Using Redundancy To ensure NER models remain current with evolving linguistic trends, our study, as elaborated in Appendix B.3, proposes integrating redundancy during the testing phase. This approach allows models to adapt to newly recognized Named Entities, overcoming limitations in initial training data. By retrieving real-time, relevant redundant information at the current time, the model dynamically updates its understanding and predictions. This strategy draws parallels to the Retrieval-Augmented Generation (RAG) used in Large Language Models (LLMs), where external data sources are utilized for immediate model adaptation. Such an approach not only enhances the model's generalization capability but also aligns with advanced methodologies in natural language processing, ensuring its applicability in diverse and evolving linguistic contexts.

7 Conclusions

In conclusion, this paper has presented a novel redundancy-enhanced framework for error correction in NER. By integrating internet-sourced related sentences into the NER process, we have demonstrated significant improvements in error correction accuracy. Our transformer-based refiner, methodologically anchored in Incremental Learning, effectively combines additional information with minimal effort, leading to a notable reduction in errors and an increase in micro-f1 scores over existing baselines. This approach not only addresses key challenges in NER, such as context dependency and data sparsity, but also marks a step forward in the utilization of redundancy for enhancing machine learning models. Our results on the WNUT17 dataset underline the potential of this framework in advancing the field of NER and set the stage for future research in this area.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019a. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, ACL '19, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019b. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrew Eliot Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University, USA.

Razvan Bunescu and Raymond Mooney. 2004. Collective information extraction with relational Markov networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 438–445, Barcelona, Spain.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy approach using global information. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 160–163.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL '20, pages 8440–8451, Online. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, WNUT '17, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1 (Long and Short Papers) of *NAACL-HLT '19*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexander Dunn, John Dagdelen, Nicholas Walker, Sanghoon Lee, Andrew S. Rosen, Gerbrand Ceder, Kristin Persson, and Anubhav Jain. 2022. Structured information extraction from complex scientific text with fine-tuned large language models.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan. Association for Computational Linguistics.

Tao Gui, Jiacheng Ye, Qi Zhang, Zhengyan Li, Zichu Fei, Yeyun Gong, and Xuanjing Huang. 2020. Uncertainty-aware label refinement for sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2316–2326, Online. Association for Computational Linguistics.

Felix Hamborg, Corinna Breitinger, and Bela Gipp. 2019. Giveme5w1h: A universal system for extracting main events from news articles. In *Proceedings of the 13th ACM Conference on Recommender Systems, 7th International Workshop on News Recommendation and Analytics*, INRA '19, Copenhagen, Denmark.

Richard W Hamming. 1950. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160.

Anwen Hu, Zhicheng Dou, and Ji-rong Wen. 2019. Document-level named entity recognition by incorporating global and neighbor features. In *Information Retrieval: 25th China Conference, CCIR 2019, Fuzhou, China, September 20–22, 2019, Proceedings 25*, pages 79–91. Springer.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Zhuolin Jiang, Amro El-Jaroudi, William Hartmann, Damianos Karakos, and Lingjun Zhao. 2020. Cross-lingual information retrieval with BERT. In *Proceedings of the workshop on Cross-Language Search and Summarization of Text and Speech (CLSSTS2020)*, pages 26–31, Marseille, France. European Language Resources Association.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, ICLR '14.

Vijay Krishnan and Christopher D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, COLING-ACL '06, pages 1121–1128, Sydney, Australia. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, California. Morgan Kaufmann Publishers Inc.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, and Myung-Gil Jang. 2006. Fine-grained named entity recognition using conditional random fields for question answering. In *Proceedings of the Third Asia conference on Information Retrieval Technology*, AIRS '06, pages 581–587, Singapore. Springer-Verlag.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7999–8009.

Mengyang Liu, Haozheng Luo, Leonard Thong, Yinghao Li, Chao Zhang, and Le Song. 2022. Sciannotate: A tool for integrating weak labeling sources for sequence labeling.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301–5307, Florence, Italy. Association for Computational Linguistics.

Zihan Liu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. 2021. Ner-bert: A pre-trained model for low-resource entity tagging.

Ying Luo, Fengshun Xiao, and Hai Zhao. 2019. Hierarchical contextualized representation for named entity recognition.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 1–8, Bergen, Norway. Association for Computational Linguistics.

Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. 2021. Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys (CSUR)*, 54(1):1–39.

Letian Peng, Zihan Wang, and Jingbo Shang. 2023. Less than one-shot: Named entity recognition via extremely weak supervision.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1104–1112, Beijing, China. Association for Computing Machinery.

Charles Sutton and Andrew McCallum. 2010. An introduction to conditional random fields.

Antonio Toral, Elisa Noguera, Fernando Llopis, and Rafael Muñoz. 2005. Improving question answering using named entity recognition. In *Natural Language Processing and Information Systems*, NLDB '05, pages 181–191, Berlin, Heidelberg. Springer Berlin Heidelberg.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. 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 of *ACL '21*, pages 1800–1812, Online. Association for Computational Linguistics.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2020. Named entity recognition with context-aware dictionary knowledge. In *Chinese Computational Linguistics: 19th China National Conference, CCL 2020, Hainan, China, October 30–November 1, 2020, Proceedings 19*, pages 129–143. Springer.

Roman Yangarber and Lauri Jokipii. 2005. Redundancy-based correction of automatically extracted facts. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT/EMNLP '05, pages 57–64, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*.

Mengdi Zhu, Zheye Deng, Wenhan Xiong, Mo Yu, Ming Zhang, and William Yang Wang. 2020. Neural correction model for open-domain named entity recognition.

Supplementary Material

A Contrasting Implicit and Explicit Redundancy Utilization Methods

As outlined in Section 1, there are two primary methods for incorporating redundancy in Named Entity Recognition (NER): (1) utilizing implicit redundancy (M_I) and (2) utilizing explicit redundancy (M_E). M_I involves retrieving implicit redundant information, such as from a Search Engine, and feeding it into an NE tagger along with the target sentence to identify NEs, as demonstrated by (Wang et al., 2021). In contrast, M_E employs explicit redundant information—sentences tagged with NEs—inputted into a recovery model for final decision-making (refer to Table 4 for a detailed comparison). This study focuses specifically on M_E.

M_E adopts a two-stage decision-making process. The first stage conducts NER on the local sentence and the retrieved redundant sentences. The second stage serves as an error recovery module, refining decisions based on the predictions from the first stage. This method distinctly separates two abilities: (1) NER capability (A_1) and (2) matching and selection capability (A_2). A_1 tends to rely on a memory-like mapping between text content and NE mentions, often leading to poor generalization. Conversely, our emphasis is on A_2, which involves non-memory-based activities like matching, selecting, and copying predictions based on the context of both local and non-local sentences. This approach could theoretically bolster the model's generalizability.

One might argue that M_I could implicitly encompass both A_1 and A_2 in a single-stage process. However, our analysis suggests that M_I primarily focuses on A_1, memorizing more NEs without effectively learning the dual abilities, as observed in (Wang et al., 2021)'s work (see Appendix E). Consequently, M_I tends to struggle with generalization, particularly when encountering unseen NEs. The separation of A_1 and A_2 in M_E not only enhances training efficiency but also ensures that the system possesses the crucial A_2 ability. Additionally, while A_1 is more task-specific and less transferable across different Information Extraction tasks, A_2 is theoretically more versatile and reusable. We propose two implementations of the A_2 error recovery module: a majority voting approach and a Transformer-based approach. For further details on these implementations, please refer to Section 3.2.

B The Role of Redundancy in Named Entities

In this work, we delve into the crucial role of redundancy in enhancing Named Entity Recognition (NER) effectiveness. Redundancy, as defined in Section 1, refers to the repeated occurrences of the same Named Entity (NE) across diverse sources, with its frequency and distribution having a significant impact on NER system performance.

To illustrate, consider an NE, X, which may be a new concept or entity introduced at a specific time, t_0, as shown in Figure 3. We can envision a universal set, $U(X, t_0)$, comprising all mentions of X globally at time t_0. Alongside, we have an NER training dataset, $TR(X, t_0)$, which is a subset of $U(X, t_0)$, containing mentions of X.

B.1 The Origin of Redundancy in NEs

Redundancy in NEs is largely influenced by the entity's global recognition. For instance, an NE, X, which might be a newly emerged concept at time t_0, would be less known and hence mentioned less frequently, such as in micro-blogs like Twitter. This results in limited redundancy for X. Conversely, widely recognized NEs are mentioned more frequently across various platforms, leading to higher cross-document redundancy.

B.2 Redundancy in Training Improves the Performance

There are many advantages if a high level of redundancy exists in the training data. The different occurrences of X in $TR(X, t_0)$ allows the NER model to know what contexts X appears in the training procedure. And both implicit and explicit types of redundancy provide the trained model the ability to recognize NEs.

Implicit Redundancy in Pre-Training for Language Modeling. Implicit redundancy plays a critical role in the development of language models, particularly in the context of Named Entity Recognition (NER). This type of redundancy is characterized by the absence of explicit tags that identify the span and type of Named Entities (NEs) in unannotated text corpora. Despite the lack of explicit annotations, implicit redundancy is prevalent in these vast corpora, offering an indirect yet valuable source of information about NEs.

During the training phase of most language models, this implicit redundancy is inherently incorporated as part of the unsupervised language modeling task. By processing large volumes of text that contain repeated, untagged references to various NEs, language models gain a deeper understanding of the essence and semantic context of these entities. This exposure enables the models to develop an intuitive recognition of NEs, enhancing their ability to identify and interpret NEs in a wide range of contexts.

In essence, implicit redundancy contributes significantly to the foundational knowledge of language models, particularly in understanding and identifying NEs. It serves as an indirect form of learning, where the frequency and context of NE mentions in large text corpora provide the models with a nuanced understanding of these entities without the need for explicit tagging.

Explicit Redundancy in Fine-Tuning for NER. Explicit redundancy plays a distinct and crucial role during the fine-tuning stage of language models for Named Entity Recognition (NER) tasks. Unlike implicit redundancy, explicit redundancy is characterized by clear indicators or tags that specify the types and spans of Named Entities (NEs) within a corpus. This form of redundancy has been conventionally employed in the fine-tuning process of NER models, often without explicit recognition of its connection to the concept of redundancy.

During fine-tuning, explicit redundancy is leveraged through the incorporation of annotated data, where NEs are clearly labeled. This process involves loss propagation, a technique that refines the model's understanding of NEs by exposing it to various contexts in which the same NE appears. Each occurrence of an NE in the training corpus, despite referring to the same entity, is surrounded by different textual contexts. These varied contexts are crucial as they enable the model to learn not just the identification of NEs but also their possible semantic roles and relationships in different situations.

Error Tolerance in Training Data. The volume of redundancy present in training data significantly influences two key aspects in Named Entity Recognition (NER) models: performance enhancement and error tolerance.

Firstly, the level of redundancy—whether implicit within the language modeling task or explicit during the fine-tuning stage for NER—directly impacts the model's predictive accuracy for a specific Named Entity (NE), X. A higher redundancy level means that X appears in various contexts within the training data. This repeated exposure enables the model to gain a comprehensive understanding of the NE, both semantically and syntactically. It learns not only to identify X but also understands the diverse contexts and situations in which X can be used. This deepened understanding inherently improves the model's performance in accurately recognizing and classifying NEs.

Secondly, a high level of redundancy in the training data provides a mechanism for error tolerance, particularly valuable in scenarios involving data imperfections. Such imperfections might include typos, case errors, formatting issues, or misclassifications (e.g., mistaking a personal NE for a corporate one in Twitter comments) in the unlabeled or the labeled corpus used for training. In these instances, the presence of a high volume of correct usages and occurrences of NEs in different sentences acts as a buffer. It allows the model to discern the correct interpretation or classification of an NE despite the presence of errors. This error tolerance mechanism ensures that the model's learning is not significantly derailed by a few inaccuracies in the data, thereby maintaining the integrity and reliability of the NER process.

B.3 Redundancy in Testing Enhances Generalization Ability

Incorporating redundancy during the testing phase is as crucial as in training, significantly impacting the model's generalization ability. For instance, consider a Named Entity (NE) X that was newly introduced at a previous time t_0 (e.g., a newly released movie), as depicted in Figure 3. At t_0, the universal set $U(X, t_0)$ encompasses all global mentions of X, and the NER model is trained on a subset $TR(X, t_0)$ from $U(X, t_0)$. Initially, due to X's novelty, both $|TR(X, t_0)|$ and $|U(X, t_0)|$ are small, leading to unreliable NER predictions for X due to limited implicit and explicit redundancy.

Over time, as X becomes more recognized ($|U(X, t)| > |U(X, t_0)|$), its mention in the testing phase is likely to increase ($P(x = X)$), highlighting the model's initial performance issues. To address this, one could consider retraining the model with a new dataset $TR(X, t)$ sampled from $U(X, t)$ at a later time t. However, this approach is labor-intensive and lacks scalability and generalization, as it's impractical to constantly update the training dataset with every new NE introduction.

Instead, our work proposes incorporating redundancy directly during the testing phase. We retrieve the relevant, redundant information $R(X, t)$ from $U(X, t)$ using a search engine at time t. This approach enables the model to adapt to new mentions

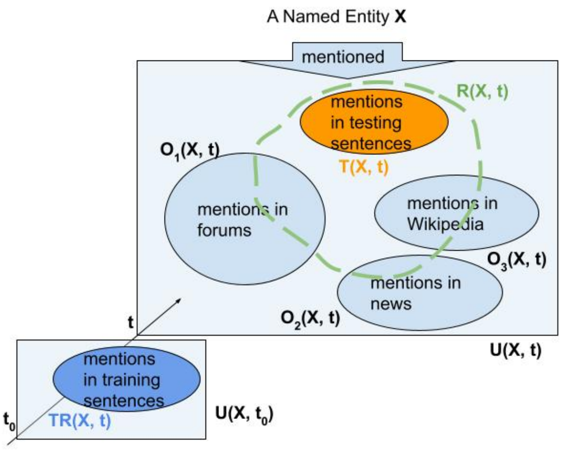

Figure 3: **How unseen NEs naturally raise in the real world.** $T(X, t)$, $O(X, t)$, $R(X, t)$ is a set of mentions of X in the given testing document, some external documents, and retrieved related documents at a certain time t. Each is a subset of $U(X, t)$, a universal set of all mentions of NE X in the world at a certain time t. $TR(X, t_0)$ is a set of mentions of X in the training data of the NER model at a previous time t_0.

of X without the need for constant retraining or human annotations. By leveraging current, real-time redundant data, the model's predictions for X become more accurate and generalized, effectively keeping pace with the evolving linguistic landscape.

Note this approach of incorporating redundancy during the testing phase in NER shares similarities with the Retrieval-Augmented Generation (RAG) technique used in Large Language Models (LLMs). Just as RAG leverages external knowledge sources to enrich the response generation process in LLMs, our method utilizes up-to-date information retrieved from external sources to enhance the accuracy and generalization of NER predictions.

C Related Work of NER

Named Entity Recognition (NER) is a critical task in Natural Language Processing (NLP) and Information Extraction (IE), aimed at identifying and classifying named entities within text into predefined categories such as persons, organizations, locations, time expressions, quantities, and monetary values. It is one of the fundamental steps in both Natural Language Processing (NLP) and Information Extraction (IE), and serves as a stepping stone for many downstream tasks such as event extraction (Ritter et al., 2012; Hamborg et al., 2019), and question answering (Toral et al., 2005; Lee et al., 2006), etc. In the past decade (Nasar et al., 2021; Li et al., 2020; Yu et al., 2020), NER has seen a tremendous amount of advances due to the rapid development of deep learning and contextual word embeddings (e.g., ELMo, BERT, RoBERTa)(Devlin et al., 2019; Conneau et al., 2020). Recently, studies show that they still suffer in information-scarce tasks, such as no reliable hints (for identifying associated NEs) are provided in the given short sentence, and many uncommon language uses exist such as abbreviations and rare/unseen NEs (Derczynski et al., 2017), where the state-of-the-art performance ((Wang et al., 2021)) still struggles at a f1-score of 60.45. One of the possible solutions is to find the reliable hints in some related reference sentences if they can be additionally provided. In other words, for each local sentence, if we could incorporate information other than the local context, e.g., from a document collection, or from the Internet, we could form the "redundant" information from external context, which then helps to recover errors in the local sentence (c.f. Appendix D for more details). This is the focus of this research.

D Related Work of IE with Redundant Information

In the world filled with tremendous amount of data, redundancy or duplication of information exists and sometimes is intentionally formed across various fields. It is common to utilize redundant information to be resilient to errors in areas such as database systems, software architecture, communication protocols, random access memory, chromosome genes in biology. However, the past decades see few studies of using redundancy in the sphere of Machine Learning or Deep Learning with only some exceptions (Hamming, 1950; Yangarber and Jokipii, 2005; Krishnan and Manning, 2006). Below we list all the related work to the best of our knowledge. We focus on the utilization of redundancy information in Natural Language Processing (NLP) based on Deep Neural Network (DNN).

The concept of utilizing "redundancy" to do error correction/detection is not new. Redundancy, also known as "duplicate information', has been explicitly/widely utilized in various computer systems (e.g., on-line memory correction (Hamming, 1950), database entry correction (Yangarber and Jokipii, 2005)) to enhance the reliability of a system. Even in NER, the NE redundancy within given documents has been explicitly utilized to raise the performance. For example, Krishnan and Manning (2006) utilize NE redundancy to model label consistency across given documents.

However, in some applications running in the wild such as Twitter, the related documents are not available (Derczynski et al., 2017). Moreover, due to the time difference between the model training time and the real inference time, the problem of unseen NEs naturally arises in the real world (Derczynski et al., 2017) (which is more distinct than other tasks that have relatively same concepts over time such as parsing). Therefore, in those applications, the unseen NE redundancy has not been explored and explicitly utilized in the literature with only one exception from the work of Wang et al. (2021). Specifically, Wang et al. (2021) query a Search Engine with the sentence text to retrieve related sentences to collect redundant information.

Recent studies in NER (Wu et al., 2020; Liu et al., 2022; Peng et al., 2023) and other Information Extraction (IE) fields (Dunn et al., 2022; Lin et al., 2020; Jiang et al., 2020) are incorporating redundant information through statistical models. These methodologies broadly fall into two distinct categories: the One-Stage (Liu et al., 2019; Luo et al., 2019) and Two-Stage approaches (Borthwick, 1999; Gui et al., 2020). The One-Stage approach employs a singular model that directly executes NER by simultaneously considering all pertinent text passages, as shown in Table 5. In contrast, the Two-Stage approach is more sequential. It initially performs NER across various documents, which constitutes the first stage. Subsequently, in the second stage, it refines these NER results by leveraging cross-sentence or cross-document redundant information. We systematically categorize these related works across different dimensions, as delineated in Table 6.

D.1 One-Stage Approach

The One-Stage Approach in NER employs a unified model to simultaneously process multiple sentences for entity identification. A quintessential example is *BERT-NER* (Liu et al., 2021), which concatenates as many consecutive sentences as its token limit (512 sub-tokens) allows. This methodology facilitates inter-sentence information sharing, enhancing entity recognition. However, challenges arise in managing longer dependencies. Notable efforts to address these include:

Hand-Crafted Features Chieu and Ng (2002, 2003) developed global features, like a token's majority label, to capture dependencies across sentences. These features were integrated into a Conditional Random Field (CRF)-based (Sutton and McCallum, 2010) NE sequence tagger.

Table 4: Comparison between the methods of incorporating implicit redundancy (M_I) and explicit redundancy (M_E)

Approach	**Implicit Redundancy (M_I)**	**Explicit Redundancy (M_E)**
Process Flow	Single stage	Dual stages
Component(s)	Named Entity (NE) Tagger	[$Component_1$] Named Entity (NE) Tagger [$Component_2$] Recovery Module
Inputs to Final Component	Local sentence and redundant sentences retrieved	Similar to M_I, with the addition of NE predictions from the first stage
Capabilities	Utilizes both local and non-local information to identify NEs, exhibiting memory-like behavior (A_1)	[$Component_1$] Similar to M_I (A_1) [$Component_2$] Exhibits non-memory like behavior, making refined decisions through a voting-like process (A_2)
Final Component	Task-specific	Potentially task-agnostic

Non-Local Feature Extractors Innovations like those by Hu et al. (2019) involve automatic extraction of features such as document-level token representations. They achieve this by merging embeddings of identical tokens from different sentences or documents. Similarly, Liu et al. (2019) employed this technology for sentence representation, combining it with individual token representations.

Skip Connections Techniques implemented by researchers like Bunescu and Mooney (2004), Sutton and McCallum (2004), and Finkel et al. (2005) involve creating direct links between model nodes or neurons representing the same token in various contexts. These skip connections directly model non-local, non-sequential dependencies, effectively using a virtual memory segment to recall and utilize prior representations or embeddings.

Virtual Memory Usage Examples include Akbik et al. (2019b) and Luo et al. (2019), who utilized virtual memory for querying or updating the hidden states of repeated tokens across different sentences or documents.

D.2 Two-Stage Method

The Two-Stage Method, distinct from the One-Stage approach, emphasizes computational efficiency in NER. While the One-Stage approach, as shown in Table 1, conducts NER within a single document, the Two-Stage method leverages information across multiple documents, which is more resource-efficient. This method was pioneered in response to the computational challenges noted by Finkel et al. (2005) in conducting within-document NER. The Two-Stage approach operates sequentially: the first stage performs initial NER, and the second stage refines these results using broader, non-local dependencies. Our research focuses primarily on this second stage of refinement.

Two-Stage methods can be further categorized along four dimensions:

D.2.1 Scope of Information Used in Recovery Model

Specified Document-Collection Utilizes cross-document information but confines to a pre-specified collection (e.g., Mikheev et al. (1999); Krishnan and Manning (2006); Yangarber and Jokipii (2005)).

Unspecified Collections Extends beyond the given dataset, incorporating external information sources (e.g., Ji and Grishman (2008)).

D.2.2 Type of Information Used in Recovery Model

Label Consistency Exploits the consistency in labels of the same token sequence across occurrences (e.g., Krishnan and Manning (2006); Yangarber and Jokipii (2005); Ji and Grishman (2008); Gui et al. (2020)).

Other Information Sources Includes additional data like coreference resolution (e.g., Borthwick (1999)) or a correction dataset (e.g., Zhu et al. (2020)).

D.2.3 Recovery Model Used

The models range from rule-based and statistical-based to neural-based. While earlier approaches (e.g.,Yangarber and Jokipii (2005); Ji and Grishman (2008)) relied on hand-crafted rules, recent trends, like Zhu et al. (2020), favor neural-based models for their ability to learn error patterns without extensive feature engineering.

Table 5: Overview of One-Stage Approaches in Named Entity Recognition in chronological order: This table compares various works based on their scope of information used, type of non-local information utilized, method type, model description, and the specific NER task addressed. It highlights the evolution from within-document redundancy handling through hand-crafted features, skip connections, and feature extractors to more advanced techniques like virtual memory and representation consistency in both within-document and cross-document contexts.

Work	Scope of Information Used	Non-local Information Type Used	Method Type	Model Description	Task
Chieu and Ng (2002, 2003)	within-document redundancy	label consistency	hand-crafted features	CRF with within-document features	NER
Bunescu and Mooney (2004)	within-document redundancy	label consistency	skip connection	Relational Markov Network (Global Clique Templates that use Repeat Template (potential) to connect the label nodes of multiple entities.)	NER
Lafferty et al. (2001)	within-document redundancy	label consistency	skip connection	skip-chain CRF	NER
Finkel et al. (2005)	within-document redundancy	label consistency	skip connection	skip-chain CRF with penalties for label inconsistency	NER
Akbik et al. (2019b)	cross-document redundancy	representation consistency	virtual memory	BiLSTM with token representation memory	NER
Hu et al. (2019)	within-document redundancy	representation consistency	feature extractor	BiLSTM with non-local feature extractor (that obtains document-level token representation for multiple occurrences of the same token)	NER
Liu et al. (2019)	within-document redundancy	context information	features extractor	BiLSTM with non-local feature extractor (that obtains sentence representation)	NER
Luo et al. (2019)	cross-document redundancy	context information + representation consistency	features extractor + virtual memory	BiLSTM with token representation memory + non-local feature extractor (that obtains sentence representation)	NER

D.2.4 Processing Unit of Recovery Model

Cluster-Based Processes a cluster of mentions at a time, assigning uniform labels within the cluster (e.g., Yangarber and Jokipii (2005)).

Sentence-Based Re-tags each token in a sentence individually (e.g., Zhu et al. (2020)).

In our research, we adopt a neural-based model for its adaptability and potential portability across different NE recognizers, focusing on the refinement stage to investigate the effectiveness of integrating non-local information.

E An Example of Wang et al. (2021)'s Lack of Focus On the Reference Sentences

As mentioned in Section 1, our work relies on the explicit redundancy, while the work of Wang et al. (2021) is based on the implicit redundancy. Our claim is that using the explicit redundancy can make the model more aware of the information from the reference sentences, while using the implicit redundancy tends to ignore them and get stuck on the given sentence.

To illustrate, below shows an example, where the sentence to be tagged is "Venom is not good', the special token ⟨SEP⟩ means the end of the local sentence, and the tokens behind ⟨SEP⟩ are the tokens of all reference sentences, and $[span]_{type}$ is used to denote the NE span and the NE type.

Our experiment showed that Wang et al. (2021) identified "Venom" as a person in this sentence without any reference sentences:

(Wang et al., 2021)'s prediction: $[\text{Venom}]_{\text{person}}$ is not good.

This seems to be correct, since it could mean the fictional character "Venom" in Marvel Comics, hence a "person" NE. However, this sentence is actually ambiguous and its annotation should depend on the context and the time when it was used. For example, it was highly likely to be a "person" NE if it was written prior to the release of the film "Venom" by Marvel Studio. But if it was posted by a user on his Twitter page just one hour after the movie "Venom (2018)" had its debut, it is more likely that "Venom" should be identified as a "creative-work" in WNUT17 regime instead.

That is, we are interested if a model can make different predictions based on the retrieved reference sentences. Therefore, we prompted the model from Wang et al. (2021) with the some pseudo

Table 6: Comparative Analysis of Two-Stage Approaches in Named Entity Recognition: This table methodically outlines different two-stage NER methodologies, detailing the scope of information used, types of information leveraged in the second stage, models used in both stages, and the processing unit of the second stage. It showcases a range of techniques from statistical models with hand-crafted features to advanced neural network applications, highlighting the evolution and diversity in tackling NER tasks.

Work	Scope of Information Used	Information Type Used By 2nd Stage	1st Stage (Extraction)	2nd Stage Model Type (Refinement)	2nd Stage Model	2nd Stage Processing Unit	Task
Borthwick (1999)	cross-document redundancy	label consistency	Maximum Entropy	Statistical (Maximum Entropy)	hand-crafted features	sentence-based	NER
Borthwick (1999)	cross-document redundancy	label consistency	Maximum Entropy	Statistical (Maximum Entropy)	hand-crafted features	sentence-based	NER
Krishnan and Manning (2006)	cross-document redundancy	label consistency	CRF	Statistical (CRF)	hand-crafted features	sentence-based	NER
Yangarber and Jokipii (2005)	cross-document redundancy	label consistency	HMM-based IE	Rules	hand-crafted features	cluster-based	Event Extraction
Ji and Grishman (2008)	cross-document redundancy + cross-corpus redundancy (external unlabeled corpus)	label consistency	HMM-based IE	Rules	hand-crafted features	cluster-based	Event Extraction
Zhu et al. (2020)	*	*others (recovery patterns learnt in the error correction dataset manually annotated)	*	Neural network (BERT)	hand-crafted features	sentence-based	Refine NER dataset
Gui et al. (2020)	within-document redundancy	representation consistency + label consistency	BiLSTM	Neural network (Transformer)	feature extractor + entity memory	sentence-based	NER
Ours	**external redundancy**	learned by 2nd-stage neural-based model	Transformer Encoder	Transformer Encoder	no features needed	sentence-based	NER

reference sentences stating that the definition of "Venom", e.g., "Venom is a film. ". Ideally, the model should be able to adjust its prediction based on the reference sentences as in

Benchmark: [Venom]$_{\text{creative-work}}$ is not good. ⟨SEP⟩ Venom is a film.

However, our experiment showed that Wang et al. (2021) was still stubborn on its original prediction as shown in Table 7. This showed that the work of Wang et al. (2021) did not pay enough attention to the references sentences. Hence, the focus of our work is to perform error recovery with the redundant information in a more explicit way than the implicit methods (cf. Appendix A).

F Pseudo-code of Majority-Voting Refiner and Oracle Refiner

Algorithm 1 shows the pseudo-code of the Majority-Voting Refiner. It mainly performs clustering, string-matching and majority voting as described in Section 3.2. This rule-based refiner follows the steps below for each sentence s and its NS': (i) collect all NEs in s and NS', (ii) cluster all NEs by grouping them if they share the same content words (normalized by lowercasing), and then record their NE predictions. This forms multiple NE clusters, (iii) in order to address false negative errors and NE span errors such as missing one token, we identify the spans string-matched by the content words of each NE cluster within the local sentence and its non-local sentences. Then we add those spans and their predictions to their corresponding NE cluster. Note that the cluster with the longest string matching is favored if multiple clusters can match such span. (iv) then the majority NE predictions are voted for each cluster and assigned to each NEs in the local sentence within that cluster.

G Generation of NER Recovery Dataset proposed by Krishnan and Manning (2006)

This section shows the generation process of NER recovery dataset proposed by Krishnan and Manning (2006) as shown in Figure 4. This dataset should have the NER tags predicted by the 1st-stage NE tagger along with the ground truth tags, so the error recovery model can learn what to and how to recover possible NER errors with the information of the original first-stage predictions. Therefore, we need to get predictions on the train data, the development data, and the test data. For predictions on the testing data (i.e., the development set or the test set), all the train data is used to train the 1st –stage NE tagger. For predictions on the train set, on the other hand, "N-fold cross-validation" is applied so that the predictions would be reflective of the performance on the test data. Note that their work did not have external sentences, while our work did have such sentences from a Search Engine. That is to say, we will also need to obtain the

Table 7: The Inputs and Outputs of One Example in Different Settings

model	input to the model	annotation on **Venom**
Benchmark	**Venom** is not good. ⟨SEP⟩ Venom is a film.	Creative Work
Wang et al. (2021)	**Venom** is not good.	Person
Wang et al. (2021)	**Venom** is not good. ⟨SEP⟩ Venom is a film.	Person

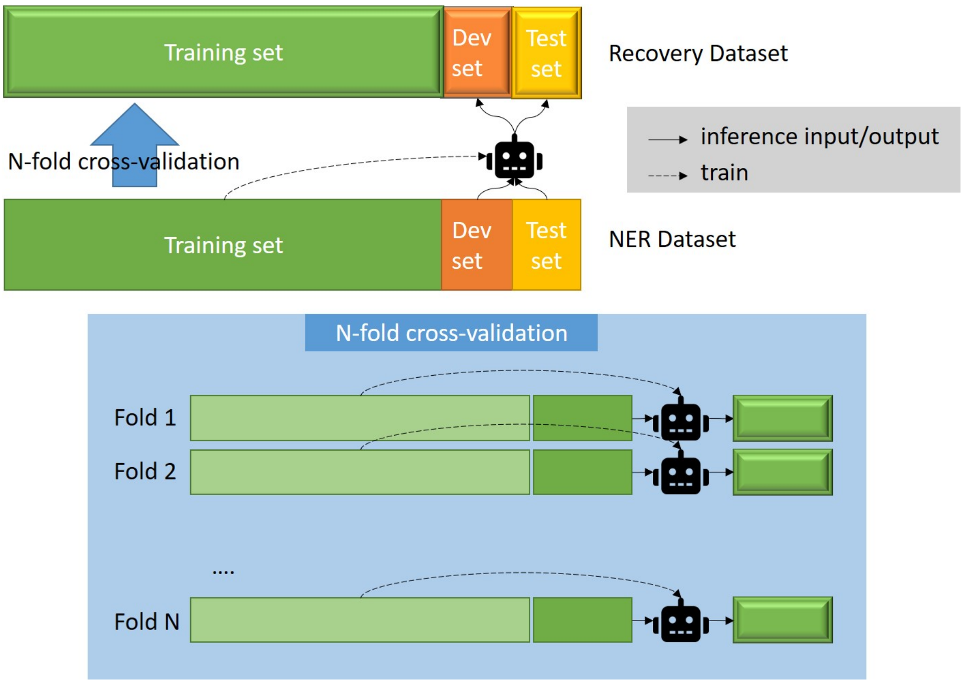

Figure 4: Generation of Recovery Dataset

first-stage predictions for the non-local reference sentences in addition to the local sentence.

H Statistics of WNUT17 Dataset

WNUT17 is the dataset adopted in the shared task of the 3rd Workshop on Noisy User-generated Text (Derczynski et al., 2017). It focuses on identifying unusual, previously-unseen entities in the context of emerging discussions for the purpose of evaluating the system's ability of generalizing on unseen data. This task provides an NER dataset of emerging and of rare entities from a variety of domains such as politics, news, and sports, etc., as shown in Table 8 This task concentrates on 6 types of named entities: Person, Location, Corporation, Group, Product and Creative-Work. These rare and unseen entities were created by collecting training data (no later than 2015) and testing data (Jan-May 2017) at different times and from different sources as shown in Table 9. Additionally, Derczynski et al. (2017) also ensured none of the entities between the training data and the testing data share the same surface form (by simply removing seen entities). Therefore, this is why it is by far the hardest NER dataset, where the SOTA model only has an average f_1 score of about 60.45 only (Wang et al., 2021).

I Implementation Details for the Other Modules

I.1 Query Generator

To extract the query spans from the sentence, we follow Huang et al. (2015) to specifically train a mention detector by the state-of-the-art sequence tagging framework (Wang et al., 2021) on the dataset of WNUT17. This comprises of: (1) one layer of embedding layer, which concatenates both the TWITTER embedding (Akbik et al., 2019a) and the XLM-RoBERTa embedding (Conneau et al., 2020), followed by (2) a linear re-projection layer to reduce the dimension, and (3) one final classification layer with CRF (Lafferty et al., 2001).

Table 8: Statistics of the adopted NER datasets, WNUT17

# Entities	3,850
# Tokens	101,858
# Sentences	5,689
# Documents	(no document split)
Tag Schemes	person, location, corporation, group, product, creative-work
Domains	Noisy texts from the forums or the micro-blogs of politics, sports, news, movies, science, countries and cities of Anglosphere (both high- and low-traffic)
SOTA f_1	60.45 (Wang et al., 2021)

Table 9: Statistics of the discrepancies in the three splits of WNUT17.

	train	**dev**	**test**
# Sentences	3,394	1,008	1,287
Source	Twitter	YouTube comments	Twitter, Reddit, StackExchange comments
Created Time	No later than 2015	Jan-May 2017	

I.2 Reference Sentence Selector

This model performs a classification task based on a pair of input, one local sentence and one non-local sentence, and the output is a binary class, "keep" or "discard". We use a conventional Transformer-based framework to perform such task, which consists of one Transformer encoder layer as the contextual word embeddings, one feed forward layer to map to a two-dimensional class space, and one softmax layer to obtain the probability. We form the input to the Transformer encoder by concatenating the local and the non-local sentence together with a special token ⟨SEP⟩ to separate them. Specifically, we choose ALBERT (Lan et al., 2020) as the Transformer encoder in this module for its known effectiveness on pair-wise classification.

I.3 Reference Sentence Selector

To train a model for *Reference Sentence Selector*, we need a dataset for pairwise sentence classification. Although it is true that paraphrase corpus could be used for pairwise classification, it is not useful for our module. This is because a paraphrased sentence in such corpus, such as replacing some words with their synonyms, is not preferred in our module, while a sentence which provides useful information, like the information that signals the NE type of the suspicious entity, is favored. Since there is no existing dataset suitable for such task, which needs to consider both "relatedness" and "usefulness" (in terms of aiding the NER task) for one sentence to another, we create a dataset for such purpose by our own in an automatic way.

Generally, by making use of the existing benchmarks from a NER dataset and using an existing NE tagging tool to tag NEs in nonlocal sentences, we can generate a benchmark (keep or discard) for each nonlocal sentence. To elaborate, if a nonlocal sentence has NEs that share the same string and the same NE type of the gold local NEs, it is said to be "keep" and "discard" otherwise.

Specifically, five steps are done (Figure 5): for a specific example, one local sentence (blue) with multiple corresponding non-local sentences retrieved from Google (orange)

1. we first cluster all NEs (with all different NE types) in both the local sentence (orange solid rectangle) and all the nonlocal sentences (blue solid rectangle), where a cluster (black circle) contains all NEs that share the same string in the lowercase form.

2. To also include "non-entity decisions" for the clustered NEs, we perform string-matching (after lowercasing) on all the sentences to identify those "non-entities", as indicated as a hollow box in the figure.

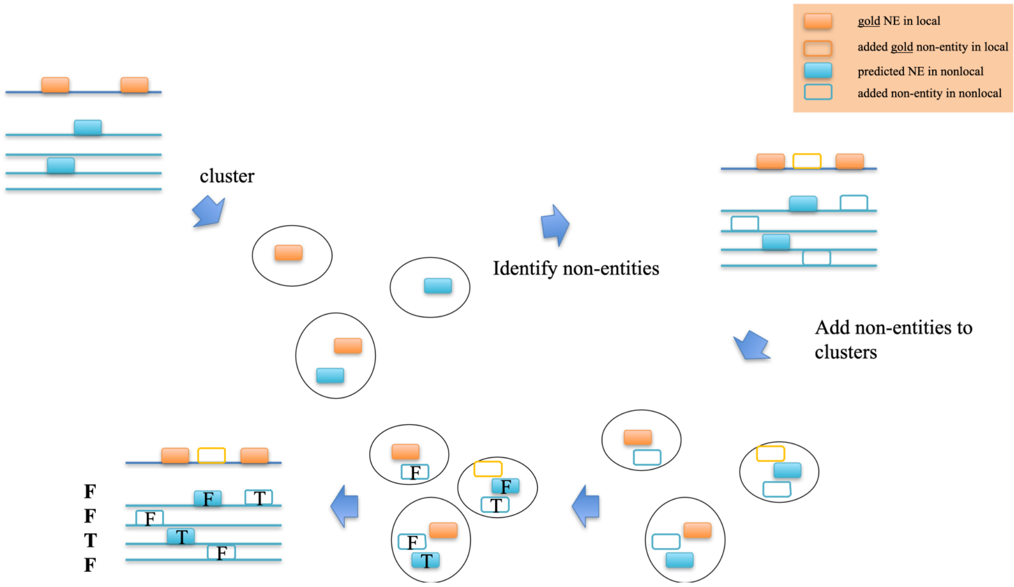

Figure 5: The Benchmark Generation for Each Non-local sentence for Reference Selection Selector

3. We then add those non-entities to each corresponding cluster

4. The clusters without any local NEs contained would be removed

5. For each cluster, if the nonlocal NE has the same type of the local NE, a benchmark of "keep" will be assigned to the sentence that the nonlocal NE belong; if not, "discard" will be assigned otherwise.

6. If all of the non-local NEs in one non-local sentence are annotated as "keep", the sentence will be annotated as "keep"; otherwise, the sentence will be annotated as "discard".

J Experimental Details

This section lists the details for reproducing our results. To have a fair comparison among various approaches, we adopt exactly the same hyperparameters as those used in (Wang et al., 2021): all word-embedding-vectors are tunable for fine-tuning; the word dropout rate is 0.1; A negative-log-likelihood loss is used after the last CRF layer; Adam optimizer (Kingma and Ba, 2015) is used with an epsilon of 10^{-6}; beta1 is 0.9; beta2 is 0.999; learning rate is 5×10^{-6} for all parameters in the model except that for CRF (in which it is set to 0.05); the mini-batch size is 2, and the batch accumulation is used with a step size of 2, giving an effective batch size of 4. A scheduler that linearly decays the learning rate is used, and the adopted model achieves the best micro-f1 score on the development.

K Differences between Our System and Wang et al. (2021)

CLNER (w/ Our Sents) in our experiment mostly follows the setting adopted by Wang et al. (2021) except the following points. First, we have more queries (through Query Generator) rather than the only one sentence query. Second, comparing *Reference Sentences Retrieval* with the reported setting of the Google Search Engine in their paper, we consider the title and the snippet in retrieved passages as correlated, and concatenate them to form one long sentence for re-ranking; in contrast, they view them as different sentences to be scored by the re-ranker. This allows us to rank/select different search results also based on their title information. Third, we enlarge the number of results retrieved from the default number of 12 to 100. Fourth, we train a specific selector, *Reference Sentence Selector*, rather than BERTScore used in their paper.

L A Correctly Recovered Example

This section shows an example of correctly recovered local sentence, "Venom is not good! ", with "Venom" as a creative-work NE. As shown in Figure 6, an ordinary NE recognizer fails to identify it as a creative-work NE and instead a person. This is likely to be caused by memorizing "Venom" as a person during pre-training language modeling over a large corpus with "Venom" as a common name of fictional characters.

Algorithm 1 Majority-Voting Refiner

Input: local sentence s;
Input: selected nonlocal sentences $\text{NS}' \leftarrow \{ns_1, ns_2, \ldots\}$;
CS $\leftarrow$ all NEs in s;
CNS$'$ $\leftarrow$ all NEs in NS$'$';
Instantiate a table clusters_table;
for each unique entity E in $Union(\text{CS}, \text{CNS}')$ **do**
  $key \leftarrow \text{lowercase}(E)$;
  if key is not present in the keys of clusters_table **then**
    store a new list in key of clusters_table;
    add E to the list stored in key;
  end if
end for
for each key in the clusters_table **do**
  $spans \leftarrow$ all spans string-matched by key in s
  for each $span$ in $spans$ **do**
    add $span$ to the list stored in key of clusters_table;
  end for
end for
for each key in clusters_table **do**
  Instantiate a table count_table;
  NE_list $\leftarrow$ clusters_table$[key]$;
  for each entity E in NE_list **do**
    $key_2 \leftarrow \text{get\_predictions}(E)$;
    if key_2 is not present in count_table **then**
      count_table$[key_2] \leftarrow 0$;
    end if
    increment count_table$[key_2]$ by 1;
  end for
  majority $\leftarrow$ get_majority_prediction(count_table);
  for each entity E in NE_list **do**
    if E belongs to the local sentence s **then**
      the prediction of $E \leftarrow$ majority;
    end if
  end for
end for

This problem still appears in the method of incorporating implicit redundancy ($\mathbf{M_I}$). For example, the method proposed by (Wang et al., 2021) (we train one model from the code provided) did not make a change on the NE prediction even when the retrieved sentences indicating that "Venom" is a film are provided. This shows that $\mathbf{M_I}$ still fails to fully utilize the information from the non-local sentences. The reason is likely to be the memory-like behavior aforementioned.

However, in the method of $\mathbf{M_E}$, both the majority-based approach and the DNN-based model successfully corrected the error by recognizing "Venom" as a creative work.

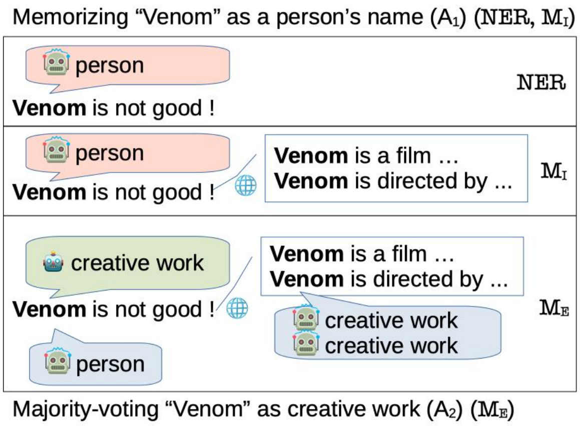

Figure 6: **A Correctly Recovered Example.** A local sentence "Venom is not good" is to be identified NEs along with the non-local sentences retrieved from the Internet. NER is an ordinary NE recognizer, while $\mathbf{M_I}$ and $\mathbf{M_E}$ is the method incorporating implicit and explicit redundancy respectively. Both NER and $\mathbf{M_I}$ made wrong predictions, while $\mathbf{M_E}$ (the model **ReTRF**) correctly identify "Venom" as a creative-work, where the second module made a majority-voting like behavior from the first module predictions (i.e., one person and two creative works).

M Limitations

This study, while contributing significant advancements in Named Entity Recognition (NER) error correction, has certain limitations that should be acknowledged. Primarily, the experimental validation of our proposed redundancy-enhanced framework and Incremental Learning approach was conducted on a limited number of datasets. This constraint may affect the generalizability of our findings across diverse linguistic contexts and text genres. Different datasets, especially those with varying linguistic structures or from distinct domains, could present unique challenges not encountered in our current experimental setup.

Additionally, our methodology was tested using only one model architecture. While this model demonstrated effective performance in our experiments, reliance on a single model may limit insights into how our approach would perform with alternative architectures, especially those with differing underlying principles or capabilities. Future research could explore the applicability and effectiveness of our framework with a variety of model architectures to establish broader validity and to understand the nuances of its performance across different NER models.

In summary, while our study presents a promising approach to NER error correction, further research involving a wider range of datasets and model architectures is necessary to fully ascertain the versatility and robustness of our methodology.

N Licenses and Intended Use of Resources

In this study, we utilized several external resources, each governed by its own license agreement. Understanding and adhering to these licenses is crucial for responsible and legal use of these resources.

WNUT17 Dataset (Derczynski et al., 2017): Licensed under the Creative Commons Attribution 4.0 International License (CC-BY-4.0), this dataset is open for use in academic and research contexts. The CC-BY-4.0 license allows for sharing and adapting the material, provided appropriate credit is given and any changes are indicated.

XLM-RoBERTa Pretrained Embeddings (Conneau et al., 2020) and CLNER (Wang et al., 2021): Both of these resources are under the MIT License, a permissive free software license. It permits reuse within proprietary software provided that the license is distributed with that software.

Hugging Face Transformers: This library is licensed under the Apache License, Version 2.0, which is a permissive free software license similar to the MIT License, but with additional terms concerning patents and contributions.

Flair: Also under the MIT License, Flair is free to use in both open-source and proprietary software, with the same conditions as XLM-RoBERTa and CLNER.

In addition to these licenses, the use of ChatGPT for brainstorming research directions and grammar error checking in this study should be noted. ChatGPT, as an AI language model provided by OpenAI, is designed for a wide range of applications including research assistance and language tasks. The use of ChatGPT in this study aligns with its intended purpose, offering support in developing research ideas and ensuring language accuracy, but not for making autonomous research decisions.

Throughout the research, all resources were used strictly for academic and research purposes, in line with their intended use as stated in their respective licenses. This approach ensures compliance with legal and ethical standards, supporting the integrity and reproducibility of the research.