Good Examples Make A Faster Learner Simple Demonstration-based Learning for Low-resource NER

Anonymous ACL submission

Abstract

Recent advances in prompt-based learning have shown strong results on few-shot text classification by using cloze-style templates. Similar attempts have been made on named entity recognition (NER) which manually design templates to predict entity types for every text span in a sentence. However, such methods may suffer from error propagation induced by entity span detection, high cost due to enumeration of all possible text spans, and omission of inter-dependencies among token labels in a sentence. Here we present a simple demonstration-based learning method for NER, which lets the input be prefaced by task demonstrations for in-context learning. We perform a systematic study on demonstration strategy regarding what to include (entity examples, with or without surrounding context), how to select the examples, and what templates to use. Results on in-domain learning and domain adaptation show that the model's performance in low-resource settings can be largely improved with a suitable demonstration strategy (e.g., 4-17% improvement on 25 train instances). We also find that good demonstration can save many labeled examples and consistency in demonstration contributes to better performance. [1]

1 Introduction

Neural sequence models have become the *de facto* approach for named entity recognition (NER) and have achieve state-of-the-art results on various NER benchmarks (Lample et al., 2016; Ma and Hovy, 2016; Liu et al., 2018). However, these data-hungry models often rely on large amounts of labeled data manually annotated by human experts, which are expensive and slow to collect (Huang et al., 2020; Ding et al., 2021b), especially for specialized domains (*e.g.*, research papers). To improve NER performance on low-resource (label scarcity) settings, prior works seek auxiliary supervisions, such as entity dictionary (Peng et al., 2019; Shang et al., 2018; Yang et al., 2018; Liu et al., 2019) and labeling rules (Safranchik et al., 2020; Jiang et al., 2020), to either augment human-labeled data with pseudo-labeled data, or incorporate meta information such as explanation (Lin et al., 2020; Lee et al., 2020, 2021), context (Wang et al., 2021), and prompts (Ding et al., 2021a; Cui et al., 2021) to facilitate training. However, such methods have the following challenges: (1) human efforts to create auxiliary supervisions (e.g., dictionaries, rules, and explanations); (2) high computational cost to make predictions. For example, Ding et al. (2021a) shows effectiveness on entity type prediction given the entity span by constructing a prompt with the structure *"[entity span] is [MASK]"*. However, when the entity span is not given, cloze-style prompts need to be constructed over all the entity candidates in the sentence with the structure *"[entity candidate] is [MASK]"* to make a prediction (Cui et al., 2021). Such brute-force enumerations are often expensive.

In this paper, we propose *demonstration-based learning* (Gao et al., 2021; Liu et al., 2021), a simple-yet-effective way to incorporate automatically constructed auxiliary supervision. The idea was originally proposed in prompt-based learning to show some task examples before the cloze-style template so that the model can better understand and predict the masked slot (Gao et al., 2021). This paper proposes modified version of demonstration-based learning for NER task. Instead of reformatting the NER task into the cloze-style template, we augment the original input instances by appending automatically created task demonstrations and feed them into pre-trained language models (PTLMs) so that the model can output improved token representations by better understandings of the tasks. Unlike existing efforts which require additional human labor to create such auxiliary supervisions, our

[1] Code and data have been uploaded and will be published.

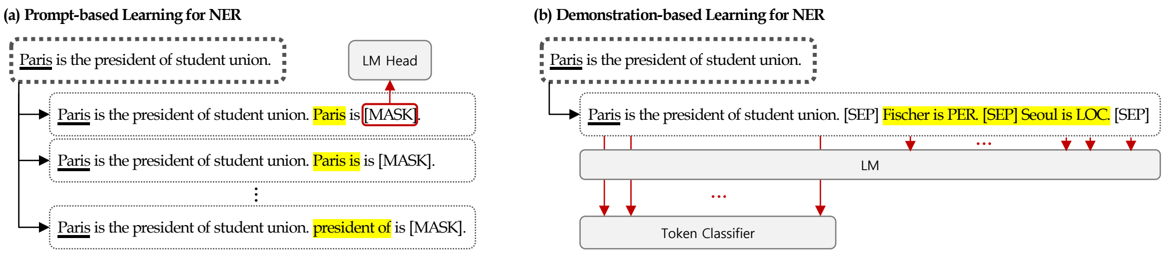


Figure 1: Prompt-based learning frameworks for NER mostly neglect entity span detection which leads to a huge time cost to generate prompts over all the entity candidates in the sentence, while our demonstration-based learning framework integrates prompt into the input itself to make better input representations for the token classification.

model can be automatically constructed by picking up proper task examples from the train data. Moreover, unlike approaches that need to change the format of token classification into cloze-style mask-filling prediction which can neglect latent relationships among token labels, our approach can be applied to existing token classification module in a plug-and-play manner (See Figure 1 (a) vs (b)).

We investigate the effectiveness of task demonstration in two different low-resource settings: (1) in-domain setting which is a standard NER benchmark settings where the train and test dataset come from the same domain; and (2) domain-adaptation setting which uses sufficient labeled data in source domain to solve new tasks in a target domain. Here, we study which variants of task demonstration are useful to train an accurate and label-efficient NER model and further explore ways to adapt the source model to target domain with a small amount of target data. We propose two ways of automatic task demonstration construction: (1) *entity-oriented demonstration* selects an entity example per entity type from train data to construct the demonstration. It allows the model to get a better sense of entity type by showing its entity example; and (2) *instance-oriented demonstration* retrieves instance example similar to input sentence in train data. It allows the model to get a better sense of the task by showing similar instances and their entities.

We show extensive experimental results on CoNLL03, Ontonotes 5.0 (generic domain), and BC5CDR (biomedical domain) over 3 different templates and 5 selection/retrieval strategies for task demonstrations. For *entity-oriented demonstration*, we present 3 selection strategies to choose appropriate entity example per entity type: (1) `random` randomly selects entity example per entity type; (2) `popular` selects the entity example which occurs the most per entity type in the train data; and (3) `search` selects the entity example per entity type that shows the best performance in the development set. And for *instance-oriented demonstration*, we present 2 retrieval strategies to choose appropriate instance example (`SBERT` (Reimers and Gurevych, 2019) vs. `BERTScore` (Zhang et al., 2020)).

Our findings include: (1) good demonstration can save many labeled examples to reach a similar level of performance in low-resource settings. Our approach consistently outperforms standard fine-tuning by up to 3 points in terms of F1 score (p-value < 0.02); (2) demonstration becomes more effective when we also provide context. For example, not only showing 'Fischler is PER', but also the sentence that contains 'Fischler' as person, such as 'France backed Fischler's proposal'; and (3) consistency in demonstration contributes to better performance. Our experiments show that using consistent demonstration for all instances rather than varying per instance lead to better performance

2 Related Works

NER with additional supervision Recent attempts addressing label scarcity have explored various types of human-curated resources as auxiliary supervision. One of the research lines to exploit such auxiliary supervision is distant-supervised learning. These methods use entity dictionaries (Peng et al., 2019; Shang et al., 2018; Yang et al., 2018; Liu et al., 2019) or labeling rules (Safranchik et al., 2020; Jiang et al., 2020) to generate noisy-labeled data for learning a NER model. Although these approaches largely reduce human efforts in annotation, the cross-entropy loss may make the model be overfitted to the wrongly labeled tokens due to noisy labels (Meng et al., 2021). Another line of research is incorporating such auxiliary supervision during training and inference in a setting of supervised learning. These approaches usually incorporate external information that is encoded

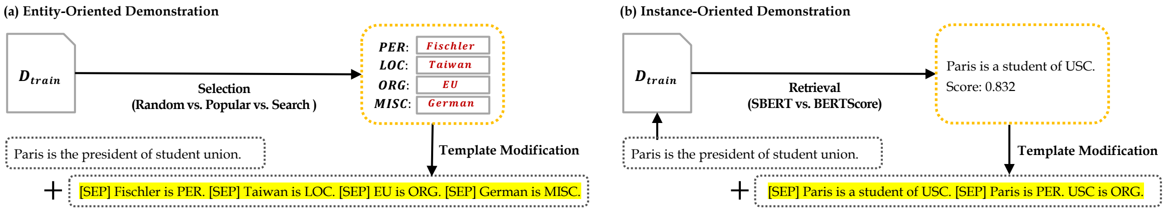


Figure 2: **Task Demonstration for NER.** (a) Entity-oriented demonstration selects an entity example per each entity type from the train data to append to the sentence; while (b) instance-oriented demonstration retrieves an instance from the train data to append to the sentence (along with the entities therein).

including POS labels, syntactic constituents, dependency relations (Nie et al., 2020; Tian et al., 2020), explanations (Lin et al., 2020; Lee et al., 2020, 2021), retrieved context (Wang et al., 2021) and prompts (Ding et al., 2021a; Cui et al., 2021).

Demonstration-based Learning Providing a few training examples in a natural language prompt has been widely explored in autoregressive LMs (Brown et al., 2020; Zhao et al., 2021). Such prompt augmentation is called demonstration-based learning (Gao et al., 2021). This is designed to let prompt be prefaced by a few examples before it predicts label words for *[MASK]* in the cloze-style question. Recent works on this research line explore a good selection of training examples (Gao et al., 2021) and permutation of them as demonstration (Kumar and Talukdar, 2021).

3 Problem Definition

In this section, we introduce basic concepts of named entity recognition, standard fine-tuning for sequence labeling, and domain adaptation for sequence labeling. We then formally introduce our goal – generating task demonstration and then developing a learning framework that uses them to improve NER models.

3.1 Named Entity Recognition

Here, we let $\mathbf{x} = [x^{(1)}, x^{(2)}, \dots x^{(n)}]$ denote the sentence composed of a sequence of n words and $\mathbf{y} = [y^{(1)}, y^{(2)}, \dots y^{(n)}]$ denote the sequence of NER tags. The task is to predict the entity tag $y^{(i)} \in \mathcal{Y}$ for each word $x^{(i)}$, where $\mathcal{Y}$ is a pre-defined set of tags such as {B-PER, I-PER, ..., O}. In *standard fine-tuning*, NER model $\mathcal{M}$ parameterized by θ is trained to minimize the cross entropy loss over token representations $\mathbf{h} = [h^{(1)}, h^{(2)}, \dots h^{(n)}]$ which are generated from the pre-trained contextualized embedder as follows:

$$\mathcal{L} = -\sum_{i=1}^{n} \log f_{i,y_i}(\mathbf{h}; \boldsymbol{\theta}) \quad (1)$$

where f is the model's predicted conditional probability that can be either from linear or CRF layer.

3.2 In-domain Low-resource Learning

We let $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ denote the labeled train and test dataset, respectively, consisting of $\{(\mathbf{x_i}, \mathbf{y_i})\}$. Here, we expect the number of labeled instances in $\mathcal{D}_{\text{train}}$ is extremely limited (e.g., $n < 50$). Given such small labeled instances, our goal is to train an accurate NER model with task demonstrations compared to standard fine-tuning and show the effectiveness of demonstration-based learning. We evaluate the trained models on $\mathcal{D}_{\text{test}}$.

3.3 Low-resource Domain Adaption

Domain adaptation aims to exploit the abundant data of well-studied source domains to improve the performance in target domains of interest. We consider two different settings: (1) *label-sharing* setting in which the label space $\mathbf{L} = \{l_1, \dots, l_{|L|}\}$ (e.g., $l_i = PERSON$) of source-domain data $\mathcal{S}$ and target-domain data $\mathcal{T}$ are equal; (2) *label-different* setting which $\mathbf{L}$ is different.

In domain adaptation, we first train a model $\mathcal{M}_s$ on source-domain data $\mathcal{S}$. Next, we initialize the weights of the new model $\mathcal{M}_t$ by weights of $\mathcal{M}_s$. Here, we can either transfer the whole model weights or only the weights of contextualized embedder from $\mathcal{M}_s$ to $\mathcal{M}_t$. Then, we further tune $\mathcal{M}_t$ on target-domain data $\mathcal{T}$. In our preliminary experiments, we find that transferring only the embedder from $\mathcal{M}_s$ to $\mathcal{M}_t$ is much more effective than transferring the whole model weights (See first rows in Table 2 and Table 3). For this paper, we focus on the effectiveness of our models to adapt to the target domain with a $\mathcal{T}$, for which the number of instances is extremely limited. We then

Input X	Paris is the president of student union.
Label space L	PER, ORG
Selected Entity e	PER: Fischler , ORG: EU
Retrieved Instance s	Paris is a student of USC.

(a) Entity-Oriented Demonstration

no-context	Fischler is PER. [SEP] EU is ORG.
context	France backed Fischler's proposal. Fischler is PER. [SEP] EU rejects German call. EU is ORG.
lexical	France backed PER's proposal. [SEP] ORG rejects German.

(b) Instance-Oriented Demonstration

context	Paris is a student of USC. Paris is PER. USC is ORG.
lexical	PER is a student of ORG.

Figure 3: **Demonstration Template** T. Given input $\mathbf{x}$ and label space $\mathbf{L}$, entity-oriented demonstration selects entity e per each label $l \in \mathbf{L}$ to construct three types of templates (`no-context`, `context`, `lexical`) while instance-oriented demonstration retrieve instance s to create two types of templates (`context`, `lexical`).

compare the results of tasks with demonstration to those without demonstration.

4 Demonstration-based NER

In this work, we focus on how to create effective task demonstration $\tilde{\mathbf{x}}$ to elicit better token representations for $\mathbf{x}$, and then we propose an efficient learning framework that can be improved by the effect of $[\mathbf{x}; \tilde{\mathbf{x}}]$. This section introduces the concepts of *demonstration-based learning*, and provides details of the approach. Here, we study example sampling strategies and templates to construct the demonstration (Sec 4.1) and how we can train the NER model with the demonstration (Sec 4.2).

4.1 Task Demonstration

Task demonstration $\tilde{\mathbf{x}} = [\textit{[SEP]}; \hat{\mathbf{x}}_1; \cdots; \hat{\mathbf{x}}_l]$ is constructed by selecting entity example e or retrieving instance example s from $\mathcal{D}_{\text{train}}$ ($\mathcal{T}_{\text{train}}$ for domain adaptation) and modifying by template T to form $\hat{\mathbf{x}}_i$. The demonstration sequence $\tilde{\mathbf{x}}$ is then appended to the original input $\mathbf{x}$ to create a demonstration-augmented input $[\mathbf{x}; \tilde{\mathbf{x}}]$. Here, *[SEP]* in front of $\tilde{\mathbf{x}}$ is to separate $\mathbf{x}$ and $\tilde{\mathbf{x}}$. The key challenge of constructing task demonstration is to choose appropriate e or s and template T that can be helpful to demonstrate how the model should solve the task. As shown in Figure 2, we categorize the demonstration into (1) *entity-oriented demonstration*; and (2) *instance-oriented demonstration* by whether we choose e or s respectively, for demonstration.

Entity-oriented demonstration. Given an entity type label set $\mathbf{L} = \{l_1, \dots, l_{|L|}\}$, we select an entity example e per label l from $\mathcal{D}_{\text{train}}$. Then, we modify it using template T. To select e per each l, we first enumerate all the $e \in \mathcal{D}_{\text{train}}$ and create a mapping $\{l_i : [e_1, \dots, e_n] \mid l_i \in \mathbf{L}\}$ between l and corresponding list of entities. Then for each label l, we select e by three selection strategies: (1) `random` randomly chooses e from the list; (2) `popular` chooses e that occurs the most frequently in the list; and (3) `search` conducts grid search over possible entity candidates per label. Here, we samples top-k frequent entities per label, and search over combinations of entity candidates ($= k^{|L|}$). We find the best combination that maximizes the F1 score on the dev set $\mathcal{D}_{\text{dev}}$. Here, $\tilde{\mathbf{x}}_\mathbf{i}$ for every $\mathbf{x}_\mathbf{i}$ is different in `random` while $\tilde{\mathbf{x}}_\mathbf{i}$ for every $\mathbf{x}_\mathbf{i}$ is same in `popular` and `search`.

Instance-oriented demonstration. Given an input $\mathbf{x}$, we retrieve an instance example s that is the most relevant to the input from $\mathcal{D}_{\text{train}}$. Then, we modify the s along with its $\{e, l\} \in s$ by template T. For retrieval, we present two strategies: (1) `SBERT` (Reimers and Gurevych, 2019) retrieves semantically similar sentence using pre-trained bi-encoder. It produces CLS embeddings independently for an input $\mathbf{x}$ and $s \in \mathcal{D}_{\text{train}}$, and compute the cosine similarity between them to rank $s \in \mathcal{D}_{\text{train}}$; (2) `BERTScore` (Zhang et al., 2020), which is originally used as a text generation metric, retrieves token-level semantically similar sentence by computing a sum of cosine similarity between token representations of two sentences. Since the NER task aims to token classification, sentence-level similarity may retrieve a sentence that is semantically relevant but has no relevant entities.

Fixed vs Variable demonstration. As described in previous sections, the demonstration in some strategies varies per instance while in others it stays fixed globally. We can divide the demonstration strategies into two categories: (1) Variable demonstration: `random`, `SBERT`, `BERTScore` (2) Fixed demonstration: `popular`, `search`

Demonstration template. As shown in Figure 3, we select three variants of template T:
(1) `no-context` shows selected e per l with a simple template "e *is* l.", without including the spe-

cific sentence where the entities show up. Between each pair of (e, l) (of different entity labels l), we concatenate with separator *[SEP]*. This template is only applied to the entity-oriented demonstration. (2) `context` in entity-oriented demonstration shows selected e per l along with an instance sentence s that contains e as a type of l. For each triple of (e, l, s), it is modified into "s. e *is* l." and concatenated with *[SEP]*. For instance-oriented demonstration, it shows the retrieved instance s along with all the entities mentioned in the sentence $e \in s$. It is modified into "s. e_1 *is* l_1. ... e_n *is* l_n.". (3) `lexical` in entity-oriented demonstration also shows selected e per l along with an instance sentence s. But here we only show s, which the entity span e is replaced by its label string l. For instance-oriented demonstration, we show retrieved s by replacing $e \in s$ with the corresponding l. We expect such templates can form labeling rules and let the model know how to label the sentence.

4.2 Model Training with Demonstration

Transformer-based standard fine-tuning for NER first feeds the input sentence $\mathbf{x}$ into a transfomer-based PTLMs to get the token representations $\mathbf{h}$. The token representations $\mathbf{h}$ are fed into a CRF layer to get the conditional probability $p_\theta(\mathbf{y} \mid \mathbf{h})$, and the model is trained by minimizing the conditional probability by cross entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{n} \log p_\theta(\mathbf{y} \mid \mathbf{h}) \tag{2}$$

In our approach, we define a neural network parameterized by θ that learns from a concatenated input $[\mathbf{x}; \tilde{\mathbf{x}}]$. For both model training and inference, we feed the input and retrieve the representations:

$$[\mathbf{h}; \tilde{\mathbf{h}}] = [h^{(1)}, \ldots h^{(n)}, \tilde{h}^{(1)}, \ldots \tilde{h}^{(n)}] = \text{embed}([\mathbf{x}; \tilde{\mathbf{x}}]) \tag{3}$$

As shown in Figure 1, we then feed $\mathbf{h}$ into the CRF layer to get predictions and train by minimizing the conditional probability $p_\theta(\mathbf{y} \mid \mathbf{h})$ as Equation 2.

For domain adaptation, we first train $\mathcal{M}_s$ with standard fine-tuning. Then, transfer the weights of embedder of $\mathcal{M}_s$ to $\mathcal{M}_t$ and further fine-tune $\mathcal{M}_t$ with our approach.

5 Experimental Setup

5.1 Datasets

We consider three NER datasets as target tasks. We consider two datasets for a general domain (**CoNLL03** (Tjong Kim Sang, 2002), **Ontonotes 5.0** (Weischedel et al., 2013)) and one dataset for a bio-medical domain (**BC5CDR** (Li et al., 2016)). **CoNLL03** is a general domain NER dataset that has 22K sentences containing four types of general named entities: LOCATION, PERSON, ORGANIZATION, and MISCELLANEOUS entities that do not belong in any of the three categories. **Ontonotes 5.0** is a corpus that has roughly 1.7M words along with integrated annotations of multiple layers of syntactic, semantic, and discourse in the text. Named entities in this corpus were tagged with a set of general 18 well-defined proper named entity types. We split the data following (Pradhan et al., 2013). **BC5CDR** has 1,500 articles containing 15,935 CHEMICAL and 12,852 DISEASE mentions.

Dataset	Label	Train Data	
		25	50
CoNLL03	PER (Person)	$16.0_{\pm 3.52}$	$29.2_{\pm 4.52}$
	LOC (Location)	$15.6_{\pm 3.92}$	$30.4_{\pm 4.07}$
	ORG (Organization)	$21.8_{\pm 2.31}$	$32.6_{\pm 3.77}$
	MISC (Miscellaneous)	$11.0_{\pm 2.52}$	$15.6_{\pm 2.33}$
Ontonotes 5.0	PER (Person)	$10.8_{\pm 2.22}$	$21.4_{\pm 4.02}$
	LOC (Location)	$16.0_{\pm 3.52}$	$25.0_{\pm 7.32}$
	ORG (Organization)	$13.8_{\pm 3.48}$	$24.2_{\pm 6.17}$
	MISC (Miscellaneous)	$23.8_{\pm 5.56}$	$62.6_{\pm 7.93}$
BC5CDR	Disease	$25.8_{\pm 6.01}$	$29.2_{\pm 4.52}$
	Chemical	$51.0_{\pm 7.49}$	$65.8_{\pm 7.12}$

Table 1: **Data statistics.** Average number of entities per each entity type over 5 different subsamples.

5.2 Baselines

To show its effectiveness in few-shot NER, we also show baselines of few-shot NER methods NNShot and StructShot (Yang and Katiyar, 2020). NNshot is simple token-level nearest neighbor classification system while StructShot extends NNshot with a decoding process using abstract tag transition distribution. Here, both the classification model and the transition distribution should be pre-trained on the source dataset. Thus, we consider this as domain adaptation setting.

5.3 Experiments and Implementation Details

We implement all the baselines and our frameworks using PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020). We set the batch size and learning rate to 4 and 2e-5, respectively, and use `bert-base-cased` model for all the experiments. For each variant, we run 50 epochs over 5 different sub-samples and 3 random seeds with early-stopping 20 and show its average and stan-

Demonstration / Method	Strategy	Template	CoNLL03 25	CoNLL03 50	Ontonotes 5.0 25	Ontonotes 5.0 50	BC5CDR 25	BC5CDR 50
BERT+CRF w/o demonstration	-	-	52.72 ±2.44	62.75 ±0.98	38.97 ±4.62	54.51 ±3.27	52.56 ±0.46	60.20 ±2.01
BERT+CRF w/ Instance-oriented demonstration	`SBERT` (`variable`)	`lexical`	48.92 ±2.81	57.68 ±0.37	36.58 ±4.61	44.47 ±2.58	49.41 ±0.94	51.98 ±2.14
		`context`	53.62 ±1.64	64.21 ±1.87	42.18 ±5.21	53.07 ±3.46	54.71 ±2.09	59.78 ±1.47
	`BERTScore` (`variable`)	`lexical`	49.55 ±3.18	58.85 ±1.06	35.42 ±3.88	44.70 ±2.41	49.37 ±0.19	51.61 ±2.45
		`context`	53.97 ±1.52	64.66 ±2.04	37.56 ±5.29	53.13 ±3.22	<u>54.81 ±2.11</u>	59.63 ±1.94
BERT+CRF w/ Entity-oriented demonstration	`random` (`variable`)	`no-context`	53.95 ±1.89	63.31 ±2.14	42.25 ±3.61	55.71 ±3.82	53.58 ±0.48	59.97 ±1.89
		`lexical`	55.20 ±2.24	63.60 ±2.32	44.02 ±4.73	56.31 ±3.83	53.79 ±0.61	59.65 ±1.71
		`context`	54.84 ±2.12	63.51 ±2.83	43.57 ±3.73	56.76 ±3.69	54.08 ±0.97	59.94 ±1.70
	`popular` (`fixed`)	`no-context`	54.34 ±3.33	64.30 ±2.76	43.02 ±4.33	56.65 ±3.35	53.86 ±0.86	60.51 ±1.77
		`lexical`	56.22 ±3.88	<u>64.95 ±2.04</u>	45.31 ±5.02	58.24 ±3.17	54.14 ±0.67	60.67 ±1.58
		`context`	56.52 ±3.34	64.47 ±2.35	<u>45.52 ±4.69</u>	58.40 ±3.24	54.31 ±0.80	<u>61.31 ±1.51</u>
	`search` (`fixed`)	`no-context`	54.63 ±2.12	64.50 ±2.76	42.88 ±5.41	56.96 ±4.09	53.97 ±1.32	60.84 ±2.14
		`lexical`	<u>56.57 ±3.61</u>	**65.11 ±2.71**	44.87 ±5.09	<u>58.51 ±3.42</u>	54.39 ±1.57	60.76 ±2.12
		`context`	**57.00 ±4.03**	64.82 ±3.16	**45.74 ±5.57**	**59.00 ±3.27**	**55.83 ±1.25**	**62.87 ±2.41**

Table 2: **In-domain performance comparison (F1-score)** on CoNLL03, Ontonotes 5.0, and BC5CDR by different number of training instances. We randomly sample k training instances with a constraint that sampled instances should cover all the IOBES labels in the whole dataset. Best variants are **bold** and second best ones are <u>underlined</u>. Scores are average of 15 runs (5 different sub-samples and 3 random seeds) and the backbone LM model is `bert-base-cased`.

Baselines		*Label Sharing* CoNLL03 -> Ontonotes 25	*Label Sharing* CoNLL03 -> Ontonotes 50	*Label Different* CoNLL03 -> BC5CDR 25	*Label Different* CoNLL03 -> BC5CDR 50
BERT+CRF w/o demonstration		61.22 ±1.93	66.44 ±1.75	52.31 ±1.02	62.10 ±1.01
NNShot		46.67 ±5.48	46.34 ±2.66	44.93 ±1.78	48.12 ±2.72
StructShot		43.61 ±4.58	43.02 ±3.19	25.86 ±4.14	27.81 ±2.10
Strategy	**Template**				
`SBERT` (`variable`)	`lexical`	**63.34 ±1.53**	68.52 ±0.98	53.50 ±2.26	60.52 ±0.71
	`context`	62.33 ±1.63	67.86 ±0.89	51.93 ±1.96	60.09 ±1.27
`BERTScore` (`variable`)	`lexical`	62.26 ±1.43	68.68 ±0.25	52.07 ±2.11	59.90 ±0.05
	`context`	62.46 ±1.69	67.46 ±0.79	53.58 ±1.98	58.95 ±0.38
`random` (`variable`)	`no-context`	62.28 ±1.70	69.32 ±1.34	53.61 ±1.04	62.57 ±0.97
	`lexical`	62.41 ±1.85	68.84 ±1.78	53.85 ±1.12	62.30 ±0.75
	`context`	62.58 ±2.20	69.26 ±1.51	54.05 ±0.63	63.04 ±0.31
`popular` (`fixed`)	`no-context`	62.31 ±1.60	69.39 ±1.59	54.33 ±0.80	62.87 ±0.23
	`lexical`	62.50 ±2.41	69.34 ±1.38	54.30 ±1.12	63.05 ±0.45
	`context`	62.59 ±2.38	<u>69.91 ±1.24</u>	54.45 ±0.96	<u>63.40 ±0.33</u>
`search` (`fixed`)	`no-context`	62.38 ±2.47	69.57 ±1.50	54.51 ±2.25	62.93 ±1.96
	`lexical`	62.51 ±2.43	68.93 ±1.69	<u>54.70 ±2.26</u>	62.88 ±2.90
	`context`	<u>62.63 ±2.94</u>	**69.98 ±1.63**	**54.97 ±1.99**	**63.55 ±1.58**

Table 3: **Domain adaptation performance comparison (F1-score)** on Ontonotes 5.0 and BC5CDR by different number of training instances. $\mathcal{M}_s$ is trained on CoNLL03 and $\mathcal{M}_t$ is initialized with embedder of $\mathcal{M}_s$. Scores are average of 15 runs (5 different sub-samples and 3 random seeds) and the backbone LM model is `bert-base-cased`.

dard deviation of F1 scores. Unlike existing sampling methods for few-shot NER (Yang and Katiyar, 2020), in which the training sample refers to one entity span in a sentence, we consider a real-world setting that humans annotate a sentence. We sub-sample data-points by random sampling with a constraint that sampled instances should cover all the BIOES labels (Chiu and Nichols, 2016) in the whole dataset. For Ontonotes, we aggregate all other entity types rather than person, location, and organization into miscellaneous to set the *label sharing* setting for domain adaptation experiments. Table 1 presents statistics of average number of entities per entity type over 5 different sub-samples.

6 Experimental Results

We first compare the overall performance of all baseline models and our proposed framework with the amount of training data 25 and 50 to show the impact of our approach in a low-resource scenario, assuming a task that needs to be annotated from scratch. Then, we show performance analysis to show the effectiveness of our approach and whether the model really learns from the demonstration.

6.1 Performance Comparison

In-domain setting In Table 2, we can observe that most variants of demonstration-based learning consistently and significantly (with p-value < 0.02) outperform the baseline by a margin ranging from 1.5 to 7 F1 score in three low-resource NER datasets (25, 50 train instances respectively). It demonstrates the potential of our approach for serving as a plug-and-play method for NER models.

Domain adaptation setting First, we observe that simple domain adaptation technique can improve the performance (First rows of Table 2 vs. Table 3). Here, we only transfer the embedder weights of $\mathcal{M}_s$ to $\mathcal{M}_t$, and we expect the performance gain can be attributed to the embedder of $\mathcal{M}_s$, which is trained in task adaptive pre-training manner on NER task formats (Gururangan et al., 2020). In Table 3, we can see that the most variants of demonstration-based learning allow the source model $\mathcal{M}_s$ to be adapted to the target domain in fast with a small amount of target data $\mathcal{T}$, compared to baselines without demonstration including few-shot NER methods.

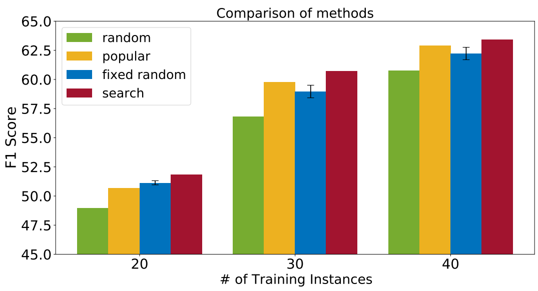


Figure 4: **Performance (F1-score) of randomly select one fixed entity per entity type for demonstration** (`fixed random`) on CoNLL03 by different numbers of train data (20, 30, 40). Error bars show standard deviation across 3 trials using 3 different random seeds for entity selection.

6.2 Performance Analysis

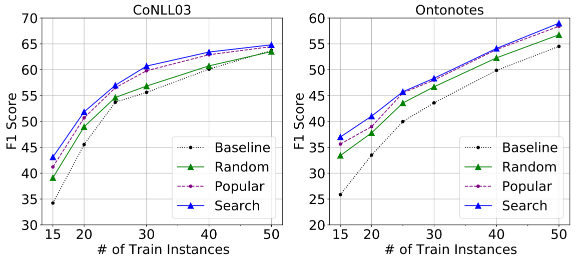
Entity vs. Instance-oriented demonstration. *instance-oriented demonstration* performs worse than *entity-oriented demonstration* due to the difficulty of finding an appropriate similar instance in a low resource train data. In our analysis, we find that the average cosine similarity between retrieved example s and input x is less than 0.4 which shows many of the retrieved examples are not appropriate similar examples to the input.

Fixed vs. Variable demonstration. As mentioned in section 4.1, `random` doesn't pick a fixed set of demonstrations the same way as `popular` and `search`. Instead, it picks random demonstrations for each input instance. In a low-resource setting, there are often no significantly popular entities. Therefore, the fact that `popular` outperforms `random` in our experiments might suggest that the consistency of demonstration selection, rather than popularity of selected entities, is a crucial factor in better few-shot learning. To test this, we randomly select one entity per entity type and attach it as the demonstration to all instances, we call it (`fixed random`). As shown in Figure 4, it outperforms `random` and is on par with `popular` and `search`. We believe this serves as evidence for two hypotheses: (1) consistency of demonstration is essential to performance, and (2) in low-resource settings, the effectiveness of combinations of entities as demonstrations might be a rather random function and not too affected by the combination's collective popularity in the training dataset, which further implies that the idea of `search` is on the right track.

Performance in other model variants To show the effectiveness of demonstration-based learning as plug-and-play method, we present performance in other model variants: `bert-large-cased`, `roberta-base` and `roberta-large`. As shown in Table 4, our method shows consistent improvement over baselines (p-value < 0.05). It shows that demonstration-based learning can be applied to any other model variants and output better contextualized representations for NER tasks and show its potential for scalability.

LM	Strategy	Template	*In-domain* CoNLL03 25	*In-domain* CoNLL03 50	*Label Sharing* CoNLL03 -> Ontonotes 25	*Label Sharing* CoNLL03 -> Ontonotes 50
`BL`	-	-	52.08 $\pm$2.02	66.42 $\pm$2.14	63.50 $\pm$0.96	70.59 $\pm$1.16
`RB`	-	-	59.67 $\pm$4.65	70.17 $\pm$3.93	68.43 $\pm$2.09	74.11 $\pm$1.19
`RL`	-	-	59.15 $\pm$2.93	71.51 $\pm$3.44	68.16 $\pm$2.65	74.45 $\pm$1.02
`BL`	`popular`	`context`	57.60 $\pm$3.37	67.11 $\pm$2.31	64.09 $\pm$2.95	70.88 $\pm$1.09
`RB`	`popular`	`context`	59.76 $\pm$4.27	70.21 $\pm$3.41	69.09 $\pm$2.63	74.53 $\pm$1.32
`RL`	`popular`	`context`	59.99 $\pm$2.16	72.15 $\pm$3.81	68.78 $\pm$2.89	74.93 $\pm$1.07

Table 4: **Performance comparison (F1-score)** with various backbone LMs: `bert-large-cased (BL)`; `roberta-base (RB)`; and `roberta-large (RL)`. Scores are average of 15 runs (5 different sub-samples and 3 random seeds).

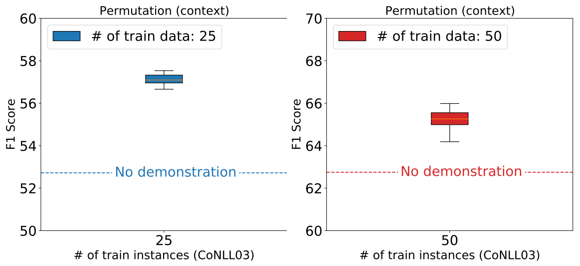


(a) CoNLL03 (b) Ontonotes

Figure 5: **Performance (F1-score) trend with entity-oriented demonstration** on CoNLL03 and Ontonotes by different numbers of train data (15, 20, 30, 40, 50).

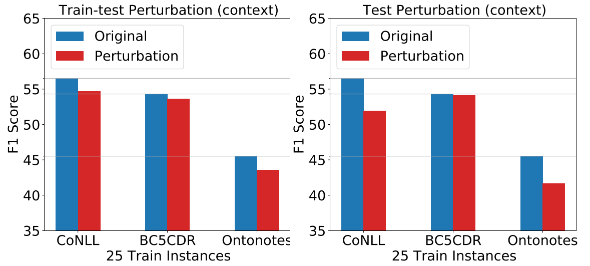
Effectiveness of `search`. `search` consistently outperforms all other strategies. It shows that not only the entity selection, but also the combination of entity examples per each entity type affects the performance. To see whether it consistently outperforms the baseline over various low-resource data points, we show the performance trend of *entity-oriented demonstration* in Figure 2.

Templates of entity-oriented demonstration. *entity-oriented demonstration* becomes more effective when not only showing the entity example per each entity type, but also the corresponding instance example as a context. `context` and `lexical` consistently outperform `no-context`. We explore other templates as well, and these three are the best among them. We present details on Appendix A. To see whether the order of entity type in *entity-oriented demonstration* affects the performance, we present analysis of entity type permutation, e.g., `person` - `organization` - `location` - `miscellaneous`. There is no



(a) 25 instances (b) 50 instances

Figure 6: **Performance (F1-score) variance by different permutation of entity type orders.** Performance is based on template `basic`, strategy `popular`, and CoNLL03.



(a) Test perturbation (b) Train-Test perturbation

Figure 7: **Performance (F1-score) difference between original and perturbed demonstration.** Performance is based on template `basic`, strategy `popular`, and CoNLL03 25 train instances.

Train	Infer	CoNLL03		Ontonotes 5.0		BC5CDR	
		25	50	25	50	25	50
X	X	52.72 ±2.44	62.75 ±0.98	38.97 ±4.62	54.51 ±3.27	52.56 ±0.46	60.20 ±2.01
X	O	51.24 ±2.10	61.02 ±2.05	40.48 ±3.90	52.12 ±3.85	52.16 ±0.55	58.12 ±1.67
O	X	37.71 ±4.65	53.17 ±3.47	31.98 ±4.25	45.27 ±5.19	51.94 ±1.04	57.73 ±1.52
O	O	56.52 ±3.34	64.47 ±2.35	45.52 ±4.69	58.40 ±3.24	54.31 ±0.80	61.31 ±1.51

Table 5: **Effects of demonstration (F1-score)** with/without the demonstration (denoted by “O" and “X", respectively) at training and inference time.

Strategy	Template	CoNLL03		BC5CDR	
		50%	100%	50%	100%
-	-	91.24 ±0.13	91.82 ±0.12	84.58 ±0.17	85.89 ±0.32
`random`	`context`	90.60 ±0.13	91.22 ±0.38	84.32 ±0.07	85.58 ±0.14
`popular`	`context`	90.81 ±0.11	91.85 ±0.07	84.12 ±0.48	85.61 ±0.12

Table 6: **Performance (F1-score) in fully supervised setting** by different percentages of train data.

clear pattern of which entity type order is better (spearman correlation between F1-scores over different entity type orders with 25 and 50 training instances < 0), but all the permutations outperform the baseline as shown in Figure 6, which show that *demonstration-based learning* can be effective regardless of the order (See Appendix Figure 8).

Demonstration perturbation. To investigate whether the model really learns from demonstration, we explore the performance of our approach with perturbed demonstration which selects random entities, labels, and context sentences as demonstration. Here, we present two studies: (1) *Test perturbation* which train with correct demonstration and test with perturbed demonstration; and (2) *Train-test perturbation* which both train and test with perturbed demonstration. Figure 7 shows perturbed demonstration disturbs the model in a large margin for both case. This shows that the model affects by demonstration, and proper demonstration can improve the model's performance. Full results are available in Appendix Table 9.

Effects of demonstration in train & inference. Table 5 shows the effects of demonstration in training and inference stage. A comparison of row 0 with row 3 shows that applying demonstration in the training stage but not in the inference stage would make the model perform worse than the fine-tuning baseline. This is another evidence that consistency of demonstration is essential to the method's performance.

Fully supervised setting. Table 6 shows the performance in fully supervised setting, where the train data is sufficient. We can see that demonstration-based learning yields similar performance as baselines (p-value < 0.1), which shows that demonstrations are rather redundant when data is abundant.

7 Conclusion

In this paper, we propose *demonstration-based learning* for named entity recognition. Specifically, we present *entity-oriented demonstration* and *instance-oriented demonstration* and show that they successfully guide the model towards better understandings of the task in low-resource settings. We observe that *entity-oriented demonstration* is more effective than *instance-oriented demonstration*, and `search` strategy consistently outperforms all other variants. Moreover, we find that consistent demonstration for all the instances is crucial to the superior performance of our approach. We believe that our work provides valuable cost reduction when domain-expert annotations are too expensive and opens up possibilities for future work in automatic demonstration search for few-shot named entity recognition.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. 2021a. Prompt-learning for fine-grained entity typing. *ArXiv preprint*, abs/2108.10604.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021b. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

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 *Proceedings of ACL*.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2020. Few-shot named entity recognition: A comprehensive study. *arXiv preprint arXiv:2012.14978*.

Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu. 2020. Cold-start and interpretability: Turning regular expressions into trainable recurrent neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3193–3207, Online. Association for Computational Linguistics.

Sawan Kumar and Partha Talukdar. 2021. Reordering examples helps during priming-based few-shot learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4507–4518, Online. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. 2020. LEAN-LIFE: A label-efficient annotation framework towards learning from explanation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 372–379, Online. Association for Computational Linguistics.

Dong-Ho Lee, Ravi Kiran Selvam, Sheikh Muhammad Sarwar, Bill Yuchen Lin, Mahak Agarwal, Fred Morstatter, Jay Pujara, Elizabeth Boschee, James Allan, and Xiang Ren. 2021. Autotrigger: Named entity recognition with auxiliary trigger extraction. *ArXiv preprint*, abs/2109.04726.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database : the journal of biological databases and curation*.

Bill Yuchen Lin, Dong-Ho Lee, Ming Shen, Ryan Moreno, Xiao Huang, Prashant Shiralkar, and Xiang Ren. 2020. TriggerNER: Learning with entity triggers as explanations for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8503–8511, Online. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th* innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5253–5260. AAAI Press.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv preprint*, abs/2107.13586.

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.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Xuan Wang, Yu Zhang, Heng Ji, and Jiawei Han. 2021. Distantly-supervised named entity recognition with noise-robust learning and language model augmented self-training. *ArXiv preprint*, abs/2109.05003.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020. Improving named entity recognition with attentive ensemble of syntactic information. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4231–4245, Online. Association for Computational Linguistics.

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. *arXiv preprint arXiv:2101.05779*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and Xuanjing Huang. 2019. Distantly supervised named entity recognition using positive-unlabeled learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2409–2419, Florence, Italy. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Esteban Safranchik, Shiying Luo, and Stephen H. Bach. 2020. Weakly supervised sequence tagging from noisy rules. In *AAAI*.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064, Brussels, Belgium. Association for Computational Linguistics.

Yuanhe Tian, W. Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020. Improving biomedical named entity recognition with syntactic information. *BMC Bioinformatics*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

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)*, pages 1800–1812, Online. Association for Computational Linguistics.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised NER with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2159–2169, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *ArXiv preprint*, abs/2102.09690.

A Template Analysis

Here we present 4 other variants of templates that we have not presented in *entity-oriented demonstration*: (1) `context-all` shows selected e per l along with an instance sentence s that contains e as a type of l. Unlike `context`, it shows all the $e \in s$. For each triple of (e, l, s), it is modified into "s. e_1 *is* l_1. $\ldots e_n$ *is* l_n." and concatenated with *[SEP]*. (2) `lexical-all` shows selected e per l in instance example s and further replaces the entity span e by its label string l. Unlike `lexical`, it replaces all the $e \in s$ by its label string l. (3) `structure` follows augmented natural language format, which is a structured format (Paolini et al., 2021). It shows selected e per l along with an instance sentence s that contains e as a type of l. For each triple of (e, l, s), e in s is replaced with *[* e | l *]* and concatenated with *[SEP]*. (4) `structure-all` also follows augmented natural language format, and shows selected e per l along with an instance sentence s that contains e as a type of l. Unlike `structure` it shows all the $e \in s$. For each triple of (e, l, s), for each e_i in s it is replaced with *[* e_i | l_i *]* and concatenated with *[SEP]*.done Table. 7 shows that `context` and `lexical` are more effective than others.

B Effects of Batch Size

Table 8 shows the main results in Table 2 with batch size 10. Overall performance is much lower than Table 2. It shows that choosing a lower batch size is important in a extremely low resource, where the number of train data is 25 or 50.

Template	CoNLL03				Ontonotes 5.0				BC5CDR			
	50	100	150	200	50	100	150	200	50	100	150	200
-	58.51 $\pm$2.99	69.44 $\pm$4.40	73.94 $\pm$5.69	75.83 $\pm$5.61	46.34 $\pm$4.46	60.36 $\pm$7.52	65.69 $\pm$7.41	68.81 $\pm$7.52	55.68 $\pm$5.33	64.24 $\pm$2.79	68.37 $\pm$2.55	71.09 $\pm$2.84
`no-context`	58.23 $\pm$3.09	69.52 $\pm$3.32	72.99 $\pm$4.63	76.33 $\pm$4.49	49.63 $\pm$3.49	62.10 $\pm$6.53	67.48 $\pm$6.20	69.68 $\pm$7.00	56.04 $\pm$5.34	64.32 $\pm$2.63	68.55 $\pm$2.82	71.14 $\pm$3.29
`context`	59.14 $\pm$2.53	**69.75 $\pm$3.50**	73.35 $\pm$4.24	76.59 $\pm$3.96	52.93 $\pm$4.64	63.37 $\pm$7.02	68.05 $\pm$6.40	**70.23 $\pm$6.28**	57.10 $\pm$4.55	64.42 $\pm$3.14	68.46 $\pm$2.94	71.27 $\pm$3.43
`lexical`	59.62 $\pm$3.12	69.22 $\pm$3.94	**74.23 $\pm$4.26**	77.01 $\pm$4.07	52.69 $\pm$4.47	62.80 $\pm$7.12	67.78 $\pm$6.02	70.02 $\pm$6.86	57.83 $\pm$4.53	64.52 $\pm$3.36	68.51 $\pm$2.57	71.14 $\pm$3.04
`structure`	**60.61 $\pm$2.60**	68.35 $\pm$3.85	73.95 $\pm$4.60	76.56 $\pm$4.38	**53.35 $\pm$3.59**	**63.45 $\pm$6.23**	68.10 $\pm$5.99	69.99 $\pm$6.74	57.45 $\pm$4.79	**64.72 $\pm$2.79**	68.32 $\pm$2.77	**71.55 $\pm$3.20**
`context-all`	58.82 $\pm$2.01	69.22 $\pm$3.37	71.22 $\pm$3.45	76.07 $\pm$4.53	52.85 $\pm$4.23	62.80 $\pm$7.40	**68.22 $\pm$6.18**	69.87 $\pm$6.63	**57.92 $\pm$4.58**	64.69 $\pm$2.72	**68.83 $\pm$2.28**	71.32 $\pm$3.13
`lexical-all`	59.34 $\pm$2.72	69.71 $\pm$3.65	74.16 $\pm$4.47	**77.31 $\pm$4.04**	52.46 $\pm$4.47	63.03 $\pm$7.33	67.22 $\pm$6.82	70.21 $\pm$6.68	56.76 $\pm$5.01	64.42 $\pm$2.91	68.05 $\pm$3.18	71.17 $\pm$3.13
`structure-all`	59.27 $\pm$2.28	69.17 $\pm$3.28	73.69 $\pm$4.43	76.14 $\pm$4.21	53.33 $\pm$4.39	62.69 $\pm$6.48	67.99 $\pm$6.08	70.09 $\pm$6.34	56.99 $\pm$5.56	64.42 $\pm$2.71	68.43 $\pm$2.94	70.92 $\pm$3.12

Table 7: **Template performance comparison (F1-score) in `popular` strategy** on CoNLL03, Ontonotes 5.0, and BC5CDR by different number of training instances. We randomly sample k training instances with a constraint that sampled instances should cover all the IOBES labels in the whole dataset. Best variants are **bold** and second best ones are underlined. For efficient training, here the batch size is 10.

Demonstration	Strategy	Template	CoNLL03		Ontonotes 5.0		BC5CDR	
			25	50	25	50	25	50
No Demonstration	-	-	42.65 $\pm$4.77	60.14 $\pm$3.28	29.11 $\pm$5.21	49.00 $\pm$4.92	50.59 $\pm$3.64	57.44 $\pm$4.51
Instance-oriented Demonstration	`SBERT (variable)`	`lexical`	39.25 $\pm$5.57	54.13 $\pm$4.72	26.41 $\pm$5.84	41.09 $\pm$4.07	47.08 $\pm$5.65	50.78 $\pm$4.77
		`context`	41.09 $\pm$5.82	59.92 $\pm$4.78	30.55 $\pm$6.61	48.46 $\pm$5.03	51.72 $\pm$5.81	57.53 $\pm$4.58
	`BERTScore (variable)`	`lexical`	40.27 $\pm$6.36	55.85 $\pm$4.39	23.84 $\pm$6.10	41.34 $\pm$3.99	47.24 $\pm$5.53	49.73 $\pm$5.43
		`context`	41.42 $\pm$6.5	60.65 $\pm$4.64	25.79 $\pm$5.74	42.21 $\pm$3.23	51.85 $\pm$5.87	56.68 $\pm$5.31
Entity-oriented Demonstration	`random (variable)`	`no-context`	44.19 $\pm$4.98	58.87 $\pm$3.80	33.07 $\pm$7.14	50.02 $\pm$5.48	51.07 $\pm$2.85	58.08 $\pm$3.45
		`lexical`	46.83 $\pm$3.69	59.94 $\pm$3.82	34.52 $\pm$6.58	50.69 $\pm$5.64	51.72 $\pm$2.75	57.62 $\pm$3.33
		`context`	47.39 $\pm$3.89	59.81 $\pm$3.58	35.39 $\pm$7.10	50.80 $\pm$5.63	51.86 $\pm$2.71	58.12 $\pm$2.97
	`popular (fixed)`	`no-context`	46.51 $\pm$4.50	60.67 $\pm$2.97	34.50 $\pm$6.51	52.38 $\pm$4.61	51.12 $\pm$3.28	57.71 $\pm$4.46
		`lexical`	49.92 $\pm$3.52	60.75 $\pm$3.29	36.99 $\pm$6.11	54.56 $\pm$4.59	52.23 $\pm$3.56	58.53 $\pm$4.64
		`context`	50.54 $\pm$3.43	61.08 $\pm$3.10	37.97 $\pm$6.14	54.66 $\pm$4.43	52.78 $\pm$2.71	58.69 $\pm$4.17
	`search (fixed)`	`no-context`	47.80 $\pm$3.45	60.74 $\pm$3.50	34.44 $\pm$6.04	53.06 $\pm$4.78	51.65 $\pm$2.94	58.32 $\pm$4.08
		`lexical`	50.77 $\pm$3.32	61.67 $\pm$3.66	37.41 $\pm$6.74	54.62 $\pm$4.17	52.89 $\pm$3.43	58.80 $\pm$4.23
		`context`	**51.57 $\pm$3.25**	**62.26 $\pm$2.75**	**38.17 $\pm$6.60**	**54.99 $\pm$4.09**	**53.01 $\pm$3.42**	**59.15 $\pm$3.96**

Table 8: **In-domain performance comparison (F1-score)** on CoNLL03, Ontonotes 5.0, and BC5CDR by different number of training instances. We randomly sample k training instances with a constraint that sampled instances should cover all the IOBES labels in the whole dataset. Best variants are **bold** and second best ones are underlined. Scores are average of 15 runs (5 different sub-samples and 3 random seeds) and the backbone LM model is `bert-base-cased`. Unlike Table 2, here the batch size is 10.

Template	Test Perturbation	CoNLL03		Ontonotes 5.0		BC5CDR	
		25	50	25	50	25	50
`no-context`	X	54.34 $\pm$3.33	64.30 $\pm$2.76	43.02 $\pm$4.33	56.65 $\pm$3.35	53.86 $\pm$0.86	60.51 $\pm$1.77
`no-context`	O	53.83 $\pm$3.65	62.86 $\pm$2.16	41.59 $\pm$5.76	54.63 $\pm$3.89	53.06 $\pm$0.84	59.67 $\pm$1.55
`context`	X	56.52 $\pm$3.34	64.47 $\pm$2.35	45.52 $\pm$4.69	58.40 $\pm$3.24	54.31 $\pm$0.8	61.31 $\pm$1.51
`context`	O	51.93 $\pm$5.96	62.21 $\pm$2.66	41.63 $\pm$5.61	53.80 $\pm$4.74	54.12 $\pm$0.95	59.63 $\pm$1.24

Template	Train-Test Perturbation	CoNLL03		Ontonotes 5.0		BC5CDR	
		25	50	25	50	25	50
`no-context`	X	54.34 $\pm$3.33	64.30 $\pm$2.76	43.02 $\pm$4.33	56.65 $\pm$3.35	53.86 $\pm$0.86	60.51 $\pm$1.77
`no-context`	O	54.13 $\pm$2.31	62.88 $\pm$2.36	42.34 $\pm$4.91	55.17 $\pm$3.46	53.16 $\pm$0.70	59.93 $\pm$2.31
`context`	X	56.52 $\pm$3.34	64.47 $\pm$2.35	45.52 $\pm$4.69	58.40 $\pm$3.24	54.31 $\pm$0.8	61.31 $\pm$1.51
`context`	O	54.67 $\pm$3.04	63.93 $\pm$1.92	43.55 $\pm$5.64	56.09 $\pm$3.37	53.59 $\pm$0.82	59.45 $\pm$1.66

Table 9: **Perturbation Analysis.**

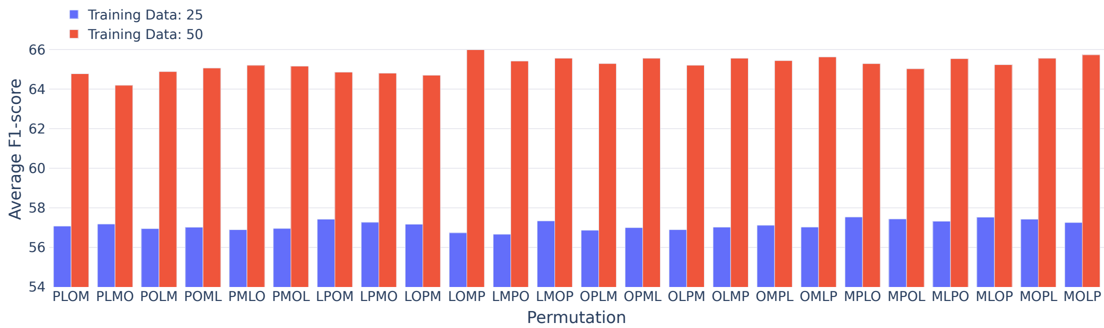


Figure 8: **Performance comparison (F1-score) by different entity type order in entity-oriented demonstration.** Performance is based on template `basic` and strategy `popular`, and dataset is CoNLL03. We construct the demonstration by different entity type order (P: Person, L: Location, O: Organization, M: Miscellaneous). Scores are average of 15 runs (5 different subsamples and 3 random seeds).