Self-Adaptive Named Entity Recognition by Retrieving Unstructured Knowledge

Kosuke Nishida[†‡] **Naoki Yoshinaga**[§] **Kyosuke Nishida**[†]

[†]NTT Human Informatics Laboratories, NTT Corporation
[‡]The University of Tokyo
[§]Institute of Industrial Science, the University of Tokyo
[†]{kosuke.nishida.ap, kyosuke.nishida.rx}@hco.ntt.co.jp
[§]ynaga@iis.u-tokyo.ac.jp

Abstract

Although named entity recognition (NER) helps us to extract domain-specific entities from text (*e.g.*, artists in the music domain), it is costly to create a large amount of training data or a structured knowledge base to perform accurate NER in the target domain. Here, we propose self-adaptive NER, which retrieves external knowledge from unstructured text to learn the usages of entities that have not been learned well. To retrieve useful knowledge for NER, we design an effective two-stage model that retrieves unstructured knowledge using uncertain entities as queries. Our model predicts the entities in the input and then finds those of which the prediction is not confident. Then, it retrieves knowledge by using these uncertain entities as queries and concatenates the retrieved text to the original input to revise the prediction. Experiments on CrossNER datasets demonstrated that our model outperforms strong baselines by 2.35 points in F_1 metric.

1 Introduction

Named entity recognition (NER) helps us to extract entities from text in various domains such as biomedicine (Kim et al., 2003), disease (Doğan et al., 2014), and COVID-19 (Wang et al., 2020). However, accurate neural NER requires a massive amount of training data (Chiu and Nichols, 2016; Ma and Hovy, 2016; Yadav and Bethard, 2018). As well, the annotation of a domain-specific NER dataset costs a lot of money because it requires the involvement of domain experts.

To compensate for the lack of training data in NER, researchers have utilized external knowledge. Traditional feature-based NER uses features based on gazetteers or name lists (Florian et al., 2003; Cohen and Sarawagi, 2004; Luo et al., 2015) as external knowledge. Although recent neural NER methods can even benefit from gazetteers and name lists (Seyler et al., 2018; Liu et al., 2019; Mengge et al., 2020), only a few domains with structured knowledge bases (gazetteers) have this merit. Thus, several studies have resorted to using raw text (unstructured knowledge) to perform weakly-supervised learning on general-domain structured knowledge (Cao et al., 2019; Mengge et al., 2020; Liu et al., 2021a).

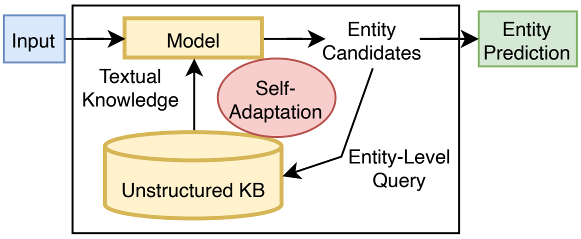

Figure 1: Concept of self-adaptive NER: the model predicts entity candidates to conduct entity-level retrieval from the unstructured KB; then it revises the prediction with reference to the retrieved knowledge.

In this paper, we explore the potential of utilizing unstructured knowledge in the NER task by referring to it at inference time. Our basic idea is inspired by recent retrieval-augmented language models (LMs) (Guu et al., 2020). These models are pre-trained with retrieval-augmented masked language model (MLM), so that they can perform well in open-domain question answering (ODQA) by retrieving relevant unstructured knowledge using a question as a query. However, as we will later confirm in the experiments, the models designed for ODQA are not effective in the NER task because it requires an understanding of many entities in the input text.

To deal with this problem, we propose a retrieval-augmented model capable of determining which entities to focus on in the input text for knowledge retrieval. The proposed **self-adaptive NER (SA-NER) with unstructured knowledge model** searches an unstructured knowledge base (UKB) when it lacks confidence in its prediction. We create the UKB automatically by splitting a raw text corpus into pieces and assigning dense vectors as keys to each piece of unstructured knowledge. To help in understanding local semantics, we design a retrieval system tailored for NER; our model predicts the entities and then retrieves knowledge in terms of those it is not confident in predicting.

To evaluate our method's capability of retrieving useful knowledge about entities, we conducted experiments on various NER datasets (Tjong Kim Sang and De Meulder, 2003; Salinas Alvarado et al., 2015; Liu et al., 2021b), some of which have domain-specific types.

Our contributions are summarized as follows:

- We are the first to integrate retrieval-augmentation into NER. SA-NER retrieves entity-level knowledge dynamically for NER.
- In experiments, SA-NER outperformed strong baselines pre-trained in a supervised and self-supervised fashion by 1.22 to 2.35 points.
- We reveal why knowledge retrieval is useful for NER. We found that our model is effective on entities not included in the general-domain pre-training dataset.

2 Task Settings

We developed SA-NER to solve the problems of NER with unstructured knowledge. NER is a sequence tagging task in which the model inputs a token sequence $X \in V^L$, where V is the vocabulary and L is the maximum sequence length. The model outputs a BIO label sequence of the same length. Let C be the number of types. Then, the number of the BIO labels is $2C + 1$.

SA-NER assumes a corpus as an unstructured knowledge, which is split into token sequences of length L, following the existing retrieval-augmented language model (LM) (Borgeaud et al., 2021), in order to store a large corpus efficiently. We retrieve m pieces of knowledge and concatenate them into X. We feed the concatenated text $X^+ \in V^{(m+1)L}$ to the model.

3 Related Work

Here, we review NER that uses raw text (unstructured knowledge) without structured knowledge, with in-domain structured knowledge, with general-domain structured knowledge, and for pre-training of billion-scale LMs . Also, we review the retrieval-augmented LMs.

3.1 NER with unstructured knowledge

Researchers have utilized various clues to retrieve useful raw text for NER. Traditional NER models focus on surrounding contexts (Sutton and McCallum, 2004; Finkel et al., 2005; Krishnan and Manning, 2006) and linked documents (Plank et al., 2014) to capture non-local dependencies. More recent neural NER models benefit from neighbor sentences to obtain better contextualized word representations (Virtanen et al., 2019; Luoma and Pyysalo, 2020). Meanwhile, Banerjee et al. (2019) and Li et al. (2020) encode knowledge contexts on entity types such as questions, definitions, and examples taken from in-domain structured KBs (*e.g.,* UMLS Meta-thesaurus). In this study, we developed a generic method that retrieves useful raw text (unstructured text) for NER.

Distant supervision (Mintz et al., 2009) uses structured knowledge to annotate raw text with pseudo labels. Performing distantly supervised fine-tuning with in-domain structured knowledge after the MLM pre-training is effective in domain-specific NER (Wang et al., 2021; Trieu et al., 2022). However, domain-specific distant supervised learning depends on the structured knowledge's coverage of the label set of the downstream task.

Weakly supervised learning with general-domain structured knowledge (Cao et al., 2019; Liang et al., 2020; Mengge et al., 2020; Liu et al., 2021a) can transfer general-domain knowledge to the target domain. Its methods learn the entity knowledge through weakly supervised learning, even though the target task has domain-specific entities and types (Liu et al., 2021a). We confirmed that our model achieved a performance gain by using raw text as unstructured knowledge at inference time because the world knowledge cannot be stored in the limited-sized model.

Pre-trained LMs memorize factual knowledge in their models through pre-training on unstructured corpus (Petroni et al., 2019; Cao et al., 2021; Dhingra et al., 2022). Recently, billion-scale generative pre-trained LMs have been proposed (Raffel et al., 2020; Brown et al., 2020). Although the generative models cannot be applied naively to structured prediction tasks such as NER, some papers tackled NER with the generative LMs (Paolini et al., 2021; Yan et al., 2021; Zhang et al., 2022; Chen et al., 2022). One of the advantages of retrieval-augmented LMs over billion-scale LMs is ease of maintenance; For instance, the models can use up-to-date Wikipedia as the UKBs.

3.2 Retrieval-Augmented Language Models

LMs using external knowledge have recently been proposed (Guu et al., 2020; Lewis et al., 2020; Izacard and Grave, 2021; Singh et al., 2021; Borgeaud et al., 2021). However, they focus on language modeling and ODQA, and successful retrieval-augmented LMs in NER have not been reported. They obtain queries for knowledge retrieval in such a way that each query represents the whole input or a fixed-length chunk split from the input. Therefore, they cannot retrieve knowledge that tells the usages of the entities, which is important for NER. In addition, because an input may include many entities, the model should focus on only those entities whose knowledge is not stored in the model. However, retrieval-augmented LMs have not incorporated such a mechanism to create and filter multiple queries.

Wang et al. (2022) and Shinzato et al. (2022) found that retrieving knowledge from the training data is also useful, as it provides knowledge not stored in the trained model. Therefore, we implemented SA-NER in such a way that it uses both labeled and unlabeled UKBs.

de Jong et al. (2022) used a virtual knowledge base whose values are vector representations. Focusing on entity knowledge, they extracted mentions from hyperlinks in Wikipedia to learn their representations. They reported that the virtual KB was less accurate but more efficient than FID (Izacard and Grave, 2021), which reads the input and textual knowledge with attention.

4 Method

Here, we present SA-NER. We explain the construction of the unstructured knowledge base (§4.1), the encoder architecture (§4.2), the two-stage NER algorithm which revises the prediction using the unstructured knowledge (§4.3), the training method (§4.4), and the pre-training method (§4.5).

4.1 Unstructured KB Construction

We create an unlabeled UKB from raw text and a labeled UKB from the training data. We assume in-domain text as a source of unlabeled unstructured knowledge and split it into token sequences of length L, which is equal to the maximum length of the SA-NER inputs. In addition, following Wang et al. (2022), we add the model's training data as labeled unstructured knowledge. We set $L = 64$ to avoid truncating most of the original inputs.

The unstructured knowledge is stored in the UKBs with associated keys. The keys of the sequence are the sentence embedding and the n-gram embeddings. Huang et al. (2021) showed that the average of the token embeddings is more useful for sentence embedding than the first `[CLS]` embedding and that the embeddings in the lower layers are also important, as well as those in the last layer. Therefore, we define the sentence embedding and n-gram embedding as the average pooling of the token representations. The token representations are the concatenations of the frozen BERT input and output, so that both the context-free and contextualized meanings are considered.

To select only entity-like n-grams as the keys, we remove those n-grams that have stop words or have no capital letters. In addition, we use string matching for filtering. We hold only the knowledge that includes the n-grams appearing in the training data for the UKBs used at training time. Also, we hold the knowledge that includes the n-grams appearing in the training or development (test) data for the UKBs at the inference on the development (test) data. Instead of string matching, we can use a summarization-based filtering for n-gram keys, as detailed in Appendix C. We formulate the extraction of a fixed number of representative n-grams from a sequence as an extractive summarization. We use a sub-modular function as the objective (Lin and Bilmes, 2011); thus, the greedy algorithm has a $(1 - 1/e)$ approximation guarantee.

Following Wang et al. (2022), we use the labeled UKB even in training to reduce the training-test discrepancy; in such case, the model does not retrieve the input itself from the labeled UKB.

4.2 Encoder

We use BERT (Devlin et al., 2019) and a linear classifier with a softmax activation as the encoder f. Figure 2 shows the encoder structure. To represent the label information from the labeled knowledge base in the model, we provide additional token-type embeddings. Though the token type is always zero in the conventional BERT model for NER, we use $2C + 3$ token-type IDs;

$$t_i = \begin{cases} 0 & \text{if } x_i \text{ is the original input} \\ 1 & \text{if } x_i^+ \text{ is unlabeled knowledge} \\ l_i + 2 & \text{if } x_i^+ \text{ is labeled knowledge} \end{cases},$$

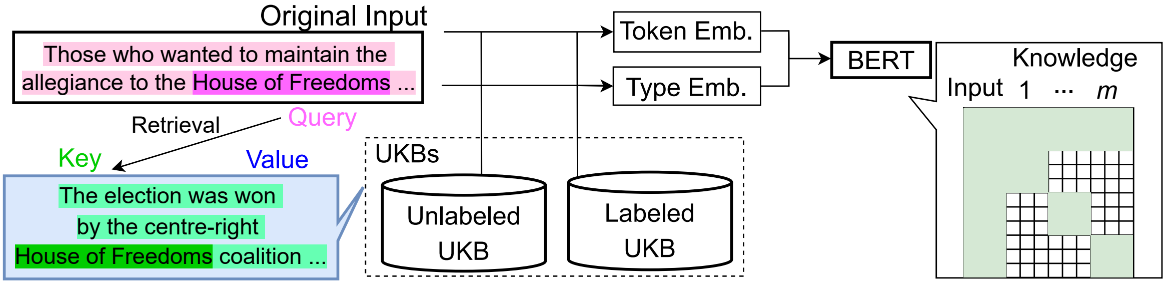
Original Input
Those who wanted to maintain the allegiance to the House of Freedoms ...
Retrieval
Query
Key
Value
The election was won by the centre-right House of Freedoms coalition ...
UKBs
Unlabeled UKB
Labeled UKB
Token Emb.
Type Emb.
BERT
Knowledge
Input 1 ··· m

Figure 2: Overview of our self-adaptive NER with knowledge retrieval from UKBs, which store text with n-gram and sentence embeddings as keys. The labeled UKB has text with labels encoded as token type embeddings. The queries are embeddings of unconfident entities and input. We use a sparse matrix in the self-attention modules in BERT.

where l_i is the label of the labeled knowledge, and X^+ is the concatenated text.

In the self-attention module, we use the sparse attention technique to reduce the space and time complexity from $\mathcal{O}(m^2L^2)$ to $\mathcal{O}(mL^2)$. As shown in Figure 2, we mask the inter-knowledge interaction.[1] Let k be a function that returns 0 as the sentence id if the sequence is the input X and $1, ..., m$ if the sequence is the knowledge. Accordingly, the attention matrix before the softmax operation is

$$A_{ij} = \begin{cases} \frac{\boldsymbol{Q}_i^\top \boldsymbol{K}_j}{\sqrt{d_k}} & \text{if } k(i) = k(j) \text{ or } k(i)k(j) = 0 \\ -\infty & \text{otherwise} \end{cases},$$

where i, j are the token indices, d_k is the number of dimensions of the attention head, and $\boldsymbol{Q}$ and $\boldsymbol{K} \in \mathbb{R}^{(m+1)L \times d_k}$ are query and key matrixes.

4.3 Two-stage Tagging of Self-Adaptive NER

SA-NER performs two-stage tagging, *i.e.*, calculation of $\boldsymbol{P} = f(X)$, and calculation of $\boldsymbol{P}^+ = f(X^+)$. The purpose of the first stage is to find the entities that require additional information and obtain queries for knowledge retrieval. The second stage is to refine the labels with the retrieved knowledge. The motivation behind this design is to retrieve useful entity-wise knowledge to disambiguate individual tokens in NER. We predict the entity spans for entity-level retrieval. We use only the unconfident entities as the entity-based queries in order to exclude unnecessary knowledge from the retrieved results. The pseudo-code of the model is listed in Algorithm 1.

Algorithm 1 Two-stage self-adaptive NER

```
Require: input $X$, KBs, hyperparamters $m, \lambda_{conf}$
Predict probability $\boldsymbol{P} = f(X)$
Compute confidence score $c_e = \min_{i \in I_e} P_{i,\hat{y}_i}$ for each predicted entity $e \in \mathcal{E}$ with span $I_e$
Obtain unconfident entities $\mathcal{U} = \{e | e \in \mathcal{E}, c_e < \lambda_{\mathrm{conf}}\}$
Add the sentence and unconfident-entity embeddings to the queries, $Q$
Initialize the retrieval results $R = \Phi$
for query $q_i$ in the queries, $Q$ do
    Retrieve $m$ nearest-neighbor keys for $q_i$ from the KBs
    Store their values with the distance in $R$
end for
Deduplicate $R$ to obtain top-$m$ knowledge $K_1^m$ from $R$
Output probabilities $\boldsymbol{P}$ and $\boldsymbol{P}^+ = f(X^+ = [X; K_1^m])$
```

We obtain the classification probabilities of the given text $\boldsymbol{P} = f(X) \in \mathbb{R}^{L \times (2C+1)}$ or that of the text with knowledge $\boldsymbol{P}^+ = f(X^+) \in \mathbb{R}^{L \times (2C+1)}$, where the vectors after position L are ignored. The model parameters are shared in the two stages.

First Stage We collect unconfident entities $\mathcal{U}$ in X and feed X to the model to obtain the classification probability $\boldsymbol{P} \in \mathbb{R}^{L \times (2C+1)}$. Then, we extract the entities $\mathcal{E}$ from X in accordance with the predicted labels $\hat{\boldsymbol{y}} = \mathrm{argmax}_c \boldsymbol{P}_{\cdot c} \in \mathbb{N}^L$. The confidence score of a predicted entity e is $c_e = \min_{i \in I_e} P_{i,\hat{y}_i}$, where I_e is the span of $e \in \mathcal{E}$. If the type predictions are inconsistent in an entity (*e.g.*, [B-LOC, I-PER]), we set $c_e = 0$. We collect the unconfident entities $\mathcal{U} \subseteq \mathcal{E}$ whose confidence scores are less than a threshold λ_{conf}.

Then, we obtain the queries, which are the sentence and entity embeddings. The sentence embedding is the average pooling over all token embeddings. Each unconfident entity $u \in \mathcal{U}$ has multiple entity embeddings: average-pooled vectors of n-grams which share at least one token with u. The n-grams are filtered out similarly as in the UKB construction (§ 4.1). E denotes the number of *entity embeddings* (which are embeddings of n-grams overlapping with $u \in \mathcal{U}$). Each token embedding is a concatenation of the BERT input and output.

[1] We implement X^+ as an $(m+1) \times L$ tensor. We calculate three attention matrices: intra-sequence attention ($(m+1) \times L \times L$), knowledge-to-input attention ($m \times L \times L$), and input-to-knowledge attention ($m \times L \times L$). This operation takes advantage of parallel computing on the GPU.

Note that we only consider sentence-to-sentence and entity-to-n-gram matching. We retrieve the top-m nearest neighbors of the sentence embedding from the sentence embeddings in the UKBs and of the entity embeddings from the n-gram embeddings. Then, we select the top-m nearest knowledge from the collected $2(E+1)m$ knowledge while deduplicating the backbone knowledge sequence by keeping the knowledge having the minimum distance.

Second Stage We concatenate the knowledge K_1^m to the input X and obtain the classification probability $\boldsymbol{P}^+ = f(X^+ = [X; K_1^m])$. Finally, the model outputs BIO labels in accordance with $\boldsymbol{P}$ for the tokens in the confident entities and in accordance with $\boldsymbol{P}^+$ for the other tokens.

4.4 Training

To train our two-stage SA-NER, we utilize supervision on the training data to refine unconfident entities and design the loss function.

Unconfident Entity Collection In the training phase, we add the misclassified entities, i.e., those of which the prediction is not correct, to the unconfident entities $\mathcal{U}$ described in §4.3.

Loss Function We use two cross-entropy losses, $\mathcal{L}_1$ for the model prediction without knowledge (the first step) and $\mathcal{L}_2$ for the model prediction with knowledge (the second step). The total loss function is $\mathcal{L}_2 + \lambda_1 \mathcal{L}_1$, where λ_1 is a hyperparameter.

4.5 Pre-training

As is done in retrieval-augmented language models for ODQA (Guu et al., 2020; Borgeaud et al., 2021), we add a retrieval-augmented pre-training stage before the fine-tuning. We propose two methods for NER-aware retrieval-augmented pre-training. The first method uses a general domain NER dataset, CoNLL03 (Tjong Kim Sang and De Meulder, 2003). The model is pre-trained with the method described above (§4.1~§4.4).

The second method involves a large-scale self-supervised pre-training following NERBERT (Liu et al., 2021a). Although the UKB in SA-NER and the pre-training data overlapped in some cases, SA-NER can use the knowledge effectively by referring to it at inference time.

NERBERT The pre-training corpus is Wikipedia. If the consecutive words in the corpus have a hyperlink, the words are labeled as an entity. We categorize such entities with the DBpedia Ontology (Mendes et al., 2012). If the entity exists in the ontology, we categorize it to its type. If it does not exist or it belongs to multiple types, we categorize it to the special "ENTITY" type.

We split the corpus into fixed-length token sequences,[2] and extract the sequences with tokens labeled with the DBpedia types. We reduce the proportion of "ENTITY" labels by using filtering rules and down sampling. The resulting dataset has 33M examples, 939M tokens, and 404 types.

We add a final linear layer with a trainable parameter $\boldsymbol{W}_{\mathrm{pre}} \in \mathbb{R}^{d \times (2C_{\mathrm{pre}}+1)}$ to the top of BERT, where d is the hidden size of BERT and C_{pre} is the number of types. Before fine-tuning, the final layer is replaced with a randomly initialized linear layer whose output dimension is determined by the downstream task. Refer to Appendix B and the original paper (Liu et al., 2021a) for details.

Knowledge Retrieval We use the SA-NER model in the pre-training to reduce the pre-training and fine-tuning discrepancy. We use the pre-training data itself as UKBs. We retrieve knowledge with its pseudo-labels from the data as labeled knowledge and randomly delete the pseudo-labels to make the knowledge unlabeled. We set the deletion probability as 0.95 to simulate downstream tasks where the unlabeled UKB is larger than the labeled UKB. For efficiency, we use Wikipedia hyperlinks as the keys and queries of the retrieval. Instead of a two-stage prediction, we sample m pieces of knowledge that includes an entity in the original input.

5 Evaluation

We conducted experiments on three NER datasets to evaluate the effectiveness of our self-adaptive NER with unstructured knowledge. We used the entity-level F_1 as the metric, following the literature.

5.1 Dataset

CrossNER (Liu et al., 2021b) consists of five domains: politics, science, music, literature, and AI. This small-scale dataset was created by annotating the sentences extracted from the Wikipedia articles in each domain. It provides the textual corpus extracted from Wikipedia for the in-domain pre-training. We used it for the unstructured UKB.[3]

[2] Although the original NERBERT uses a sentence as a unit, we use a fixed length in order to share the setting with UKBs.

[3] We can see if the self-adaptive NER is useful even though the unlabeled knowledge overlaps the NERBERT pre-training

	AI.	Mus.	Lit.	Sci.	Pol.	Avg.	Fin.	CoNLL03
# Train (# NE types)	100 (14)	100 (13)	100 (12)	200 (17)	200 (9)	—	1169 (4)	14987 (4)
BERT[†]	50.37	66.59	59.95	63.73	66.56	61.44	—	—
DAPT[†]	56.36	73.39	64.96	67.59	70.45	66.55	—	—
NERBERT[‡]	60.39	76.23	67.85	71.90	73.69	70.01	—	—
BERT on CoNLL03	56.97 (1.05)	69.10 (1.08)	64.37 (0.73)	65.76 (0.58)	70.16 (0.56)	65.27 (0.80)	72.35 (5.32)	—
REALM-NER on CoNLL03	58.05 (1.15)	71.17 (0.63)	64.58 (0.69)	66.33 (0.66)	69.38 (0.36)	66.56 (0.80)	70.03 (1.35)	—
SA-NER on CoNLL03	**60.31** (1.03)	**72.20** (0.79)	**66.23** (1.30)	**68.22** (0.57)	**71.18** (0.57)	**67.62** (0.85)	**74.02** (2.29)	—
BERT on NERBERT	62.05 (0.66)	76.45 (0.90)	69.68 (0.26)	72.10 (0.67)	74.38 (0.40)	70.93 (0.58)	75.05 (7.47)	90.25 (0.11)
REALM-NER on NERBERT	64.32 (0.31)	77.55 (0.69)	70.42 (0.60)	72.52 (0.42)	74.45 (0.38)	71.85 (0.43)	73.34 (1.74)	89.94 (0.42)
SA-NER on NERBERT	**65.27** (0.95)	**78.71** (0.47)	**71.79** (0.57)	**74.38** (0.19)	**74.63** (0.36)	**72.96** (0.51)	**75.77** (1.01)	**90.49** (0.49)

Table 1: Main results on the test set. The model was pre-trained on CoNLL03 or the NERBERT dataset from the BERT-base-cased model. We ran five experiments with different seeds. Standard deviations are parenthesized. Performances of the previous models are cited from Liu et al. (2021b)[†] and Liu et al. (2021a).[‡] BERT on NERBERT corresponds to our implementation of the NERBERT model.

	AI.	Mus.	Lit.	Sci.	Pol.	Avg.	Fin.	CoNLL03
DistilBERT on CoNLL03	54.16 (1.21)	66.64 (0.54)	60.53 (1.26)	64.14 (0.49)	67.61 (0.70)	62.61 (0.84)	68.78 (6.35)	—
REALM-NER on CoNLL03	53.85 (1.38)	67.03 (0.41)	61.83 (1.38)	64.19 (0.17)	69.09 (0.52)	63.20 (0.54)	70.35 (5.04)	—
SA-NER on CoNLL03	**55.31** (1.03)	**67.25** (1.14)	**61.53** (1.18)	**65.71** (1.03)	**69.36** (0.55)	**63.83** (0.99)	**72.89** (2.71)	—
DistilBERT on NERBERT	59.52 (0.89)	71.60 (1.05)	63.52 (0.47)	69.26 (0.97)	68.88 (0.64)	66.56 (0.80)	73.36 (4.17)	89.23 (0.19)
REALM-NER on NERBERT	60.39 (0.53)	71.39 (0.33)	62.89 (0.19)	68.18 (0.83)	69.79 (0.82)	66.53 (0.54)	74.35 (5.06)	88.54 (0.62)
SA-NER on NERBERT	**61.90** (0.38)	**73.61** (0.45)	**65.48** (0.31)	**70.44** (0.69)	**69.95** (0.90)	**68.27** (0.55))	**75.40** (1.46)	**89.50** (0.30)

Table 2: Main results on the test set. The models were pre-trained from DistilBERT-base-cased.

The label sets are different among the domains.

Finance (Salinas Alvarado et al., 2015) is a medium-scale NER dataset collected from U.S. SEC filings. We used the Wikipedia articles in the finance domain as the textual corpus $\mathcal{D}$ to construct the unlabeled UKB. The label set is person, organization, location, and miscellaneous.

CoNLL03 (Tjong Kim Sang and De Meulder, 2003) is a widely used large-scale NER dataset collected from Reuters news stories between August 1996 and August 1997. We used the Reuters-21578 text classification dataset (Lewis, 1997), which was collected from Reuters in 1987, as $\mathcal{D}$. The label set is the same as that of Finance.

5.2 Compared Models

Our text encoder and tokenizer were the pre-trained BERT-base-cased model (Devlin et al., 2019) or DistilBERT-base-cased model (Sanh et al., 2019). All experiments used the hyperparameters determined on the development set of CrossNER-Politics; refer to Appendix A.

We pre-trained the compared models on the CoNLL03 or NERBERT (Liu et al., 2021a)[4] datasets before fine-tuning. In addition to the BERT model (*i.e.*, BERT with CoNLL03 or NERBERT pre-training), we implemented the NER version of REALM (REALM-NER). For REALM-NER, we replaced the retrieval-augmented MLM of REALM with our retrieval-augmented pre-training methods tailored for NER to assess the effectiveness of our knowledge retrieval. Also, we set $m = 1$, removed the entity-level retrieval, and ignored the labeled UKB. We cited the results of the previous models: BERT, NERBERT, and DAPT (Gururangan et al., 2020), which is the domain-adapted BERT baseline.[5] We compared our model with models consisting of BERT and a linear classifier because the classifier architecture is out of the scope of our study.

5.3 Main Results

Table 1 and Table 2 show the main results. The proposed model outperformed the baselines across all target domains, models, and pre-training datasets.

data. Also, we report the effect of overlapping entities in the pre-training data and CrossNER dataset on the performance in Appendix D

[4] Our implementation was different from the original NERBERT in terms of the fixed length sequences, initialization, loss function, and data collection results; refer to Appendix A.

[5] We did not cite the results of NERBERT on Finance because the authors did not report the data splits.

Method	Acc	Δ
Proposed	77.33 (0.19)	
w/o Entity-level Retrieval	76.21 (0.23)	1.12
w/o Sentence-level Retrieval	76.54 (0.48)	0.79
w/o Confident Entities (*i.e.*, $\lambda_{\text{conf}} > 1$)	76.91 (0.24)	0.42
w/o using First-Step Prediction on $\mathcal{E} \setminus \mathcal{U}$	76.97 (0.33)	0.36
w/o Unlabeled Knowledge	76.23 (0.44)	1.10
w/o Labeled Knowledge	76.82 (0.45)	0.51
NERBERT	75.90 (0.22)	1.43

Table 3: Ablation studies on the development set of the politics domain. Δ shows the drop from the proposed model. Each ablation was conducted in the fine-tuning and evaluation.

The improvement is typically larger in the lower-resource domain with more types, because per-type supervision is limited in such case.

Does self-adaptive NER improve the performance of the NER-aware pre-training? SA-NER outperformed BERT with CoNLL03 and NERBERT pre-training. This indicated that the self-adaptation using unstructured knowledge at inference time has the effect of obtaining additional knowledge that is not stored in the model, even though the model has seen the unstructured knowledge in the pre-training. Moreover, because we can increase the unlabeled UKB after pre-training, the model can acquire new knowledge more efficiently than by conducting additional pre-training.

Does self-adaptive NER improve the performance of the retrieval-augmented LM baseline? SA-NER outperformed REALM-NER. SA-NER retrieves knowledge with the entity-level retrieval from the labeled and unlabeled UKB and encodes large pieces of knowledge due to the sparse attention. These techniques improved the usefulness of the knowledge for NER. The contributions of each component are discussed in the ablation studies. We also found that REALM-NER tends to be not good in the setting # Train > 1000. Because REALM-NER retrieves a piece of knowledge with only the sentence-level query, knowledge retrieval is not always useful in that setting.

5.4 Ablation Studies

Table 3 shows the results of the ablation studies. We used the best performing SA-NER with NERBERT pre-training as the full model. We found that all components of SA-NER improved performance.

Does the entity-level retrieval improve performance? First, we confirmed the usefulness of self-adaptive knowledge retrieval, because knowledge retrieval based on the model's entity prediction is more useful for NER than conventional sentence-level retrieval ($\Delta 1.12$ vs. $\Delta 0.79$). Also, we found that both knowledge retrievals improve NER performance.

Does the distinction about confidence improve the performance? Second, we investigated the efficacy of distinguishing the predicted entities in terms of confidence. The model retrieves knowledge about unconfident entities $\mathcal{U} = \{e | c_e < \lambda_{\text{conf}}, e \in \mathcal{E}\}$, and then refines the prediction for only the unconfident entities with the retrieved knowledge. We set $\lambda_{\text{conf}} > 1$ to remove the distinction. We observed that ignoring confident entities in creating queries is slightly effective ($\Delta 0.42$), because we can restrict the retrieval results to informative knowledge for NER. Then, we used the second-step prediction for all tokens. We found that reusing the first-step prediction for confident entities improved performance slightly ($\Delta 0.36$). Using the first-step prediction is important for confident entities because the retrieved knowledge is likely to be irrelevant to them. We consider that making the distinction is more useful in the smaller m setting where the amount of knowledge is limited.

Do the labeled and unlabeled UKBs improve the performance? Finally, we confirmed that both the labeled and unlabeled UKBs are important ($\Delta 1.10$ and $\Delta 0.51$). The unlabeled UKB covers various contexts, and the labeled UKB has supervision. The two types of UKB have different roles in helping the model recognize entities.

5.5 Discussion

Does the performance of our model depend on the amount of knowledge? Figure 3 plots F_1 score versus the amount of knowledge m. We can see that more pieces of knowledge led to higher F_1 scores. Because the time and space complexity of the sparse attention is linear in the number of pieces of knowledge, the sparse attention is suitable for large m. However, the dense attention did not improve performance in the case of large m. We consider that the sparse attention represents the intra- and inter-sequence interactions more effectively than the naive dense attention can.

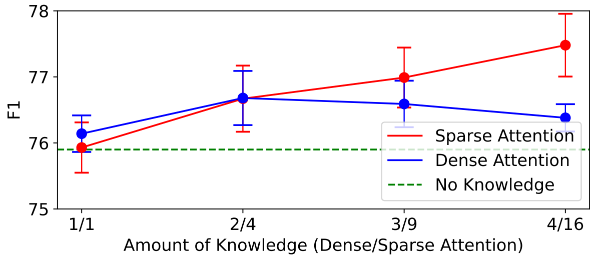

Figure 3: F_1 score versus the amount of knowledge m. The error bars show the standard deviation over five runs. For a fair comparison of the time and space complexity, we compared the methods with different values of m, since our sparse attention runs in $\mathcal{O}(ml^2)$ while the dense attention runs in $\mathcal{O}(m^2l^2)$.

	# Entities	NERBERT	Proposed
All	3472	75.90 (0.22)	77.33 (0.19)
Seen in Training	661	84.05 (1.43)	85.20 (0.21)
Unseen in Training	2811	71.39 (0.29)	73.03 (0.36)
Seen in Pre-Training	3083	77.58 (0.17)	78.83 (0.29)
Unseen in Pre-Training	389	50.90 (1.63)	54.18 (1.85)

Table 4: Detailed results on the development set.

What types of entity require external knowledge? Table 4 lists the results for when the target entities were restricted to each type, which is defined in terms of whether the supervision of an entity was included in the training and pre-training data. The proposed model outperformed NERBERT on all types. The improvement was 1.15 points for the "seen in training" type and 1.64 points for the "unseen in training" type. Therefore, self-adaptation has an effect regardless of whether or not the entity exists in the training data; we also observed this effect in the ablation studies.

Regarding the "unseen in pre-training" type, the proposed model improved performance by 3.28 points. The pre-training dataset collected from Wikipedia shares a lot of entities in the CrossNER dataset created from Wikipedia, and thus whether the tokens are labeled as entities in the pre-training dataset (i.e., the tokens have Wikipedia hyperlinks) has a large effect on performance. We confirmed that PRE-training data is more valuable than one might think, similarly to the findings of Wang et al. (2022) that the reference to the training data at inference time is worthwhile.

Is the self-adaptive NER sensitive to the unconfidence threshold? To investigate the sensitivity of SA-NER to the hyperparameter, we set λ_{conf} to various values at inference time after we trained the model with $\lambda_{\text{conf}} = 0.9$.

λ_{conf}	Acc	Unconfident Proportion
0	76.21 (0.23)	0.00%
0.1	77.14 (0.30)	3.60%
0.5	77.18 (0.23)	5.38%
0.7	77.21 (0.28)	9.60%
0.8	77.30 (0.21)	12.25%
0.9	**77.33** (0.19)	16.63%
0.95	**77.33** (0.16)	21.04%
0.97	77.24 (0.19)	24.74%
0.99	77.13 (0.16)	31.99%
0.995	77.12 (0.17)	37.61%
0.999	77.01 (0.22)	63.63%
1	76.91 (0.24)	100.00%

Table 5: F1 score versus confidence threshold λ_{coef}. Unconfident proportion indicates the proportion of unconfident entities to all entities. We omitted rows $0.2, 0.3, 0.4$, and 0.6, whose performance is the same as that of the rows directly above.

String Matching Filtering	77.33 (0.19)
w/o Knowledge Retrieval (NERBERT)	75.90 (0.22)
w/o Entity-level Retrieval	76.21 (0.23)
Summarization-based Filtering	77.02 (0.40)

Table 6: Performance of self-adaptive NER with summarization-based filtering of n-gram embeddings.

Table 5 shows the results. The performance is on par if $\lambda_{\text{conf}} \in [0.8, 0.95]$. Therefore, SA-NER is not sensitive to λ_{conf}. We also confirmed that modifying the prediction of the high-confidence entities is harmful ($\lambda_{\text{conf}} = 1$) and thus using λ_{conf} is useful. Moreover, we observed that modifying the prediction of certain entities (3.6% of the total number) is important. These entities are ones in which the token-level predictions were inconsistent, and their confidence c_e were set to 0.

Does the self-adaptive NER depend on the filtering method of the n-grams? We compared the two filtering methods for n-gram embeddings in the UKB. The string matching method used the information of the n-grams appearing in the training or development (test) splits in the evaluation on the development (test) set. The summarization-based method just set the maximum number of n-grams in each piece of knowledge.

Table 6 shows the results. Both methods outperformed the no-knowledge baseline (NERBERT) and the ablated model without the entity-level knowledge retrieval. The summarization-based filtering requires fewer assumptions and is computationally efficient, although it is less accurate.

Input	the Association for the Rose in the Fist of Lanfranco Turci and those who wanted to maintain the allegiance to the House of Freedoms coalition.
Knowledge	The election was won in Sardinia by the centre-right House of Freedoms coalition ... voted party with 30.2% .
Prediction	organization → political party
Input	Director Michael Moore partnered with producers Harvey Weinstein and Bob Weinstein in May 2017 to produce and distribute Fahrenheit 11/9 .
Knowledge	... Bob Weinstein, the founders of Miramax Films.
Prediction	politician → politician

Table 7: Qualitative Analysis. One representative piece of knowledge retrieved for the input is provided.

5.6 Qualitative Analysis

Table 7 shows examples of our model. The first example is a case in which the self-adaptation improved the model prediction. The original input itself does not have evidence that the House of Freedoms is a political party. However, the knowledge provides this evidence by mentioning it in the context of an election. The second example is the most common fault in the political domain. Because of the imbalance between the training labels of person and politician, the person entities tend to be misclassified as politician entities. Although both the input and the knowledge indicate that Bob Weinstein is not a politician, the model made the wrong prediction.

6 Conclusions

We proposed SA-NER, which is designed for NER to retrieve knowledge from the labeled and unlabeled UKBs by using unconfident entities and given inputs as queries. It encodes many pieces of knowledge efficiently with sparse attention. In experiments, SA-NER outperformed DistilBERT and BERT baselines pre-trained on the CoNLL03 and NERBERT datasets by 1.22 to 2.35 points. We found that the entity-level retrieval, the focus on the unconfident entities, the labeled and unlabeled UKBs, and the large m that is enabled by the sparse attention all contribute to SA-NER's performance.

We believe that SA-NER can help application providers to develop NER services in their target domain with domain-specific entity types that they have defined, even if they do not have an annotated dataset sufficiently.

Limitations

SA-NER would be of benefit to low-resource domains and languages. However, for languages that have no word segmentation, such as Chinese, the method of constructing UKB based on n-grams and capitalization may not be suitable. For such languages, we can use a traditional word segmenter and POS tagger to extract entity-like n-grams. Although we did not conduct any such data preprocessing in our experiments, it may also be useful for English.

Acknowledgement

This work (second author) was partially supported by JSPS KAKENHI Grant Number JP21H03494. We thank all reviewers for their hard work.

References

Pratyay Banerjee, Kuntal Kumar Pal, Murthy Devarakonda, and Chitta Baral. 2019. Knowledge guided named entity recognition for biomedical text. *arXiv preprint arXiv:1911.03869*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *ACL-IJCNLP*, pages 1860–1874.

Yixin Cao, Zikun Hu, Tat-seng Chua, Zhiyuan Liu, and Heng Ji. 2019. Low-resource name tagging learned with weakly labeled data. In *EMNLP-IJCNLP*, pages 261–270.

Xiang Chen, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, Huajun Chen, and Ningyu Zhang. 2022. LightNER: A lightweight tuning paradigm for low-resource NER via pluggable prompting. In *COLING*, pages 2374–2387.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*, 4:357–370.

William W Cohen and Sunita Sarawagi. 2004. Exploiting dictionaries in named entity extraction: combining semi-Markov extraction processes and data integration methods. In *KDD*, pages 89–98.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William Cohen. 2022. Mention memory: incorporating textual knowledge into transformers through entity mention attention. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *TACL*, 10:257–273.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *EMNLP*, pages 1286–1305.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*, pages 363–370.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *CoNLL@HLT-NAACL*, pages 168–171.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, pages 8342–8360.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *ICML*, pages 3929–3938.

Junjie Huang, Duyu Tang, Wanjun Zhong, Shuai Lu, Linjun Shou, Ming Gong, Daxin Jiang, and Nan Duan. 2021. WhiteningBERT: An easy unsupervised sentence embedding approach. In *Findings of EMNLP*, pages 238–244.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Vijay Krishnan and Christopher D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *COLING*, pages 1121–1128.

David D. Lewis. 1997. Reuters-21578 text categorization test collection, distribution 1.0.

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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, pages 9459–9474.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *ACL*, pages 5849–5859.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *KDD*, page 1054–1064.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *ACL*, pages 510–520.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019. Towards improving neural named entity recognition with gazetteers. In *ACL*, pages 5301–5307.

Zihan Liu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. 2021a. NER-BERT: A pre-trained model for low-resource entity tagging. *arXiv preprint arXiv:2112.00405*.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2021b. CrossNER: Evaluating cross-domain named entity recognition. In *AAAI*, pages 13452–13460.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *EMNLP*, pages 879–888.

Jouni Luoma and Sampo Pyysalo. 2020. Exploring cross-sentence contexts for named entity recognition with BERT. In *COLING*, pages 904–914.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*, pages 1064–1074.

Pablo Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A multilingual cross-domain knowledge base. In *LREC*, pages 1813–1817.

Xue Mengge, Bowen Yu, Zhenyu Zhang, Tingwen Liu, Yue Zhang, and Bin Wang. 2020. Coarse-to-fine pretraining for named entity recognition. In *EMNLP*, pages 6345–6354.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*, pages 1003–1011.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *ICLR*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Autodiff@NIPS*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*, pages 2463–2473.

Barbara Plank, Dirk Hovy, Ryan McDonald, and Anders Søgaard. 2014. Adapting taggers to Twitter with not-so-distant supervision. In *COLING*, pages 1783–1792.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21:1–67.

Julio Cesar Salinas Alvarado, Karin Verspoor, and Timothy Baldwin. 2015. Domain adaption of named entity recognition to support credit risk assessment. In *ALTA*, pages 84–90.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *EMC2@NeurIPS*.

Dominic Seyler, Tatiana Dembelova, Luciano Del Corro, Johannes Hoffart, and Gerhard Weikum. 2018. A study of the importance of external knowledge in the named entity recognition task. In *ACL*, pages 241–246.

Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2022. Simple and effective knowledge-driven query expansion for QA-based product attribute extraction. In *ACL*, pages 227–234.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In *NeurIPS*, pages 25968–25981.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *SRL@ICML*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*, pages 142–147.

Hai-Long Trieu, Makoto Miwa, and Sophia Ananiadou. 2022. Named entity recognition for cancer immunology research using distant supervision. In *BioNLP@ACL*, pages 171–177.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: BERT for Finnish. *arXiv preprint arXiv:1912.07076*.

Shuohang Wang, Yichong Xu, Yuwei Fang, Yang Liu, Siqi Sun, Ruochen Xu, Chenguang Zhu, and Michael Zeng. 2022. Training data is more valuable than you think: A simple and effective method by retrieving from training data. In *ACL*, pages 3170–3179.

Xuan Wang, Vivian Hu, Xiangchen Song, Shweta Garg, Jinfeng Xiao, and Jiawei Han. 2021. ChemNER: Fine-grained chemistry named entity recognition with ontology-guided distant supervision. In *EMNLP*, pages 5227–5240.

Xuan Wang, Xiangchen Song, Bangzheng Li, Yingjun Guan, and Jiawei Han. 2020. Comprehensive named entity recognition on CORD-19 with distant or weak supervision. *arXiv preprint arXiv:2003.12218*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *ACL: System Demonstrations*, pages 38–45.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *COLING*, pages 2145–2158.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *ACL-IJCNLP*, pages 5808–5822.

Shuai Zhang, Yongliang Shen, Zeqi Tan, Yiquan Wu, and Weiming Lu. 2022. De-bias for generative extraction in unified NER task. In *ACL*, pages 808–818.

A Experimental Setup

Table 8 shows the data statistics. Because the finance dataset provides no development data, we split the front half of the 306 test examples into our development split and the back half into our test split.

We collected the raw text in the finance domain from Wikipedia articles. We used the dump data of Wikipedia Circus Search.[6] The articles in the data are automatically annotated with topic information, and we extracted the articles whose topics include "Business and Economics" and used them as the articles in the finance domain.

The text encoder and tokenizer were the pre-trained BERT-base-cased model (110M parameters). The pre-training took 17 hours on eight NVIDIA Quadro RTX 8000 (48GB) GPUs. The training of the largest CoNLL dataset took 6 hours on one GPU. The hyperparameter settings are listed in Table 9. We set the early stop epoch to five only in CoNLL03 for computational efficiency. We used the Adam optimizer (Kingma and Ba, 2015), PyTorch (ver. 1.10.1)[7] (Paszke et al., 2017), and transformers (ver. 4.15.0)[8] (Wolf et al., 2020). Stop words were implemented with NLTK (ver. 3.7)[9] (Bird et al., 2009). We used faiss (ver. 1.7.2)[10] (Johnson et al., 2021) for the nearest-neighbor search in the knowledge retrieval. We set $L = 64$ for all of the data preprocessing, with a sliding window size of 16. For entities in the sliding window, we used the max operation to select from the two predictions.

We pre-trained the NERBERT model under the same hyperparameter settings as above, without knowledge retrieval (that is, $m = 0$). This pre-training was the different from the original NERBERT in terms of the sequence segmentation, initialization, and data collection results, in addition to the hyperparameters.

	# Train	# Dev	# Test	# Types	UKB
AI.	100	350	431	14	15
Mus.	100	380	456	13	467
Lit.	100	400	416	12	436
Sci.	200	450	543	17	191
Pol.	200	541	651	9	354
Fin.	1169	103	103	4	850
CoNLL03	14987	3466	3684	4	7.5

Table 8: Data Statistics. UKB indicates the size of UKB (MB).

	Pre-Training	Fine-Tuning
Batch size	1024	16
# Epochs	1	300
# Steps	10000	—
# Early stop	—	5/8
m	2	10
n	3	3
λ_{conf}	—	0.9
λ_1	—	0.1
Learning rate	5e-5	5e-5

Table 9: Hyperparameters.

B Our implementation of NERBERT

Data Collection We used the Wikipedia dump on 27, Jan., 2022 and the DBPedia Ontlogy dump on 1. Dec. 2021.[11] Then, we split the corpus into fixed-length token sequences and removed the sequences without entities that were not labeled as "ENTITY."

We reduced the proportion of "ENTITY" labels by using filtering rules and down sampling. We randomly filtered the sentences to reduce these labels. If all entities in a sentence were the top-20 frequent labels, the sentences were randomly removed from the dataset: 30% if the number of "ENTITY" entities was three, 50% if the number was four, and 70% if the number was more than four. In the pre-training, we used weighted sampling. The sampling weight of the sentence was $\min_{0\leq i\leq l} |E_{c_i}|^{-0.3}$, where E_c is the number of entities of type c in the dataset, and c_i is the type of the i-th token. As a result, the final dataset had 33M examples, 939M tokens, and 404 types.[12] With the exception of the loss function, initialization, and the use of the retrieval-augmented model, we followed the procedure of the NERBERT pre-training algorithm.

Loss Function In addition to the cross-entropy loss used in the original NERBERT, we incorporated a multi-task loss to efficiently learn the NER ability by ignoring the very frequent "ENTITY" type in the entity typing. For the entity extraction, we performed three-class classification tasks. We summed the output probabilities of the final linear

[6] https://dumps.wikimedia.org/other/cirrussearch/

[7] https://pytorch.org/

[8] https://github.com/huggingface/transformers

[9] https://www.nltk.org/

[10] https://github.com/facebookresearch/faiss

[11] We used en-specific data, which means that the types are annotated without transitive augmentation. https://databus.dbpedia.org/dbpedia/mappings/instance-types/

[12] Liu et al. (2021a) reported their data has 16.3M examples, 457.6M tokens, and 315 types. However, they had not published their data or the URLs of the dump data before our experiments.

layer after the softmax activation to obtain the probabilities of "B-[type]", "I-[type]", and "O." In the entity typing, we masked the output logits of the final linear layer corresponding to the "ENTITY" label. Then, we performed the $2C_{\text{pre}} - 1$ classification task. The total loss was the sum of the two cross-entropy losses.

Initialization We had to initialize the weight of the final linear layer and the token-type embeddings because of the mismatch of the set of the labels between the downstream and pre-training tasks. Instead of a random initialization from $\mathcal{N}(0, \sigma_0)$, where $\sigma_0 \in \mathbb{R}$ is a fixed standard deviation, we used the learned distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, where $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^d$ is the bias and the standard deviation of the weight of the final linear layer and the token-type embeddings in the pre-trained model.

C Summarization-Based Filtering

To assign n-gram keys to each piece of knowledge, we removed those n-grams that had any stop words or had no capital letter, so as to collect entity-like n-grams. In addition, we used filtering methods based on the string matching and the extractive summarization. The summarization-based filtering enabled us to limit the number of n-grams in each piece of knowledge.

We formulated the extraction of a fixed number of representative n-grams from a sequence as an extractive summarization task, as follows. Here, let $\boldsymbol{h}_i$ be an n-gram embedding whose start position is i, regardless of whether the n-gram is filtered out or not. $\boldsymbol{S} \in \mathbb{R}^{L \times L}$ is the cosine similarity matrix of $\boldsymbol{h}_i$ $(0 \leq i < L)$. We denote the token spans as $\{I_s\}$; each span is a maximal token span that does not include stop words but includes a capital letter. We should extract n-grams from different spans to increase the diversity of n-grams. $\mathcal{I}_s$ is the set of such spans.

We defined the optimization problem as follows: $Z \subseteq \{0, 1, \cdots, L-1\}$ denotes the set of n-grams. We used a sub-modular function as the objective to be maximized, under the constraint $|Z| \leq N_{max}$ (Lin and Bilmes, 2011). The objective function is

$$\mathcal{L}_{\text{cov}}(Z) = \sum_{0 \leq i < L} \min \left(\sum_{j \in Z} s_{ij}, \alpha \sum_{0 \leq k < L} s_{ik} \right),$$

$$\mathcal{L}_{\text{div}}(Z) = \sum_{I_s \in \mathcal{I}_s} \sqrt{\sum_{j \in Z \cap I_s} \left(\frac{1}{L} \sum_{0 \leq i < L} s_{ij} \right)},$$

$$\mathcal{L}_{\text{sum}}(Z) = \mathcal{L}_{\text{cov}}(Z) + \lambda_{\text{div}} \mathcal{L}_{\text{div}}(Z).$$

The hyperparameters are $\alpha = 0.1$, $\lambda_{\text{div}} = 10$, and $N_{\text{max}} = 3$. We also required Z to meet the filtering condition (that is, the inclusion of a capital letter and no stop word). $\mathcal{L}_{\text{cov}}(Z)$ measures the coverage of the n-grams and $\mathcal{L}_{\text{div}}(Z)$ measures the diversity of the n-grams.

Because this objective function is a sub-modular function, the greedy algorithm has a $(1 - 1/e)$ approximation guarantee. Therefore, we can use a lightweight computation to extract the most important n-grams.

Method	Acc	Δ
BERT on CoNLL03	70.16 (0.56)	
BERT on NERBERT (non-overlap)	73.59 (0.19)	3.43
SA-NER on NERBERT (non-overlap)	75.13 (0.19)	4.97
BERT on NERBERT (overlap)	75.90 (0.19)	5.74
SA-NER on NERBERT (overlap)	77.33 (0.19)	7.17

Table 10: Performance on the development set. The models were pre-trained on CoNLL03, NERBERT without the entity overlap, and NERBERT with the entity overlap.

D Effect of Overlapping Entities

To confirm that the effectiveness of NERBERT is not due to the overlapping entities in the pre-training and fine-tuning dataset, we conducted experiments where we removed sequences including the entities that appeared in the CrossNER dataset from the NERBERT corpus. Table 10 shows the results. We confirmed that the NER ability learned from the NERBERT corpus itself improved performance and SA-NER outperformed NERBERT in both settings.

However, we also found that the performance of NERBERT is overestimated because of entity overlap. Brown et al. (2020) and Dodge et al. (2021) also noted that leakage of the benchmark datasets from the pre-training corpus affects the performance of GPT-3 (Brown et al., 2020) and T5 (Raffel et al., 2020). The community should solve this problem in future.