Learning from Miscellaneous Other-Class Words for Few-shot Named Entity Recognition

Meihan Tong[1], **Shuai Wang**[2], **Bin Xu**[1]*, **Yixin Cao**[3], **Minghui Liu**1, **Lei Hou**[1], **Juanzi Li**[1]
[1]Knowledge Engineering Laboratory, Tsinghua University, Beijing, China
[2]SLP Group, AI Technology Department, JOYY Inc, China
[3]S-Lab Nanyang Technological University, Singapore
tongmeihan@gmail.com, wangshuai1@yy.com
xubin@tsinghua.edu.cn, caoyixin2011@gmail.com
liu-mh16@mails.tsinghua.edu.cn,greener2009@gmail.com
lijuanzi@tsinghua.edu.cn

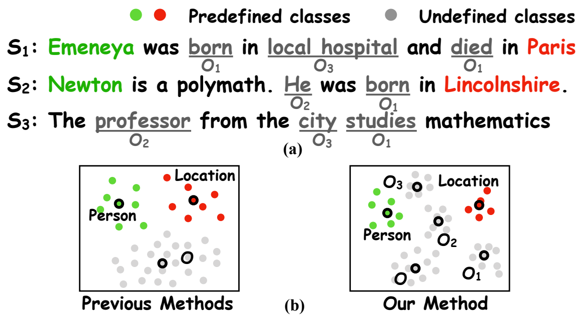
Abstract

Few-shot Named Entity Recognition (NER) exploits only a handful of annotations to identify and classify named entity mentions. Prototypical network shows superior performance on few-shot NER. However, existing prototypical methods fail to differentiate rich semantics in other-class words, which will aggravate overfitting under few shot scenario. To address the issue, we propose a novel model, **M**ining **U**ndefined **C**lasses from **O**ther-class (MUCO), that can automatically induce different undefined classes from the other class to improve few-shot NER. With these extra-labeled undefined classes, our method will improve the discriminative ability of NER classifier and enhance the understanding of predefined classes with stand-by semantic knowledge. Experimental results demonstrate that our model outperforms five state-of-the-art models in both 1-shot and 5-shots settings on four NER benchmarks. We will release the code upon acceptance. The source code is released on https://github.com/shuaiwa16/OtherClassNER.git.

1 Introduction

Named Entity Recognition (NER) seeks to locate and classify named entities from sentences into predefined classes (Yadav and Bethard, 2019). Humans can immediately recognize new entity types given just one or a few examples(Lake et al., 2015). Although neural NER networks have achieved superior performance when provided large-scale of training examples (Li et al., 2019), it remains a non-trivial task to learn from limited new samples, also known as few-shot NER (Fritzler et al., 2019).

Traditional NER models, such as LSTM+CRF (Lample et al., 2016), fail in few-shot settings. They calculate the transition probability matrix based on statistics, which requires a large number of data for optimization. Recently, prototypical network (Snell et al., 2017) shows potential on few-shot NER. The basic idea is to learn prototypes for each predefined entity class and an *other* class, then classify examples based on which prototypes they are closest to (Fritzler et al., 2019). Most existing studies focus on the predefined classes and leverage the label semantic to reveal their dependency for enhancement (Hou et al., 2020). However, they ignore the massive semantics hidden in the words of other class (O-class for short).



S₁: Emeneya was $\underset{O_1}{\text{born}}$ in $\underset{O_3}{\text{local hospital}}$ and $\underset{O_1}{\text{died}}$ in Paris

S₂: Newton is a polymath. $\underset{O_2}{\text{He}}$ was $\underset{O_1}{\text{born}}$ in Lincolnshire.

S₃: The $\underset{O_2}{\text{professor}}$ from the $\underset{O_3}{\text{city}}$ $\underset{O_1}{\text{studies}}$ mathematics

Figure 1: (a): Examples for undefined classes. (b): Different ways to handle O class (single prototype vs. multiple prototypes).

In this paper, we propose to learn from O-class words, rather than using predefined entity classes only, to improve few-shot NER. In fact, O-class contains rich semantics and can provide stand-by knowledge for named entity identification and disambiguation. As shown in Figure 1(a), if we can detect an undefined class consisting of references to named entities (such as pronouns), then due to their interchangeability (Katz and Fodor, 1963), we will obtain prior knowledge for named entity identification. For example, *Newton* can be replaced with *he* or *professor* in S_2 and S_3. If we can detect additional classes, including *he* and *professor*, we will have more evidence about where *Newton* may appear. In addition, if we can detect an undefined class that composed of *Action* (O_1), we may cap-

*Corresponding author.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 6236–6247
August 1–6, 2021. ©2021 Association for Computational Linguistics

ture underlined relations between different named entities, which is important evidence when distinguishing the named entity type (Ghosh et al., 2016; Zheng et al., 2017).

Nevertheless, it is challenging to detect related undefined classes from O class words due to two reasons: 1) Miscellaneous Semantics. O-class contains miscellaneous types of words. Based on our observations, although there are massive related yet undefined classes, the noise maybe even more, such as function and stop words. These noisy classes have little or negative impacts on the identification of target entities. Therefore, how to distinguish noise from task-related classes is a key point. 2) Lack of Golden Label. We neither have the labeled examples nor the metadata of each undefined class. The zero-shot methods (Pushp and Srivastava, 2017) fail in this case, since they need metadata (such as class name and class description) as known information. Unsupervised clustering methods also cannot meet quality requirements as shown in our experiment.

To handle the issues, we propose the **M**ining **U**ndefined **C**lasses from **O**ther-class (MUCO) model to leverage the rich semantics to improve few-shot NER. Instead of a single prototype, we learn multiple prototypes to represent miscellaneous semantics of O-class. Figure 1(b) shows the difference between our method and previous methods. To distinguish task-related undefined classes without annotations, we leverage weakly supervised signals from predefined classes and propose a zero-shot classification method called Zero-shot Miner. The main idea is inspired by transfer learning in prototypical network. Prototypical network can be quickly adapted to new class B when pre-training on related base class A. The underlined reason is that if two classes (A and B) are task-related, when we make examples in A class to cluster in the space, the examples in B class also tend to cluster in the space, even without explicit supervision on class B (Koch et al., 2015). Based on this phenomenon, we first perform prototype learning on predefined classes to cluster words in predefined classes, and then regard words in O-class that also tend to cluster as the undefined classes. Specifically, we train a binary classification to judge whether clustering occurs between any two of the words. After that, we label the found undefined classes back into sentences to jointly recognize predefined and undefined classes for knowledge transfer. Our contributions can be summarized as follows:

- We propose a novel approach MUCO to leverage rich semantics in O class to improve few-shot NER. To the best of our knowledge, this is the first work exploring O-class in this task.
- We propose a novel zero-shot classification method for undefined class detection. In the absence of labeled examples and metadata, our proposed zero-shot method creatively use the weakly supervised signal of the predefined classes to find undefined classes.
- We conduct extensive experiments on four benchmarks as compared with five state-of-the-art baselines. The results under both 1-shot and 5-shots settings demonstrate the effectiveness of MUCO. Further studies show that our method can also be conveniently adapted to other domains.

2 Related Work

Few-shot NER aims to recognize new categories with just a handful of examples (Feng et al., 2018; Cao et al., 2019). Four groups of methods are adopted to handle the low-resource issue: knowledge enhanced, cross-lingual enhanced, cross-domain enhanced, and active learning. Knowledge-enhanced methods exploit ontology, knowledge bases or heuristics labeling (Fries et al., 2017; Tsai and Salakhutdinov, 2017; Ma et al., 2016) as side information to improve NER performance in limited data settings, which suffer from knowledge low-coverage issue. Cross-lingual (Feng et al., 2018; Rahimi et al., 2019) and cross-domain enhanced methods (Wang et al., 2018; Zhou et al., 2019) respectively use labeled data from a counterpart language or a different domain as external supervised signals to avoid overfitting. When the language or domain discrepancy is large, these two methods will inevitably face the problem of performance degradation (Huang et al., 2017). Active learning methods (Wei et al., 2019) explicitly expand corpus by selecting the most informative examples for manual annotation, which need extra human-laboring. Different from previous methods, we focus on mining the rich semantics in the O class to improve few-shot NER.

2.1 Prototypical Network

Prototypical network (Snell et al., 2017), initially proposed for image classification, has been successfully applied to sentence-level classification tasks, such as text classification (Sun et al., 2019) and relation extraction (Gao et al., 2019). However, there is a dilemma to adapt prototypical network for token-level classification tasks such as NER. Prototypical network assumes that each class has uniform semantic and vectors belong to the same class should cluster in the space. However, in NER, data in O class contain multiple semantics and thus violate the uniform semantic hypothesis in prototypical network. To handle the issue, Deng et al. (2020) first trains a binary classifier to distinguish O class from other predefined classes, and then adopt traditional prototypical network methods, which suffers from pipeline error propagation. Fritzler et al. (2019) does not calculate the prototype of O class from data, but directly sets a hyper-parameter b_o as the fake distance similarity and optimize b_o during training, which still regards O class as a whole. On the contrary, we are the first to divide O class into multiple undefined classes and explicitly learn multiple spatially-dispersed prototypes for O class.

3 Methodology

Figure 2 illustrated the architecture of the proposed MUCO model. MUCO is composed of two main modules: **Undefined Classes Detection** detects multiple undefined classes hidden in O class to fully exploit the rich semantics in O class. **Joint Classification** jointly classifies the undefined classes and predefined classes, so as to leverage the stand-by semantic knowledge in undefined classes to enhance the understanding of predefined classes.

3.1 Notation

In few-shot NER, we are given training examples $D = D_c \cup D_o$, where $D_c = \{x_i, y_i|_{i=1}^{N}\}$ is the training examples of predefined classes $C = \{c_1, c_2, \ldots, c_k\}$ and $D_o = \{x_i|_{i=1}^{M}\}$ is the training examples of O class. For each example (x, y), x is composed by S and w_j, where $S =< w_1, w_2, \ldots, w_n >$ stands for the sentence and w_j is the queried named entity, y is the class label of the queried named entity w_j. We denote the prototype of class y as p_y and prototypes for all classes $C \cup O$ as $P = \{p_y|y \in C \cup O\}$. Formally, our goal is first to detect multiple undefined classes $O = \{o_1, o_2, \ldots, o_r\}$ to label the examples in D_o, and then maximize the prediction probability $P(y|x)$ on D_c and D_o.

3.2 Undefined Classes Detection

In few-shot NER, most of the words in the sentence belong to O class. Different from predefined classes, O class means none-of-the-above, and contains multiple undefined entity types. Previous methods ignore the fine-grained semantic information in O class and simply regard O as a normal class. We argue to further decouple O class into multiple undefined classes to fully exploit the rich semantics hidden in O class.

In the section, we aim to detect undefined classes from O class. It is a non-trivial task since we lack metadata and golden labels to help us distinguish undefined classes. What is worse, the examples from O class is numerous and the search space is large. To handle the issue, we propose a zero-shot classification method called Zero-shot Miner to leverage the weak supervision from predefined classes for undefined classes detection. Our method inspires by transfer learning, we argue that if an undefined class is task-related, when we push the examples in predefined classes to cluster in the space, the examples in the undefined class should also have the signs of gathering, even without explicit supervision (Koch et al., 2015). For instance, in Figure 2, if we guide *Emeneya* and *Newton* (the green points 1, 3) to cluster in the space, *professor* and *He* (the grey points 9, 12) will also tend to cluster in the space.

Based on this argument, undefined classes detection could be achieved by finding multiple groups of examples in O class that have a tendency to cluster during the training of the prototypical network on predefined classes. As shown in Figure 2, there are three steps in our zero-shot classification method. In **step 1**, we train the prototypical network on predefined classes to obtain the learned mapping function. Through the learned mapping function the examples belonging to the same class will cluster in the space. In **step 2**, we train a binary group classifier on predefined classes base on the position features from the learned mapping function and unlearned mapping function to judge whether any two points tend to cluster during the step 1 training. In **step 3**, we use the learned binary group classifier in step 2 to infer examples in O class to distinguish undefined classes from each

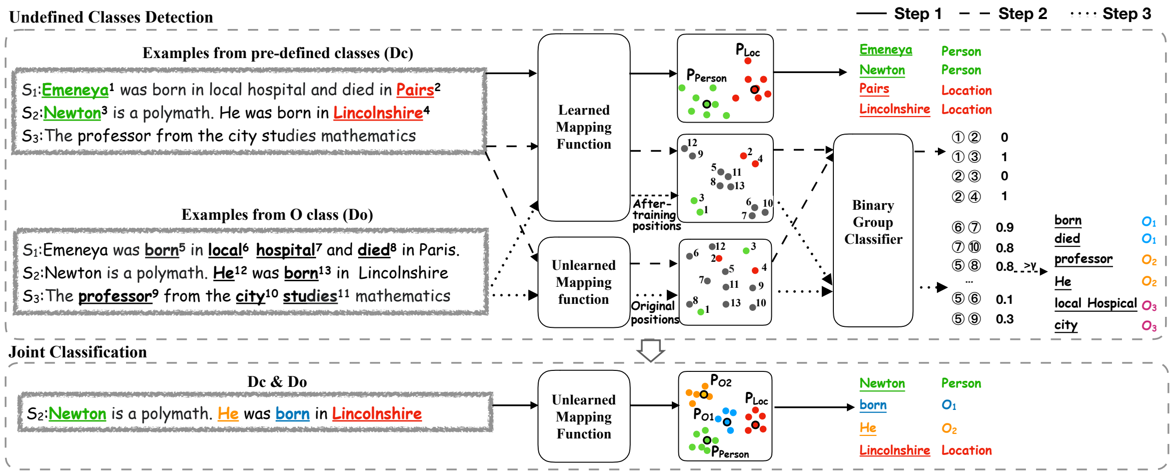

Undefined Classes Detection
Step 1
Step 2
Step 3
Examples from pre-defined classes (Dc)
S1:Emeneya1 was born in local hospital and died in Pairs2
S2:Newton3 is a polymath. He was born in Lincolnshire4
S3:The professor from the city studies mathematics
Examples from O class (Do)
S1:Emeneya was born5 in local6 hospital7 and died8 in Paris.
S2:Newton is a polymath. He12 was born13 in Lincolnshire
S3:The professor9 from the city10 studies11 mathematics
Learned Mapping Function
Unlearned Mapping function
After-training positions
Original positions
PLoc
PPerson
Binary Group Classifier
Emeneya Person
Newton Person
Pairs Location
Lincolnshire Location
① ② 0
① ③ 1
② ③ 0
② ④ 1
⑥ ⑦ 0.9
⑦ ⑩ 0.8
⑤ ⑧ 0.8
⑤ ⑥ 0.1
⑤ ⑨ 0.3
>γ
born O1
died O1
professor O2
He O2
local Hospical O3
city O3
Joint Classification
Dc & Do
S2:Newton is a polymath. He was born in Lincolnshire
Unlearned Mapping Function
PO2
PO1
PLoc
PPerson
Newton Person
born O1
He O2
Lincolnshire Location

Figure 2: The architecture of the proposed MUCO model. We first detect undefined classes from O class, and then jointly classify the predefined classes and the found undefined classes for knowledge transfer. Specifically, in undefined classes detection, we propose a zero-shot classification method, which includes three steps. In step 1, we learn a mapping function through prototypical network training on predefined classes. In step 2, we learn a binary group classifier to judge whether any two points in predefined classes tend to cluster during the step 1 training. In step 3, we use the binary group classifier to infer pairs of examples in O class to distinguish multiple undefined classes.

other. The following articles will illustrate the three steps sequentially.

3.2.1 Step 1: Mapping Function Learning

In prototypical network, mapping function $f_\theta(x)$ aims to map the example x to a hidden representation. BERT is adopted as the mapping function in our model, which is a pre-trained language representation model that employs multi-head attention as the basic unit, and have superior representation ability (Geng et al., 2019).

We train the mapping function by correctly distinguishing the predefined classes. First, we extract the feature of the queried word. Formally, given the training example $(x, y) \in D_c$, where x is composed of sentence $S =< w_1, w_2, \ldots, w_n >$ and the queried word w_j, we extract the j-th representation of the sequence output of the last layer of BERT as the hidden representation.

$$h = f_\theta(x) \tag{1}$$

Then, following (Qi et al., 2018), we randomly initialize the prototype p_y of class y at the beginning of training, and then we shorten the distance between examples in class y to prototype p_y during training. Compared to traditional prototypical learning (Snell et al., 2017), we do not need to waste part of the examples for prototype calculation.

$$d(x, p_y) = -f_\theta(x)^T p_y \tag{2}$$

where $f_\theta(x)$ and p_y are first normalized by L2 normalization.

The final optimization goal for training the mapping function is

$$L(\theta_1) = -log \frac{exp(-d(x, p_y))}{\sum_{p_c \in P_c} exp(-d(x, p_c))} \tag{3}$$

where $P_c = \{p_c | c \in C\}$ stands for the prototypes of all the predefined classes.

3.2.2 Step 2: Binary Group Classifier Training

Recall that to detect multiple undefined classes, we need to find multiple example groups, and the examples in each group should have a tendency to cluster.

To handle the issue, we learn a binary group classifier on predefined classes. The main idea is that if we can determine whether any two examples belong to the same group, we can distinguish groups from each other. Formally, given a pair of examples (x_i, y_i) and (x_j, y_j) in D_c, their original position h_i, h_j from unlearned mapping function $f_\theta(x)$, and after-training position $\tilde{h}_i$, $\tilde{h}_j$ from learned mapping function $\tilde{f}_\theta(x)$, the probability of x_i and x_j belonging to the same class is defined as follows:

$$\begin{aligned} b_{ij} = W([h_i; h_j; \tilde{h}_i; \tilde{h}_j; |h_i - h_j|; \\ |\tilde{h}_i - \tilde{h}_j|; |h_i - \tilde{h}_i|; |h_j - \tilde{h}_j|]) + b \end{aligned} \tag{4}$$

By comparing the distance variation between original positions h and the after-training positions $\tilde{h}$, we can tell whether aggregation occurs between any of the two points.

The optimization goal of the binary group classifier is

$$L(\theta_2) = \frac{1}{N^2}\sum_{i}^{N}\sum_{j}^{N}(-y_{ij} * log(b_{ij}) + (1-y_{ij}) * log(1-b_{ij})) \tag{5}$$

where N is the numbers of the examples in predefined classes, and y_{ij} is the label. If x_i and x_j are from the same predefined class (y_i=y_j), y_{ij} is 1, otherwise 0.

3.2.3 Step 3: Binary Group Classifier Inference

After training, we feed each pair of examples x_u and x_v in D_o to the binary group classifier to obtain the group dividing results. The output b_{uv} indicates the confidence that x_u and x_v belong to the same group. We set a threshold to divide the group. If b_{uv} is larger than the threshold γ, x_u and x_v shall belong to the same group (undefined class). If consecutive words belong to the same group, we will treat these words as one multi-word entity. Noted that some of the examples in O class may not belong to any group. We assume that these examples come from the task-irrelevant classes, and no further classification is made for these examples.

Soft Labeling After the process of group dividing, we obtain labels of multiple undefined classes $O = \{o_1, o_2, \ldots, o_r\}$. We further adopt the soft labeling mechanism. For each undefined class o_i, we calculate the mean of the examples as the class center, then we apply softmax on the cosine similarity between examples and its class center as the soft labels. Through soft labeling, we can consider how likely examples belong to the undefined classes.

3.3 Joint Classification

In the section, we take into consideration of both the predefined classes C and the found undefined classes O for joint classification. First, we label the examples in undefined classes back into the sentences, as shown in *Joint Classification* of Figure 2. Then, we optimize the examples to make them closer to the corresponding prototype for better discrimination. Comparing to the Equation 3, we add the prototypes from O class $P_o = \{p_{o_1}, p_{o_2}, \ldots, p_{o_r}\}$ as candidate prototypes.

Formally, given the examples $(x, y) \in D_c \cup D_o$, the corresponding prototype p_y and prototypes set $P = P_c \cup P_o$ from both predefined classes C and undefined classes O, the optimization object is defined as:

$$L(\theta_3) = -log\frac{exp(-d(x, p_y))}{\sum_{p\in\{P_c\cup P_o\}} exp(-d(x, p))} \tag{6}$$

Scale Factor When calculating $d(x, p_y)$, the $f_\theta(x)$ and p_y have been normalized and the value is limited to [-1, 1]. When softmax activation is applied, the output is unable to approach the one-hot encoding and therefore imposes a lower bound on the cross-entropy loss (Qi et al., 2018). For instance, even we give the golden prediction: giving 1 for correct category and -1 for the wrong ones, the probability of output $p(y|x) = e^1/[e^1 + (|C \cup T| - 1)e^{-1}]$ is still unable to reach 1. The problem becomes more severe as we increase the number of named entity categories by introducing more categories for O class. To alleviate the issue, we modify Eq. 6 by adding a trainable scalar s shared across all classes to scale the inner product (Wang et al., 2017).

$$L(\theta_3) = -log\frac{exp(-sd(x, p_y))}{\sum_{p\in\{P_c\cup P_t\}} exp(-sd(x, p))} \tag{7}$$

3.4 Implementation Details

Following traditional prototypical network (Snell et al., 2017), we pre-train the model on several base classes, whose types are disjoint to few-shot classes and have abundant labeled corpus. The underlined idea is to leverage existing fully annotated classes to improve the performance of the model on new classes with only a few annotations. All predefined classes (both base classes and few-shot classes) are used when searching for undefined classes, so that the annotations of undefined classes can be shared between pre-training and fine-tuning, which will improve the transfer performance of our model.

4 Experiment

4.1 Datasets

We conduct experiments on multiple datasets to reduce the dataset bias, including three English benchmarks Conll2003 (Sang and De Meulder, 2003), re3d (Science and Laborator, 2017) and

Ontonote5.0 (Pradhan et al., 2013) and one Chinese benchmark CLUENER2020 (Xu et al., 2020). Conll2003 contains 20,679 labeled sentences, distributed in 4 classes in the News domains. The data in re3d comes from defense and security domain, with 10 classes and 962 labeled sentences. Ontonotes5.0 has 17 classes with 159,615 labeled sentences in mixed domains - News, BN, BC, Web and Tele. CLUENER2020 has 10 fined grained entity types with 12,091 annotated sentences. For all of the datasets, we adopt BIO (**B**eginning, **I**nside, and **O**utside) labeling, which introduces an extra O class for non-entity words.

4.2 Data Split

We divided the classes of each benchmark into two parts: base classes and few-shot classes. The few-shot classes for Conll / re3d / Ontonote / CLUENER are Person / Person, Nationality, Weapon / Person, Language, Money, Percent, Norp / Game, Government, Name, Scene. The rest are the base classes. The division is based on the average word similarity among classes (mean similarity is reported in Appendix A). At each time, the class with the largest semantic difference from other classes is selected and added to the few-shot classes until the number of few-shot classes reaches 1/3 of the base classes. In this way, we can prevent the few-shot classes and base classes from being too similar, leading to information leakage. We do not follow previous methods (Hou et al., 2020) to adopt different datasets as base and few-shot classes, because there are overlapped classes in such data split, such as *Person*, which will reduce the difficulty of few-shot setting. For base classes, all examples are used to train the base classifier. For few-shot classes, only K examples are used for training, and the rest are used for testing. Alternatively, we adopt the N-way K-shot setting for few-shot classes, where N is the number of few-shot classes and K is the number of examples sampled from each few-shot class. K is set to 1 and 5 respectively in our experiment. Noted that we can not guarantee the number of the examples is exactly equal to K when sampling, because there will be multiple class labels in one sentence. Following (Fritzler et al., 2019), we ensure there are at least K labels for each few-shot class.

4.3 Evaluation Metrics

Following (Hou et al., 2020), we measure the precision, recall, and macro-averaged F1 scores on all few-shot classes. For fair comparison with baselines, as long as the found undefined class is classified as O class, it can be considered correct. We report the average on ten runs as the final results.

4.4 Hyperparameters

For feature extraction, we adopt BERT-base as our backbone [1], which has 12-head attention layers and 768 hidden embedding dimension. For learning rate, we adopt greedy search in the range of 1e-6 to 2e-4. We set learning rage to 2e-5 when pre-training on base classes and 5e-6 when fine-tuning on few-shot classes. The threshold γ is set to 0.68 to ensure that the found undefined classes are sufficiently relevant to the predefined classes. The batch size is 128 and the maximum sequence length 128. We set the scale factor in Eq. 7 to 10 at the beginning. Our code is implemented by Tensorflow and all models can be fit into a single V100 GPU with 32G memory. The training procedure lasts for about a few hours. The best result appears around the 100 epochs of the training process.

4.5 Baselines

We divide the baselines into two categories: 1) Supervised-Only Methods. **BERT** uses pre-trained BERT model to sequentially label words in sentence (Devlin et al., 2018). **Prototypical network (PN)** learns a metric space for each class (Snell et al., 2017). Both of the methods are only trained on the few-shot classes. 2) Few-shot Methods. **L-TapNet+CDT (LTC)** uses semantic associations between base and few-shot classes to improve the prototype quality, which is only trained on base classes (Hou et al., 2020). We use the original published code [2]. **Warm Prototypical Network (WPN)** (Fritzler et al., 2019) is the transfer learning version of PN, which is first pre-trained on base classes and then fine-tuned on few-shot classes. **MAML** first learns fast-adapted parameters on base classes and then fine-tune the parameters on few-shot classes (Finn et al., 2017).

4.6 Overall Performance

Table 1 and 2 present the overall performance of the proposed approach on four NER benchmarks - Conll2003, re3d, Ontonote5.0 and CLUENER2020. MUCO (ours) consistently outperforms state-of-the-art models, showing the effectiveness of ex-

[1]https://github.com/google-research/bert

[2]https://github.com/AtmaHou/FewShotTagging

Table 1: Overall Performance on Conll2003, re3d, Ontonote5.0 and ClUENER2020 dataset in 1-shot setting(%).

Methods	1-shot Named Entity Recognition											
	Conll2003			re3d			Ontonote5.0			CLUENER2020		
	P	R	F	P	R	F	P	R	F	P	R	F
BERT	61.00	50.46	54.28	31.49	22.56	26.13	54.92	32.09	39.92	26.95	18.68	21.77
PN	55.78	50.72	52.10	32.07	23.09	26.75	55.77	30.56	38.67	27.64	19.78	22.81
LTC	78.19	70.36	73.31	29.84	19.33	23.34	60.83	43.25	50.04	-	-	-
WPN	77.87	86.58	81.40	43.12	38.90	40.27	58.29	54.39	56.20	76.63	70.96	73.50
MAML	75.95	85.69	79.80	43.95	34.77	37.83	56.63	55.84	56.15	77.71	69.53	73.08
MUCO (ours)	**81.70**	83.98	**82.69**	43.23	**40.37**	**41.57**	60.43	55.82	**57.89**	**78.29**	**73.60**	**75.80**

Table 2: Overall Performance on Conll2003, re3d, Ontonote5.0 and ClUENER2020 dataset in 5-shot setting(%).

Methods	5-shots Named Entity Recognition											
	Conll2003			re3d			Ontonote5.0			CLUENER2020		
	P	R	F	P	R	F	P	R	F	P	R	F
BERT	73.94	68.28	70.54	32.43	25.05	28.09	61.81	56.64	59.04	71.5	68.14	69.61
PN	74.36	71.46	72.70	31.26	25.37	27.77	61.84	58.61	60.12	71.60	68.56	69.92
LTC	85.89	82.41	83.97	40.98	32.00	35.83	62.06	46.08	52.35	-	-	-
WPN	94.39	95.00	94.68	40.93	38.63	39.68	65.28	67.66	66.34	80.52	79.71	80.04
MAML	94.76	96.04	95.37	41.78	39.49	40.52	65.99	69.31	67.57	77.06	82.83	79.78
MUCO (ours)	**96.23**	95.35	**95.78**	**43.04**	**41.70**	**42.37**	**73.27**	69.00	**71.06**	78.88	82.67	**80.64**

ploiting the rich semantics in O class and the superiority of the proposed MUCO model.

Compared with supervised-only methods (BERT and PN), few-shot methods (TransferBERT, WPN, MAML, L-TapNet+CDT and MUCO(ours)) achieve better performance. By first training on base classes, these methods will learn a prior, which prevents from overfitting densely labeled words. Among few-shot methods, our model achieves the best performance. Previous methods regard O class as a single class. On the contrary, we induce different undefined classes from O class, and add more task-related classes for joint training, which directly handles the dilemma of scarcity of data in few-shot learning and provides stand-by semantics to identify and disambiguate named entity, thereby improving the performance of few-shot NER. No matter English corpus (the first three) or Chinese corpus (the last one), our methods consistently improves the F score, showing the language-independent superiority of our method. Task-agnostic superiority also shows in section 4.10. Our undefined classes detection method is completely data-driven. The found undefined classes will be automatically adjusted to be useful and task-related based on current language or task predefined classes.

To further evaluate our core module *undefined classes detection* in section 3.2, we introduce a *Word-Similarity* (WS) baseline. WS detects undefined classes by performing KMeans (Kanungo et al., 2002) in O words based on word similarity. To be fair, WS, like our method, uses soft-label enhancement (section 3.2.2). We report the final few-shot NER performance on Ontonote for comparison.

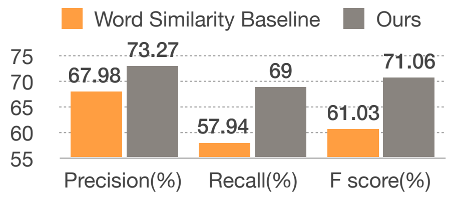


Figure 3: Few-shot NER Performance under Different Undefined Classes Detection Algorithm

As shown in Figure 3, our method achieves better performance, which shows the superior of our undefined classes detection module. *Word similarity* baseline only uses semantics of words and lacks weak supervision from predefined classes, so that noisy classes (such as punctuation) cannot be distinguished from task-related ones, which inevitably reduces the quality of undefined classes.

4.7 Quality of Found Undefined Classes

In the section, we evaluate the quality of the found undefined classes from quantitative and qualitative perspective. All the following experiments are conducted on Ontonote5.0.

For quantitative analysis, we invite three computer engineers to manually label 100 sentences for human evaluation. The metrics are Intra-class Correlation (IC) and Inter-class Distinction (ID). The IC statistics how many labels actually belong to the declared class. The ID counts how many labels belong to only one of the undefined classes, not to multiple classes. We obtain golden labels by applying the majority vote rule. Table 3 reports the average results on undefined classes.

Table 3: Human Evaluation

Metrics	IC	ID
Average Score(%)	49.15	50.85

Considering the zero-shot setting, the accuracy of 49.15% and 50.85% is high enough, which indicates that the found undefined classes basically have semantic consistency within the classes and semantic difference between classes.

For qualitative analysis, we illustrate a case study in Table 4. The words in O_1, O_2 and O_3 are mainly the general entity versions of Person, Location and Numerous respectively. According to the grammatical rules, general entities and named entities can be substituted for each other, *Lincoln* can also be called *president*, so identifying general entities can provide additional location knowledge and enhance named entity identification. The words in O_4 and O_5 are mainly *Action*, which may imply relations between different named entities and provide important evidence for named entity disambiguation (Tong et al., 2020). The errors mainly come from three aspects: 1) The surrounding words are incorrectly included, such as *from* in *businessmen from* in O_1; 2) Some strange words reduce intra-class consistency, such as *was at the tail* in O_3; 3) There is semantic overlap between classes, such as O_4 and O_5. Future work will explore how to improve the quality of the undefined classes.

4.8 Different Number of Undefined Classes

Since our model needs to manually set the number of undefined classes, we observe the performance of the model under different number settings. We set the number of undefined classes to 1/2/5/10/25/50 by adjusting the threshold γ.

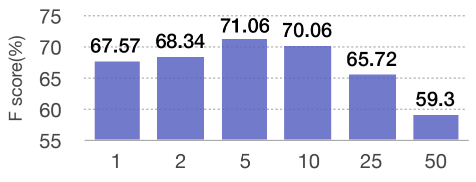


Figure 4: Different Numbers of Undefined Classes

Figure 4 illustrates the F score of MUCO (ours) on various numbers of undefined classes. It will impair the performance when the number is too large or too small. When the number is too large, the found classes will have overlapping problems, resulting in severe performance degradation (-11.51%). When the number is too small, the model is unable to find enough task-related classes, limiting the ability to capture the fine-grained semantics in O class. Empirical experiments found that when the number of undefined classes is approximately equal to the number of few-shot classes, our method achieves the best performance (the number is 5 in Figure 4). We argue that the number of predefined classes is proportional to the amount of information hidden in weak supervision. Therefore, with more predefined classes, we can also find more high-quality undefined classes.

4.9 Cross-Domain Ability

In this section, we answer whether our model could achieve superior performance facing the discrepancy of different domains. To simulate a domain adaption scenario, we choose the benchmark Conll2003 (Sang and De Meulder, 2003) as the source domain and AnEM (Ohta et al., 2012) as the target domain. The entity types in AnEM, such as *Pathological-Formation*, are all medical academic terms and can ensure the discrepancy to common classes in Conll2003.

Table 5: Domain Adaption Ability.

Method	P	R	F
PN	7.34	17.14	7.43
WPN	33.06	31.95	26.90
MUCO (Ours)	**34.17**	**32.84**	**28.31**

As illustrated in Table 5, our method achieves the best adaptation performance on the target domain. All the predefined classes, both in source domains and target domains, are used when detection undefined classes. The annotations of undefined classes can be shared between pre-training and fine-tuning, which will improve the transfer performance of our model.

4.10 Task-Agnostic Ability

In this section, we answer whether our assumption of O class is task-agnostic and effective for few-shot token-level classification tasks other than NER. We conduct experiments on two tasks of widespread concern: Slot Tagging (Hou et al., 2020) and Event Argument Extraction (Ahn, 2006). Slot Tagging aims to discover user intent from task-

Table 4: Case Study of the Found undefined Classes

	Annotated Words
O_1	gentleman; journalist; president; ambassador; I; he; they; businessmen from; and those Huwei people who;
O_2	the harbour; this land, which; over the river; with the great outdoors; outsides; to nature; the skyline;
O_3	some; a major; the small number; supplied; not only one of the; empty; large; increase of; was at the tail;
O_4	believe; comfort; attacked or threatened; arrest; geared; talks; not dealing; discussions; agreement;
O_5	stop; have; do; discussion; take; seek; sat down; negotiated; think; failed; replace;

oriented dialogue system. We adopt Snips dataset (Coucke et al., 2018) for Slot Tagging, and the split of train/test is We,Mu,Pl,Bo,Se/Re,Cr. Event Argument Extraction aims to extract the main elements of event from sentences. We adopt the ACE2005 dataset [3] with 33 classes and 6 domains. The train/test is bc,bn,cts,nw/un,wl.

Table 6: Task-Agnostic Effectiveness of Silent Majority(ours)

Methods	ST		
	P	R	F
PN	61.29	58.24	59.02
WPN	73.60	73.29	70.56
Silent Majority (Ours)	**75.92**	**73.71**	**72.04**

Methods	EAE		
	P	R	F
PN	51.02	53.14	51.85
WPN	78.39	70.59	73.13
Silent Majority (Ours)	**79.61**	**72.02**	**75.20**

As illustrated in Table 6, the proposed model achieves superior performance on both tasks, which demonstrates the generalization ability of our method. No matter what task the predefined class belongs to, our method is always able to mine the task-related classes from the O class to help eliminate the ambiguity of the predefined class. The reason is that our detection method is entirely data-driven, and does not rely on manually writing undefined class descriptions. The found category will automatically change according to the task type of the entered predefined classes. Therefore, the migration cost between tasks of our method is meager.

5 Conclusion

In this paper, we propose **M**ining **U**ndefined **C**lasses from **O**ther-class (MUCO) to utilize the rich semantics in O class to improve few-shot NER. Specifically, we first leverage weakly supervised signals from predefined classes to detect undefined classes from O classes. Then, we perform joint classification to exploit the stand-by semantic knowledge in undefined classes to enhance the understanding of few-shot classes. Experiments show that our method outperforms five state-of-the-art baselines on four benchmarks.

[3]http://projects.ldc.upenn.edu/ace/

Acknowledgements

This work is supported by the National Key Research and Development Program of China (2018YFB1005100 and 2018YFB1005101) and NSFC Key Project (U1736204). This work is supported by National Engineering Laboratory for Cyberlearning and Intelligent Technology, Beijing Key Lab of Networked Multimedia and the Institute for Guo Qiang, Tsinghua University (2019GQB0003). This research was conducted in collaboration with SenseTime. This work is partially supported by A*STAR through the Industry Alignment Fund - Industry Collaboration Projects Grant, by NTU (NTU–ACE2020-01) and Ministry of Education (RG96/20).

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

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

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2020. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 151–159.

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*.

Xiaocheng Feng, Xiachong Feng, Bing Qin, Zhangyin Feng, and Ting Liu. 2018. Improving low resource named entity recognition using cross-lingual knowledge transfer. In *IJCAI*, pages 4071–4077.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *arXiv preprint arXiv:1704.06360*.

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.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6407–6414.

Ruiying Geng, Binhua Li, Yongbin Li, Yuxiao Ye, Ping Jian, and Jian Sun. 2019. Few-shot text classification with induction network. *arXiv preprint arXiv:1902.10482*.

Souvick Ghosh, Promita Maitra, and Dipankar Das. 2016. Feature based approach to named entity recognition and linking for tweets.

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*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. 2017. Zero-shot transfer learning for event extraction. *arXiv preprint arXiv:1707.01066*.

Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892.

Jerrold J Katz and Jerry A Fodor. 1963. The structure of a semantic theory. *language*, 39(2):170–210.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

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.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2019. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855*.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 171–180.

Tomoko Ohta, Sampo Pyysalo, Jun'ichi Tsujii, and Sophia Ananiadou. 2012. Open-domain anatomical entity mention detection. In *Proceedings of the workshop on detecting structure in scholarly discourse*, pages 27–36.

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.

Pushpankar Kumar Pushp and Muktabh Mayank Srivastava. 2017. Train once, test anywhere: Zero-shot learning for text classification. *CoRR*, abs/1712.05972.

Hang Qi, Matthew Brown, and David G Lowe. 2018. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for ner. *arXiv preprint arXiv:1902.00193*.

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

Defence Science and Technology Laborator. 2017. Relationship and entity extraction evaluation dataset.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087.

Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019. Hierarchical attention prototypical networks for few-shot text classification. 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 476–485.

Meihan Tong, Bin Xu, Shuai Wang, Lei Hou, and Juaizi Li. 2020. Improving low-resource chinese event detection with multi-task learning. In *International Conference on Knowledge Science, Engineering and Management*, pages 421–433. Springer.

Yao-Hung Hubert Tsai and Ruslan Salakhutdinov. 2017. Improving one-shot learning through fusing side information. *arXiv preprint arXiv:1710.08347*.

Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049.

Zhenghui Wang, Yanru Qu, Liheng Chen, Jian Shen, Weinan Zhang, Shaodian Zhang, Yimei Gao, Gen Gu, Ken Chen, and Yong Yu. 2018. Label-aware double transfer learning for cross-specialty medical named entity recognition. *arXiv preprint arXiv:1804.09021*.

Qiang Wei, Yukun Chen, Mandana Salimi, Joshua C Denny, Qiaozhu Mei, Thomas A Lasko, Qingxia Chen, Stephen Wu, Amy Franklin, Trevor Cohen, et al. 2019. Cost-aware active learning for named entity recognition in clinical text. *Journal of the American Medical Informatics Association*, 26(11):1314–1322.

Liang Xu, Qianqian Dong, Yixuan Liao, Cong Yu, Yin Tian, Weitang Liu, Lu Li, Caiquan Liu, Xuanwei Zhang, et al. 2020. Cluener2020: Fine-grained named entity recognition dataset and benchmark for chinese. *arXiv*, pages arXiv–2001.

Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. *arXiv preprint arXiv:1706.05075*.

Joey Tianyi Zhou, Hao Zhang, Di Jin, Hongyuan Zhu, Meng Fang, Rick Siow Mong Goh, and Kenneth Kwok. 2019. Dual adversarial neural transfer for low-resource named entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3461–3471.

A Data Split

We divided the classes of each benchmark into two parts: base classes and few-shot classes. The division is based on the average word similarity among classes. At each time, the class with the largest semantic difference from other classes is selected and added to the few-shot classes until the number of few-shot classes reaches 1/3 of the base classes. In this way, we can prevent the few-shot classes and base classes from being too similar, causing information leakage. The embedding of words are extracted from BERT, and the mean similarity is reported in Table 7.

Table 7: Mean Word Similarity between Predefined Classes

Conll2003				
PER	MISC	LOC	ORG	
0.64	1.36	1.68	1.75	
re3d				
Nationality	Person	Weapon	Temporal	MilitaryPlatform
3.3	3.87	4.05	4.28	4.34
Quantity	Money	DocumentReference	Location	Organisation
4.37	4.38	5.03	5.2	5.74
Ontonotes5.0				
PERSON	MONEY	PERCENT	LANGUAGE	NORP
0.31	2.2	3.88	4.14	4.49
CARDINAL	PRODUCT	QUANTITY	ORG	LAW
4.58	5.16	5.41	5.74	5.99
TIME	ORDINAL	WORK_OF_ART	GPE	LOC
6.01	6.23	6.24	6.7	7.18
DATE	FAC	EVENT		
7.27	7.53	8.11		
ClUENER2020				
name	government	game	scene	position
7.28	7.38	7.43	7.45	7.6
address	movie	company	organization	book
7.61	7.62	7.65	7.72	7.91