CONTAINER: Few-Shot Named Entity Recognition via Contrastive Learning

contrastive

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca J. Passonneau, Rui Zhang
Pennsylvania State University
`{snigdha, arzoo, rjp49, rmz5227}@psu.edu`

Classic contrastive learning idea

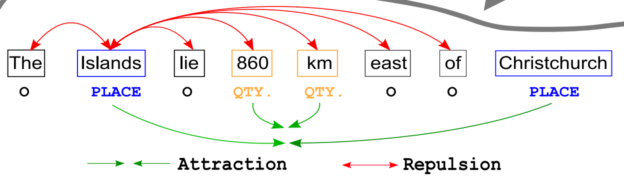


Figure 1: Contrastive learning dynamics of a token (*Islands*) with all other tokens in an example sentence from GUM (Zeldes, 2017). CONTAINER decreases the embedding distance between tokens of the same category (PLACE) while increasing the distance between different categories (QTY. and O).

Abstract

Named Entity Recognition (NER) in Few-Shot setting is imperative for entity tagging in low resource domains. Existing approaches only learn *class-specific* semantic features and intermediate representations from source domains. This affects generalizability to unseen target domains, resulting in suboptimal performances. To this end, we present CONTAINER, a novel contrastive learning technique that optimizes the inter-token distribution distance for Few-Shot NER. Instead of optimizing class-specific attributes, CONTAINER optimizes a generalized objective of differentiating between token categories based on their Gaussian-distributed embeddings. This effectively alleviates overfitting issues originating from training domains. Our experiments in several traditional test domains (OntoNotes, CoNLL'03, WNUT '17, GUM) and a new large scale Few-Shot NER dataset (Few-NERD) demonstrate that on average, CONTAINER outperforms previous methods by 3%-13% absolute F1 points while showing consistent performance trends, even in challenging scenarios where previous approaches could not achieve appreciable performance.

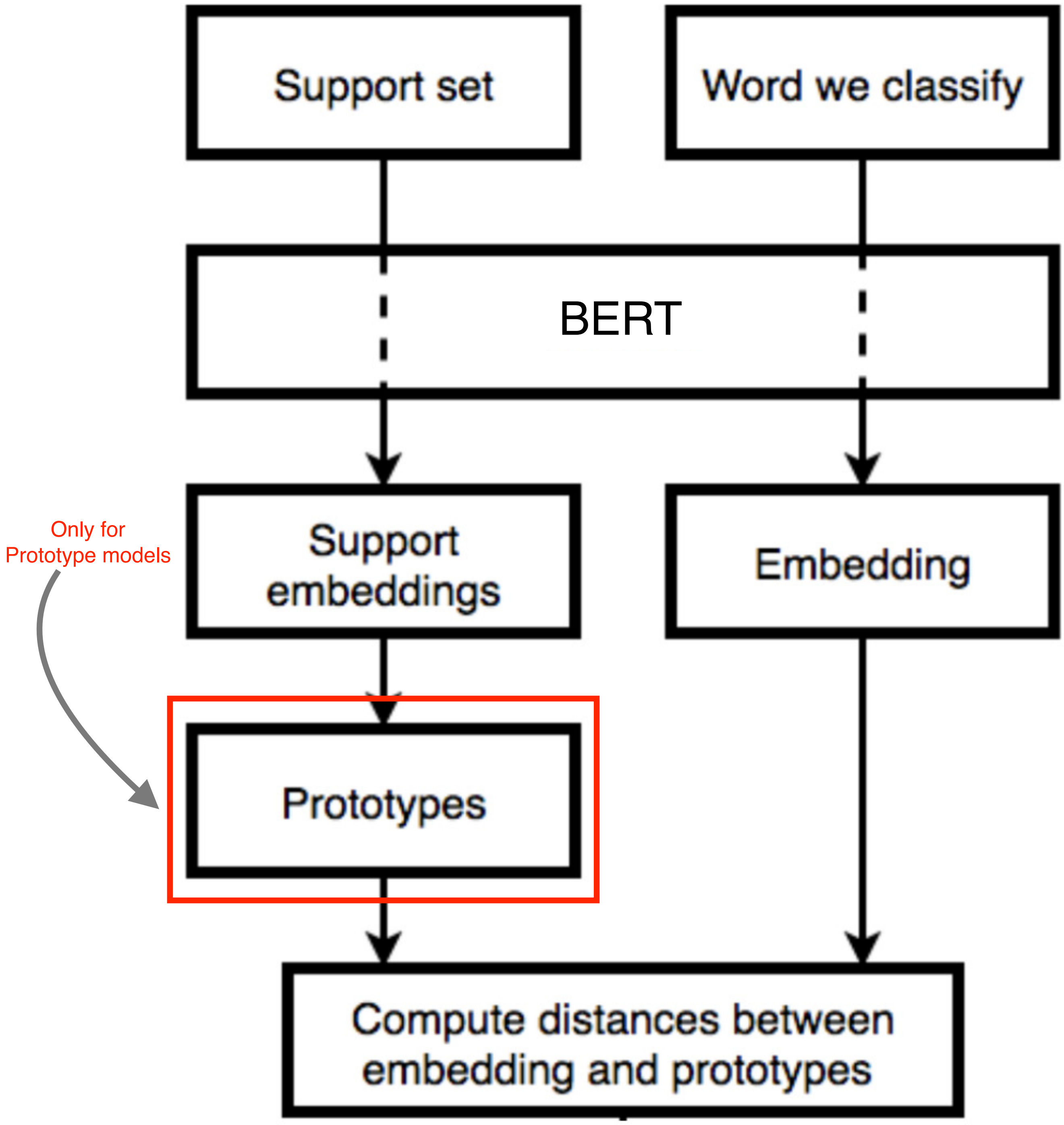
1 Introduction

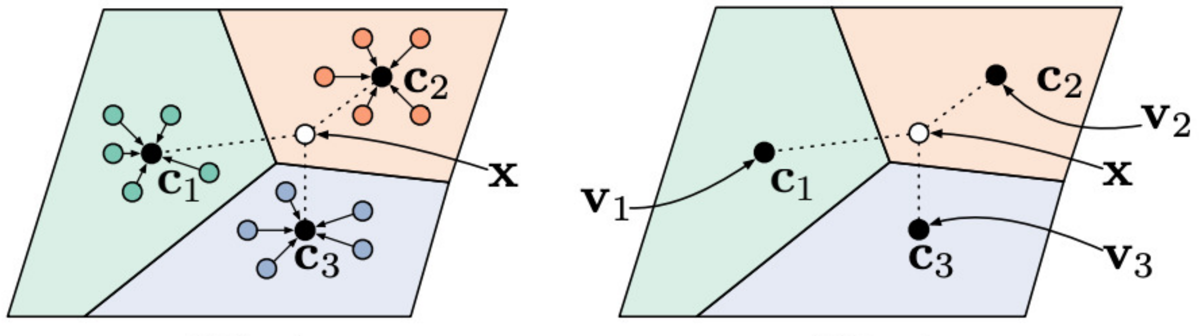
Named Entity Recognition (NER) is a fundamental NLU task that recognizes mention spans in unstructured text and categorizes them into a predefined set of entity classes. In spite of its challenging nature, recent deep-learning based approaches (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016; Peters et al., 2018; Devlin et al., 2018) have achieved impressive performance. As these supervised NER models require large-scale human-annotated datasets, few-shot techniques that can effectively perform NER in resource constraint settings have recently garnered a lot of attention.

Few-shot learning involves learning unseen classes from very few labeled examples (Fei-Fei et al., 2006; Lake et al., 2011; Bao et al., 2020). To avoid overfitting with the limited available data, meta-learning has been introduced to focus on *how to learn* (Vinyals et al., 2016; Bao et al., 2020). Snell et al. (2017) proposed Prototypical Networks to learn a metric space where the examples of a specific unknown class cluster around a single prototype. Although it was primarily deployed in computer vision, Fritzler et al. (2019) and Hou et al. (2020) also used Prototypical Networks for few-shot NER. Yang and Katiyar (2020), on the other hand, proposed a supervised NER model that learns class-specific features and extends the intermediate representations to unseen domains. Additionally, they employed a Viterbi decoding variant of their model as "StructShot".

StructShot authors

Few-shot NER poses some unique challenges that make it significantly more difficult than other few-shot learning tasks. First, as a sequence labeling task, NER requires label assignment according to the concordant context as well as the dependencies within the labels (Lample et al., 2016; Yang and Katiyar, 2020). Second, in NER, tokens that do not refer to any defined set of entities are labeled as `Outside` (`O`). Consequently, a token that is labeled as `O` in the training entity set may correspond to a valid target entity in test set. For prototypical networks, this challenges the notion of entity examples being clustered around a single prototype.





(a) Few-shot (b) Zero-shot

Figure 1: Prototypical networks in the few-shot and zero-shot scenarios. **Left**: Few-shot prototypes $\mathbf{c}_k$ are computed as the mean of embedded support examples for each class. **Right**: Zero-shot prototypes $\mathbf{c}_k$ are produced by embedding class meta-data $\mathbf{v}_k$. In either case, embedded query points are classified via a softmax over distances to class prototypes: $p_\phi(y = k|\mathbf{x}) \propto \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))$.

Few-Shot NER easily overfits

As for Nearest Neighbor based methods such as Yang and Katiyar (2020), these methods are initially "pretrained" with the objective of source class-specific supervision. As a result, the trained weights will be closely tied to the source classes and the network will project training set `O-tokens` so that they get clustered in embedding space. This will force the embeddings to drop a lot of useful features pertaining to its true target entity in the test set. Third, in few-shot setting, there are not enough samples from which we can select a validation set. This reduces the capability of hyperparameter tuning, which particularly affects template based methods where prompt selection is crucial for good performance (Cui et al., 2021). In fact, the absence of held-out validation set puts a lot of earlier language model based few-shot works into question whether their strategy is truly "Few-Shot" (Perez et al., 2021).

To deal with these challenges, we present a novel approach, CONTAINER that harnesses the power of contrastive learning to solve Few-Shot NER. CONTAINER tries to decrease the distance of token embeddings of similar entities while increasing it for dissimilar ones (Figure 1). This explicit contrast between the tokens enables CONTAINER to better capture the label dependencies. Besides, since CONTAINER is trained with a generalized objective, it can effectively avoid the pitfalls of `O-tokens` that the prior methods struggle with. Also, CONTAINER ~~does not require any dataset specific prompt or hyperparameter tuning;~~ standard settings used in prior works (Yang and Katiyar, 2020) works well across different domains in different evaluation settings. CONTAINER also gets some additional advantage compared to the prior methods. Unlike traditional contrastive learners (Chen et al., 2020; Khosla et al., 2020) that optimize similarity objective between point embeddings, CONTAINER optimizes distributional divergence effectively modeling Gaussian Embeddings. This helps CONTAINER extract embedding uncertainties resulting from data scarcity. Furthermore, even with an extremely small number of samples, this strategy lets us finetune the model to better capture unseen novel class distribution - a very useful attribute of our model that none of the previous SOTA models can take advantage of due to overfitting.

Contrastive solves overfitting

GE solve overfitting

A nearest neighbor classification scheme during evaluation reveals that on average, CONTAINER significantly outperforms previous SOTA approaches in a wide range of tests by up to 13% absolute F1-points. In particular, we extensively test our model in both in-domain and out-of-domain experiments as proposed in Yang and Katiyar (2020) in various datasets (CoNLL '03, OntoNotes 5.0, WNUT '17, I2B2). We also test our model in a large dataset recently proposed for Few-Shot NER - Few-NERD (Ding et al., 2021) where CONTAINER outperforms all other SOTA approaches setting a new benchmark result in the leaderboard.

In summary, our contributions are as follows: (1) We propose a novel Few-Shot NER approach CONTAINER that leverages contrastive learning to infer distributional distance of their Gaussian Embeddings. (2) We effectively fine-tune the model representations to adapt to the unseen novel classes, even with a prohibitively low number of support samples. (3) We extensively test CONTAINER in a wide range of experiments using several datasets and evaluation schemes. In almost every case, our model largely outperforms present SOTAs establishing new benchmark results.

2 Task Formulation

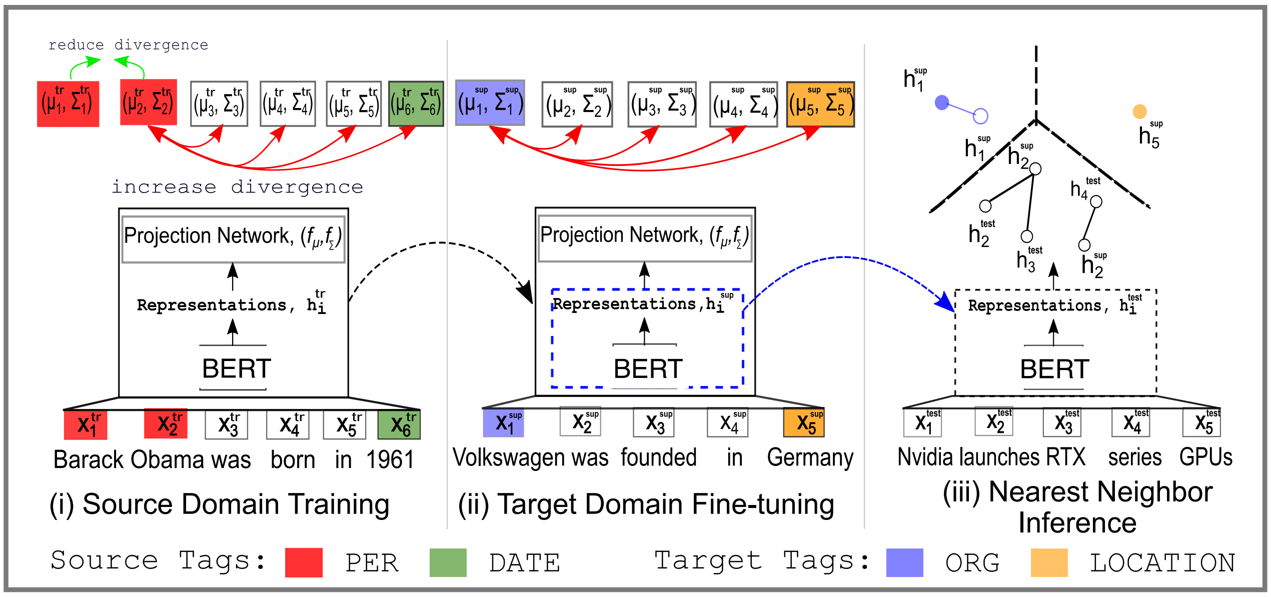
Given a sequence of n tokens $\{x_1, x_2, \ldots x_n\}$, NER aims to assign each token x_i to its corresponding tag label y_i.

Few-shot Setting For Few-shot NER, a model is trained in a source domain with a tag-set $\{C^s_{(i)}\}$ and tested in a data-scarce target domain with a tag-set $\{C^d_{(j)}\}$ where i, j are index of different tags. Since $\{C^s_{(i)}\} \cap \{C^d_{(j)}\} = \emptyset$, it is very challenging for models to generalize to unseen test tags. In an *N-way K-shot* setting, there are N tags in the target domain $|\{C^d_{(j)}\}| = N$, and each tag is associated with a support set with K examples.

Tagging Scheme For fair comparison of CONTAINER against previous SOTA models, we follow an IO tagging scheme where `I-type` represents that all of the tokens are inside an entity, and `O-type` denotes all the other tokens (Yang and Katiyar, 2020; Ding et al., 2021).

Evaluation Scheme To compare with SOTA models in Few-NERD leaderboard (Ding et al., 2021), we adpot *episode evaluation* as done by the authors. Here, a model is assessed by calculating the micro-F1 score over multiple number of test episodes. Each episode consists of a *K-shot* sup-

3 modes of the model



reduce divergence
increase divergence
Projection Network, (f_μ, f_Σ)
Representations, h_i^{tr}
BERT
Barack Obama was born in 1961
(i) Source Domain Training
Representations, h_i^{sup}
Volkswagen was founded in Germany
(ii) Target Domain Fine-tuning
Representations, h_i^{test}
Nvidia launches RTX series GPUs
(iii) Nearest Neighbor Inference
Source Tags: 🟥 PER 🟩 DATE
Target Tags: 🟦 ORG 🟧 LOCATION

Figure 2: Illustration of our proposed CONTAINER framework based on Contrastive Learning over Gaussian Embedddings: (i) Training in source domains using training NER labels PER and DATE, (ii) Fine-tuning to target domains using target NER labels ORG and LOCATION, (iii) Assigning labels to test samples via Nearest Neighbor support set labels.

Support from val
Query from test

port set and a *K-shot* unlabeled query (test) set to make predictions.

While Few-NERD is explicitly designed for episode evaluation, traditional NER datasets (e.g., OntoNotes, CoNLL'03, WNUT '17, GUM) have their distinctive tag-set distributions. Thus, sampling test episodes from the actual test data perturbs the true distribution that may not represent the actual performance. Consequently, Yang and Katiyar (2020) proposed to sample multiple support sets from the original development set and use them for prediction in the original test set. Performance over multiple support sets are averaged to report final performance. We also use this evaluation strategy for these traditional NER datasets.

3 Method

CONTAINER utilizes contrastive learning to optimize distributional divergence between different token entity representations. Instead of focusing on label specific attributes, this contradistinction explicitly trains the model to distinguish between different categories of tokens which results in better generalization. Furthermore, modeling embedding distribution instead of traditional point representation effectively lets CONTAINER capture the embedding uncertainties originating from sample scarcity in target domain. Finally, this contrastive learning technique lets us carefully finetune our model even with an extremely small number of samples without overfitting which is imperative for domain adaptation.

As demonstrated in Figure 2, we first **train** our model in source domains. Next, we **finetune** model representations using few-sample support sets to adapt it to target domains. The training and finetuning of CONTAINER is illustrated in Algorithm 1. Finally, we use an **instance level nearest neighbor classifier** for inference in test sets.

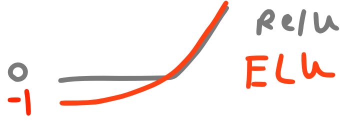
3.1 Model

Figure 2 shows the key components of our model. To generate contextualized representation of sentence tokens, CONTAINER incorporates a pretrained language model encoder PLM. For proper comparison against existing approaches, we use BERT (Devlin et al., 2018) as our PLM encoder. Thus given a sequence of n tokens $[x_1, x_2, \ldots, x_n]$, we take the final hidden layer output of the PLM as the intermediate representations $\boldsymbol{h}_i \in \mathbb{R}^{l'}$.

$$[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n] = \texttt{PLM}([x_1, x_2, \ldots, x_n]) \quad (1)$$

These intermediate representations are then channeled through simple projection layer for generating the embedding. Unlike SimCLR (Chen et al., 2020) that uses projected point embedding for contrastive learning, we assume that token embeddings follow Gaussian distributions. Specifically, we employ projection network f_μ and f_Σ for producing

mu = Linear_1(ReLU(h))
E = ELU(Linear_2(ReLU(h))) + 1



Gaussian distribution parameters:

$$\boldsymbol{\mu}_i = f_\mu(\boldsymbol{h}_i), \quad \boldsymbol{\Sigma}_i = \text{ELU}\left(f_\Sigma(\boldsymbol{h}_i)\right) + (1+\epsilon) \quad (2)$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^l, \boldsymbol{\Sigma}_i \in \mathbb{R}^{l \times l}$ represents mean and diagonal covariance of the Gaussian Embedding respectively; f_μ and f_Σ are implemented as ReLU followed by single layer networks; ELU for exponential linear unit; and $\epsilon \approx e^{-14}$ for numerical stability.

1

3.2 Training in Source Domain

For calculating the contrastive loss, we consider the KL-divergence between all valid token pairs in the sampled batch. Two tokens x_p and x_q are considered as positive examples if they have the same label $y_p = y_q$. Given their Gaussian Embeddings $\mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ and $\mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$, we can calculate their KL-divergence as following:

$$\begin{aligned} D_{\text{KL}}[\mathcal{N}_q \| \mathcal{N}_p] &= D_{\text{KL}}[\mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \| \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)] \\ &= \frac{1}{2}\Big(\text{Tr}(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Sigma}_q) \\ &\quad + (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q) \\ &\quad - l + \log\frac{|\boldsymbol{\Sigma}_p|}{|\boldsymbol{\Sigma}_q|}\Big) \end{aligned} \quad (3)$$

Both directions of the KL-divergence are calculated since it is not symmetric.

Maybe try sqrt(JS)?

$$d(p,q) = \frac{1}{2}\left(D_{\text{KL}}[\mathcal{N}_q \| \mathcal{N}_p] + D_{\text{KL}}[\mathcal{N}_p \| \mathcal{N}_q]\right) \quad (4)$$

We first train our model in resource rich source domain having training data $\mathcal{X}_{\text{tr}}$. At each training step, we randomly sample a batch of sequences (without replacement) $\mathcal{X} \in \mathcal{X}_{\text{tr}}$ from the training set having batch size of b. For each $(x_i, y_i) \in \mathcal{X}$, we obtain its Gaussian Embedding $\mathcal{N}(\mu_i, \Sigma_i)$ by channeling the corresponding token sequence through the model (Algorithm 1: Line 3-6).

We find in-batch positive samples $\mathcal{X}_p$ for sample p and subsequently calculate the Gaussian embedding loss of x_p with respect to that of all other valid tokens in the batch:

Contrastive loss

$$\mathcal{X}_p = \{(x_q, y_q) \in \mathcal{X} \mid y_p = y_q, p \neq q\} \quad (5)$$

$$\ell(p) = -\log \frac{\sum_{(x_q,y_q)\in\mathcal{X}_p} \exp(-d(p,q))/|\mathcal{X}_p|}{\sum_{(x_q,y_q)\in\mathcal{X}, p\neq q} \exp(-d(p,q))} \quad (6)$$

In this way we can calculate the distributional divergence of all the token pairs in the batch (Algorithm 1: Line 7-10). We do not scale the contrastive loss by any normalization factor as proposed by Chen et al. (2020) since we did not find it to be beneficial for optimization.

3.3 Finetuning to Target Domain using Support Set

2

After training in source domains, we finetune our model using a small number of support samples for unseen classes in the target domains. We follow a similar procedure as in the training stage. Since we have only a few samples for finetuning, we take all the few-shot samples in a single batch. When multiple few-shot samples (e.g., 5-shot) are available for the target classes, the model can effectively adapt to the new domain by optimizing KL-divergence of Gaussian Embeddings as in Eq. 4.

!

By contrast, for the 1-shot case, it turns out challenging for models to adapt to the target class distribution. If the model has no prior knowledge about target classes (either from direct training or indirectly from source domain training where the target class entities are marked as `O-type`), a single example might not be sufficient to deduce the variance of the target class distribution. Thus, for 1-shot scenario, we optimize $d'(p,q) = ||\boldsymbol{\mu}_p - \boldsymbol{\mu}_q||_2^2$, the squared Euclidean distance between mean of the embedding distributions, essentially treating them as point embeddings. When the model has direct or indirect prior knowledge about the target classes involved, we still optimize the KL-divergence of the distributions similar to the 5-shot scenario.

We demonstrate in Table 6 that optimizing with squared Euclidean distance gives us slightly better performance in 1-shot scenario. Nevertheless, in all cases with 5-shot support set, optimizing the KL-divergence between the Gaussian Embeddings gives us the best result.

Early Stopping Finetuning a model with a small support set runs the risk of overfitting. Thus an effective early stopping scheme needs to be maintained in the finetuning stage. Unfortunately, without access to a held-out validation set due to data scarcity in the target domain, we cannot keep tabs on the saturation point where we need to stop finetuning. To alleviate this, we rely on the calculated contrastive loss as our early stopping criteria. We continue finetuning our model until the loss increases

$$\ell(p) = -\log \frac{\sum\limits_{(x_q, y_q) \in \mathcal{X}_p} \exp(\underline{-d}(p, q)) / |\mathcal{X}_p|}{\sum\limits_{(x_q, y_q) \in \mathcal{X}, p \neq q} \exp(\underline{-d}(p, q))} \quad \mathbf{(6)}$$

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \quad {}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

$$\ell(f(x, \theta), y) = -\log \left(\frac{\exp f_y(x, \theta)}{\sum_{s=1}^{K} \exp f_s(x, \theta)} \right)$$

Standard practise, gives a small boost

Algorithm 1 Training and Finetuning of CONTAINER

```
Require: Training data $\mathcal{X}_{tr}$, Support Data $\mathcal{X}_{sup}$, Train loss
    function $d_{tr}$, Finetune loss function $d_{ft}$ , $f_\mu$, $f_\Sigma$, PLM
 1: // training in source domain
 2: for sampled (w/o replacement) minibatch $\mathcal{X} \in \mathcal{X}_{tr}$ do
 3:     for all $i \equiv (x_i, y_i) \in \mathcal{X}$ do
 4:         $\boldsymbol{\mu}_i = f_\mu(\mathtt{PLM}(x_i))$ //[Eq. 1]
 5:         $\boldsymbol{\Sigma}_i = ELU(f_\Sigma(\mathtt{PLM}(x_i))) + (1+\epsilon)$ //[Eq. 2]
 6:     end for
 7:     for all $i \equiv (x_i, y_i) \in \mathcal{X}$ do
 8:         Calculate $\ell(i)$ as in Eq. 5 and 6
 9:     end for
10:     $\mathcal{L}_{tr} = \frac{1}{|\mathcal{X}|} \sum_{i \in \mathcal{X}} \ell(i)$
11:     update $f_\mu$, $f_\Sigma$, PLM by backpropagation to reduce $\mathcal{L}_{tr}$
12: end for
13: // finetuning to target domain
14: $\mathcal{L}_{prev} = \infty$
15: $\mathcal{L}_{ft} = \mathcal{L}_{prev} - 1$ //Stable Initialization
16: while $\mathcal{L}_{ft} < \mathcal{L}_{prev}$ do
17:     $\mathcal{L}_{prev} = \mathcal{L}_{ft}$
18:     for all $i \equiv (x_i, y_i) \in \mathcal{X}_{sup}$ do
19:         Calculate $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ using Eq. 1, 2 //Line 4,5
20:     end for
21:     for all $i \equiv (x_i, y_i) \in \mathcal{X}_{sup}$ do
22:         Calculate $\ell(i)$ as in Eq. 5 and 6
23:     end for
24:     $\mathcal{L}_{ft} = \frac{1}{|\mathcal{X}_{sup}|} \sum_{i \in \mathcal{X}_{sup}} \ell(i)$
25:     update $f_\mu$, $f_\Sigma$, PLM by backpropagation to reduce $\mathcal{L}_{ft}$
26: end while
27: return PLM and discard $f_\mu$, $f_\Sigma$
```

to increase. (Algorithm 1: Line 16-17, 24).

3

3.4 Instance Level Nearest Neighbor Inference

After training and finetuning the network with train and support data respectively, we extract the pre-trained language model encoder PLM for inference. Similar to SimCLR (Chen et al., 2020), we found that representations before the projection layers actually contain more information than the final output representation which contributes to better performance, so f_μ and f_Σ projection heads are not used for inference. We thus calculate the representations of the test data from PLM and find nearest neighbor support set representation for inference (Wang et al., 2019; Yang and Katiyar, 2020).

The PLM representations $\boldsymbol{h}_j^{\text{sup}}$ of each of the support token $(x_j^{\text{sup}}, y_j^{\text{sup}}) \in \mathcal{X}_{\text{sup}}$ can be calculated as in Eq. 1. Similarly for test data $\mathcal{X}_{\text{test}}$, we get the PLM representations $\boldsymbol{h}_i^{\text{test}}$ where $x_i^{\text{test}} \in \mathcal{X}_{\text{test}}$. Here we assign x_i^{test} the same label as the support token that is nearest in the PLM representation space:

$$y_i^{\text{test}} = \underset{y_k^{\text{sup}} \text{ where } (x_k^{\text{sup}}, y_k^{\text{sup}}) \in \mathcal{X}_{\text{sup}}}{\arg\min} ||\boldsymbol{h}_i^{\text{test}} - \boldsymbol{h}_k^{\text{sup}}||_2^2 \quad (7)$$

Viterbi Decoding Most previous works (Hou et al., 2020; Yang and Katiyar, 2020; Ding et al., 2021) noticed a performance improvement by using CRFs (Lafferty et al., 2001). The Viterbi decoding stage of the CRF removes false predictions to improve performance. Thus we also employ Viterbi decoding in the inference stage with an abstract transition distribution as in StructShot (Yang and Katiyar, 2020). For the **transition probabilities**, the transition between three abstract tags O, I, and I-other is estimated by counting their occurrences in the training set. Then for the target domain tag-set, these transition probabilities are evenly distributed into corresponding target distributions. The **emission probabilities** are calculated from Nearest Neighbor Inference stage.

Comparing domain transfer results (Table 2) against other tasks (Table 1, 3, 4) we find that, interestingly, if there is no significant domain shift involved in the test data, contrastive learning allows CONTAINER to automatically extract label dependencies, obviating the requirement of extra Viterbi decoding stage.

4 Experiment Setups

Datasets For evaluating the Few-Shot NER capabilities, we use datasets across different domains: General (OntoNotes 5.0 (Weischedel et al., 2013)), Medical (I2B2 (Stubbs and Uzuner, 2015)), News (CoNLL'03 (Sang and De Meulder, 2003)), Social (WNUT'17 (Derczynski et al., 2017)). In addition, we also test on GUM (Zeldes, 2017) that represents a wide variety of texts: interviews, news articles, instrumental texts, and travel guides. The miscellany of domains makes it an extremely challenging dataset to work on. Ding et al. (2021) argue that the distribution of these datasets may not be suitable for proper representation of Few-Shot capability. Thus, they proposed a new large-scale dataset Few-NERD that contains 66 fine-grained entities across 8 coarse-grained entities, significantly richer than previous datasets (4-18 entities).

Baselines We compare the performance of CONTAINER with state-of-the-art Few-Shot NER models on different datasets across several settings. We first measure the model performance in traditional NER datasets in tag-set extension and domain transfer tasks as proposed in Yang and Katiyar (2020). We then evaluate our model in Few-NERD (Ding et al., 2021) dataset that is explicitly designed for Few-Shot NER and compare it against the Few-

Trained on X+Y, tested on Z
6 classes each

Model	1-shot				5-shot			
	Group A	Group B	Group C	Avg.	Group A	Group B	Group C	Avg.
Proto	19.3 ± 3.9	22.7 ± 8.9	18.9 ± 7.9	20.3	30.5 ± 3.5	38.7 ± 5.6	41.1 ± 3.3	36.7
NNShot	28.5 ± 9.2	27.3 ± 12.3	21.4 ± 9.7	25.7	44.0 ± 2.1	51.6 ± 5.9	47.6 ± 2.8	47.7
StructShot	30.5 ± 12.3	28.8 ± 11.2	20.8 ± 9.9	26.7	47.5 ± 4.0	53.0 ± 7.9	48.7 ± 2.7	49.8
CONTaiNER	**32.2 ± 5.3**	**30.9 ± 11.6**	**32.9 ± 12.7**	**32.0**	**51.2 ± 5.9**	**55.9 ± 6.2**	**61.5 ± 2.7**	**56.2**
+ Viterbi	**32.4 ± 5.1**	**30.9 ± 11.6**	**33.0 ± 12.8**	**32.1**	**51.2 ± 6.0**	**56.0 ± 6.2**	**61.5 ± 2.7**	**56.2**

Table 1: F1 scores in Tag Set Extension on OntoNotes. Group A, B, C are three disjoint sets of entity types. Results vary slightly compared to Yang and Katiyar (2020) since they used different support set samples (publicly unavailable) than ours.

supervised

Trained on OntoNotes (General)

Model	1-shot					5-shot				
	I2B2	CoNLL	WNUT	GUM	Avg.	I2B2	CoNLL	WNUT	GUM	Avg.
Proto	13.4 ± 3.0	49.9 ± 8.6	17.4 ± 4.9	17.8 ± 3.5	24.6	17.9 ± 1.8	61.3 ± 9.1	22.8 ± 4.5	19.5 ± 3.4	30.4
NNShot	15.3 ± 1.6	61.2 ± 10.4	22.7 ± 7.4	10.5 ± 2.9	27.4	22.0 ± 1.5	74.1 ± 2.3	27.3 ± 5.4	15.9 ± 1.8	34.8
StructShot	21.4 ± 3.8	**62.4 ± 10.5**	24.2 ± 8.0	7.8 ± 2.1	29.0	30.3 ± 2.1	74.8 ± 2.4	30.4 ± 6.5	13.3 ± 1.3	37.2
CONTaiNER	16.4 ± 1.7	57.8 ± 10.7	24.2 ± 2.9	17.9 ± 1.8	29.1	24.1 ± 1.9	72.8 ± 2.0	27.7 ± 2.2	24.4 ± 2.2	37.3
+ Viterbi	**21.5 ± 1.7**	61.2 ± 10.7	**27.5 ± 1.9**	**18.5 ± 4.9**	**32.2**	**36.7 ± 2.1**	**75.8 ± 2.7**	**32.5 ± 3.8**	**25.2 ± 2.7**	**42.6**

Table 2: F1 scores in Domain Extension with OntoNotes as the source domain. Results vary slightly compared to Yang and Katiyar (2020) since they used different support set samples (publicly unavailable) than ours.

NERD leaderboard baselines. Similar to Ding et al. (2021), we take Prototypical Network based Proto-BERT (Snell et al., 2017; Fritzler et al., 2019; Hou et al., 2020), nearest neighbor based NNShot, and additional Viterbi decoding based Structshot (Yang and Katiyar, 2020) as the main SOTA baselines.

Implementation Details For all of our experiments, we chose the same hyperparameters as in Yang and Katiyar (2020). In order to guarantee proper comparison against prior competitive approaches, we use the same backbone encoder for all methods. Finally, to observe the effect of Viterbi decoding on CONTAINER output, we set the re-normalizing temperature τ to 0.1. We will also release the full source code of CONTAINER upon publication. All the model trainings were conducted using an RTX A6000 GPU.

1

4.1 Tag-set Extension Setting

A common use-case of Few-Shot NER is that new entity types may appear in the same existing text-domain. Thus (Yang and Katiyar, 2020) proposed to experiment tag-set extension capability using OntoNotes (Weischedel et al., 2013) dataset. The eighteen existing entity classes are split into three groups: A, B, and C, each having six classes. Models are tested in each of these groups having few sample support set while being trained in the remaining two groups. During training, all test group entities are replaced with O-tag so that they are not observed. Since the source and destination domains are the same, the training phase will induce some indirect information about unseen target entities. Consequently, during finetuning of CONTAINER, we optimize the KL-divergence between output embeddings as in Eq. 4.

We use the same entity class splits as used by Yang and Katiyar (2020) and used `bert-base-cased` as the backbone encoder for all competing models. Since they could not share the sampled support set for licensing reasons, for a proper comparison, we sampled five sets of support samples for each group and averaged the results, as done by the authors. We show these results in Table 1.

2

4.2 Domain Transfer Setting

In this experiment, a model trained on a source domain is deployed to a previously unseen novel text-domain. Here we take OntoNotes (General) as our source text domain, and evaluate the Few-Shot performance in I2B2 (Medical), CoNLL (News), WNUT (Social) domains as in (Yang and Katiyar, 2020). Additionally, we also evaluate the performance in GUM (Zeldes, 2017) dataset due to its particularly challenging nature. We show these results in Table 2. While all the other domains have almost no intersection with OntoNotes, the target entities in CoNLL are fully contained within OntoNotes entities, which makes it comparable to supervised learning.

Really good result, but no std and no code

Model	5-way		10-way		Avg.
	1∼2 shot	5∼10 shot	1∼2 shot	5∼10 shot	
±1 StructShot	35.92	38.83	25.38	26.39	31.63
ProtoBERT	23.45	41.93	19.76	34.61	29.94
NNShot	31.01	35.74	21.88	27.67	29.08
CONTaiNER	**40.43**	**53.70**	**33.84**	**47.49**	**43.87**
+ Viterbi	**40.40**	**53.71**	**33.82**	**47.51**	**43.86**

Table 3: F1 scores in FEW-NERD (INTRA).

Model	5-way		10-way		Avg.
	1∼2 shot	5∼10 shot	1∼2 shot	5∼10 shot	
StructShot	**57.33**	57.16	**49.46**	49.39±3.08	53.34
ProtoBERT	44.44	58.80	39.09	53.97	49.08
NNShot	54.29	50.56	46.98	50.00	50.46
CONTaiNER	55.95	**61.83**	48.35	**57.12**	**55.81**
+ Viterbi	56.1	**61.90**	48.36	**57.13**	**55.87**

std ???

Table 4: F1 scores in FEW-NERD (INTER).

3

4.3 Few-NERD Setting

For few-shot setting, Ding et al. (2021) proposed two different settings: **Few-NERD (INTRA)** and **Few-NERD (INTER)**. In Few-NERD (INTRA) train, dev, and test sets are divided according to coarse-grained types. As a result, fine-grained entity types belonging to `People`, `Art`, `Product`, `MISC` coarse-grained types are put in the train set, `Event`, `Building` coarse-grained types in dev set, and `ORG`, `LOC` in test set. So, there is no overlap between train, dev, test set classes in terms of coarse-grained types. On the other hand, in Few-NERD (INTER) coarse-grained types are shared, although all the fine-grained types are mutually disjoint. Because of the restrictions of sharing coarse-grained types, Few-NERD (INTRA) is more challenging. Since few-shot performance of any model relies on the sampled support set, the authors also released train, dev, test split for both **Few-NERD (INTRA)** and **Few-NERD (INTER)**. We evaluate our model performance using these provided dataset splits and compare the performance in the Few-NERD leaderboard. All models use `bert-base-uncased` as the backbone encoder. As shown in Table 3 and Table 4, CONTAINER establishes new benchmark results in the leaderboard in both of these tests.

4

supervised setting?

5 Results and Analysis

We prudently analyze different components of our model and justify the design choices made in the scheming of CONTAINER. We also examine the results discussed in "Experiments" section that gives some intuitions about few-shot NER in general.

Works best in 5-shot scenarios

5.1 Overall Results

Table 1-4 demonstrates that overall, in every scenario CONTAINER convincingly outperforms all other baseline approaches. This improvement is particularly noticeable in challenging scenarios, where all other baseline approaches perform poorly. For example, FEW-NERD (intra) (Table 3) is a challenging scenario where the coarse grained entity types corresponding to train and test sets do not overlap. As a result, other baseline approaches face a substantial performance hit, whereas CONTAINER still performs well. In tag-set extension (Table 1), we see a similar performance trend - CONTAINER performs consistently well across the board. Likewise, in domain transfer to a very challenging unseen text domain like GUM (Zeldes, 2017), baseline models performs miserably; yet CONTAINER manages to perform consistently outperforming SOTA models by a significant margin.

Analyzing the results more closely, we notice that while CONTAINER surpasses other baselines in almost every tests, the improvement margin is more prominent in 5-shot cases. Evidently, CONTAINER is able to make better use of multiple few-shot samples thanks to distribution modeling via contrastive Gaussian Embedding optimization. In this context, note that StructShot actually got marginally higher F1-score in 1-shot CoNLL domain adaptation and 1∼2 shot FEW-NERD (INTER) cases. In CoNLL, the target classes are subsets of training classes, so supervised learning based feature extractors are expected to get an advantage in prediction. On the other hand, Ding et al. (2021) carefully tuned the hyperparameters for baselines like StructShot for best performance. We could also improve performance in a similar manner, however for uniformity of model across different few-shot settings, we use the same model architecture in every test. Nevertheless, CONTAINER shows comparable performance even in these cases while significantly outperforming in every other test.

5.2 Training Objective

Traditional contrastive learners usually optimize cosine similarity of point embeddings (Chen et al., 2020). While this has proven to work well in image data, in more challenging NLU tasks like Few Shot NER, we find it to give subpar performance. We see an example scenario in Figure 5, where we show

	Point Embedding	Gaussian Embedding
1-shot	6.37	32.17
5-shot	25.47	51.20

Table 5: F1 scores comparison in OntoNotes Group A. Gaussian Embedding surpasses point embedding based optimization in all cases.

	KL-Gaussian	Euclidean-mean
1-shot	18.78	27.48
5-shot	32.50	31.12

Table 6: F1 scores comparison in Domain Transfer Task with WNUT with different finetune objectives. While optimizing the KL-divergence of the Gaussian Embedding gives superior result in 5-shot, optimizing Euclidean distance of the mean embeddings actually achieve better result in 1-shot.

the performance comparison of tag-set extension in Group A of OntoNotes. Basically, modeling class distribution via Gaussian Distribution allows us to capture the uncertainties in Few-Shot embedding, which is useful in this data scarce environment. This is one of the novel features of CONTAINER that sets it apart from competition.

5.3 Fine-tuning Objective

Same as 3.3

During finetuning, if a model does not have any prior knowledge about the target classes, directly or indirectly, a 1-shot example may not give sufficient information about the target class distribution (i.e. the variance of the distribution). Consequently during finetuning, for 1-shot adaptation to new classes, optimizing euclidean distance of the mean embedding gives better performance. Nevertheless, for 5-shot cases, KL-divergence of the Gaussian Embedding always gives better performance indicating that it takes better advantage of multiple samples. We show this behavior in the best result of domain transfer task with WNUT in Table 6. Since this domain transfer task gives no prior information about target embeddings during training, optimizing KL-divergence in 1-shot fineutuning actually hurts performance a bit compared to euclidean finetuning. However, in 5-shot, KL-finetuning again gives superior performance as it can now adapt better to the novel target class distributions.

5.4 Modeling Label Dependencies

Analyzing the results, we observe that domain transfer (Table 2) sees some good gains in performance from using Viterbi decoding. In contrast, tag-set extension (Table 1) and FEW-NERD (Table 3,4) gets almost no improvement from using Viterbi decoding. This indicates an interesting property of CONTAINER. During domain transfer the text domains have no overlap in train and test set. So, an extra Viterbi decoding actually provides additional information regarding the label dependencies, giving us some nice improvement. On the other hand, the train and target text domain have substantial overlap in both tagset extension and FEW-NERD. Thus the model can indirectly learn the label dependencies through in-batch contrastive learning. Consequently, unless there is a marked shift in the target text domain, we can achieve the best performance even without employing additional Viterbi decoding.

~~6 Related Works~~

Meta Learning The idea of Few-shot learning was popularized in computer vision through Matching Networks (Vinyals et al., 2016). Subsequently, Prototypical Network (Snell et al., 2017) was proposed where class prototypical representations were learned. Test samples are given labels according to the nearest prototype. Later this technique was proven successful in other domains as well. Wang et al. (2019), on the other hand found simple feature transformations to be quite effective in few shot image recognition These metric learning based approaches have also been deployed in different NLP tasks (Geng et al., 2019; Bao et al., 2020; Han et al., 2018; Fritzler et al., 2019).

Contrastive Learning Early progress was made by contrasting positive against negative samples (Hadsell et al., 2006; Dosovitskiy et al., 2014; Wu et al., 2018). Chen et al. (2020) proposed SimCLR by refining the idea of contrastive learning with the help of modern image augmentation techniques to learn robust sets of features. Khosla et al. (2020) leveraged this to boost supervised learning performance as well. In-batch negative sampling has also been explored for learning representation (Doersch and Zisserman, 2017; Ye et al., 2019). Storing instance class representation vectors is another popular direction (Wu et al., 2018; Zhuang et al., 2019; Misra and Maaten, 2020).

Few-Shot NER Established few-shot learning approaches have also been applied in Named Entity Recognition. Fritzler et al. (2019) leveraged prototypical network (Snell et al., 2017) for few shot NER. Inspired by the potency of simple feature extractors and nearest neighbor inference (Wang et al., 2019; Wiseman and Stratos, 2019) in few-Shot learning, Yang and Katiyar (2020) used supervised learner based feature extractors for Few-Shot NER. Pairing it with abstract transition tag Viterbi decoding, they achieved current SOTA result in Few-Shot NER tasks. Huang et al. (2020) proposed noisy supervised pre-training for Few-Shot NER. However, this method requires access to a large scale noisy NER dataset such as WiNER (Ghaddar and Langlais, 2017) for the supervised pretraining. Acknowledging the shortcomings and evaluation scheme disparity in Few-Shot NER, Ding et al. (2021) proposed a large scale dataset specifically designed for this task. Wang et al. (2021) explored model distillation for Few-Shot NER. However, this requires access to a large unlabelled dataset for good performance. Very recently, prompt based techniques have also surfaced in this domain (Cui et al., 2021). However, the performance of these methods rely heavily on the chosen prompt. As denoted by the author, the performance delta can be massive (upto 19% absolute F1 points) depending on the prompt. Thus, in the absence of a large validation set, their applicability becomes limited in true few-shot learning (Perez et al., 2021).

7 Conclusion

In this work, we propose a contrastive learning based framework CONTAINER that models Gaussian embedding and optimizes inter token distribution distance. This generalized objective helps us model a class agnostic feature extractor that avoids the pitfalls of prior Few-Shot NER methods. Additionally, CONTAINER can take advantage of few-sample support data to adapt the model representations to new tag domains. Extensive evaluations in multiple traditional and recent few-shot NER datasets reveal that, CONTAINER does not only outperform prior SOTAs, but it also performs consistently even in challenging scenarios. While we investigate the efficacy of distribution optimization based contrastive learning in Few-Shot NER, it will be of particular interest to investigate its potency in other domains as well.

8 Acknowledgment

We thank Nan Zhang and Ranran Haoran Zhang for their insightful feedback on the draft.

References

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *ICLR*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart. *arXiv preprint arXiv:2106.01760*.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Carl Doersch and Andrew Zisserman. 2017. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060.

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. 2014. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27:766–774.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 993–1000.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. *arXiv preprint arXiv:1902.10482*.

Abbas Ghaddar and Philippe Langlais. 2017. Winer: A wikipedia annotated corpus for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 413–422.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. *arXiv preprint arXiv:1810.10147*.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. *arXiv preprint arXiv:2006.05702*.

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*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. 2011. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Ishan Misra and Laurens van der Maaten. 2020. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Amber Stubbs and Özlem Uzuner. 2015. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/uthealth corpus. *Journal of biomedical informatics*, 58:S20–S29.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *NeurIPS*.

Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. 2019. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*.

Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2021. Meta self-training for few-shot neural sequence labeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1737–1747.

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*.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. *arXiv preprint arXiv:1906.04225*.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. *arXiv preprint arXiv:2010.02405*.

Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. 2019. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*.

Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. 2019. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6002–6012.