Auxiliary Learning for Named Entity Recognition with Multiple Auxiliary Training Data

Taiki Watanabe[1,2], **Tomoya Ichikawa**[2], **Akihiro Tamura**[2]
Tomoya Iwakura[1], **Chunpeng Ma**[1], **Tsuneo Kato**[2]
[1] Fujitsu Ltd., [2] Doshisha University
{watanabe-taiki,iwakura.tomoya,ma.chunpeng}@fujitsu.com
ctwg0109@mail4.doshisha.ac.jp
{aktamura,tsukato}@mail.doshisha.ac.jp

Abstract

Named entity recognition (NER) is one of the core technologies for knowledge acquisition from text and has been used for knowledge extraction of chemicals and medicine. As one of the NER improvement approaches, multi-task learning that learns a model from multiple training data has been used. Among multi-task learning, an auxiliary learning method, which uses training data of an auxiliary task for improving its target task, has shown higher NER performance than conventional multi-task learning for improving all the tasks simultaneously. The conventional auxiliary learning method uses only one auxiliary training dataset. We propose *M*ultiple *U*tilization of *N*ER *C*orpora *H*elpful for *A*uxiliary *BLES*sing (**MUNCHABLES**). MUNCHABLES utilizes multiple training datasets as auxiliary training data by the following methods : the first one is to fine-tune the NER model of the target task by sequentially performing auxiliary learning for each auxiliary training dataset, and the other is to use all training datasets in one auxiliary learning. We evaluate MUNCHABLES on eight chemical/biomedical/scientific domain NER tasks, where seven training datasets are used as auxiliary training data. The experiment results show that our proposed methods achieve higher NER performance than conventional multi-task learning methods on average and that NER performance can be improved by using multiple auxiliary training data. Furthermore, the proposed models outperform state-of-the-art models on the datasets.

1 Introduction

Named entity recognition (NER) is a fundamental natural language processing technology for extracting named entity (NE) and technical terms from input texts and has been put to practical use in various situations. For example, NER is used as one of the core technologies for structuring and accumulating information on interrelationships among chemical substances and physical properties of chemical substances, which are reported daily in papers and patents, to develop new materials and products.

NER has been actively studied for a long time, and many NER methods have been proposed. In recent years, neural network (NN)-based methods have become dominant, and a BiLSTM-CRF model (e.g., Huang et al. (2015)), composed of two recurrent neural networks (RNNs) and conditional random fields (CRF), and a Transformer-based model (e.g., Lee et al. (2019)) have achieved high performance in NER.

In addition, it has been reported that the performance of an NER model is improved by multi-task learning, which uses training data of a task different from the target task and simultaneously learns features from multiple NER training datasets (Wang et al., 2019a; Crichton et al., 2017a; Khan et al., 2020; Mehmood et al., 2020; Wang et al., 2019b). Remarkably, Wang et al. (2019a) have shown that, an NER in the biotechnology field (BioNER) with an auxiliary learning method, which is a variant of multi-task learning, achieves higher performance in the target task, compared to a standard multi-task learning method. The auxiliary learning uses a task other than the target task as an auxiliary task for improving the target task performance, in contrast the standard multi-task learning learns models for multiple tasks to improve performance of the multiple tasks.

We propose a new auxiliary learning paradigm that uses multiple NER datasets as auxiliary training data, *M*ultiple *U*tilization of *N*ER *C*orpora *H*elpful for *A*uxiliary *BLES*sing (**MUNCHABLES**), whereas existing auxiliary learning uses only one type of auxiliary training data. Specifically, we propose two types of multi-auxiliary learning: the first one is to fine-tune the NER model of the target task by sequentially performing auxiliary learning for each auxiliary training dataset (MUNCHABLES-stack model), and the other is to use all types of training data in single auxiliary learning. As for the latter, we propose two models: one is to concatenate all the multiple auxiliary training datasets and make a batch by randomly selecting data from the auxiliary training dataset (MUNCHABLES-concatenation model), and the other is to change auxiliary training datasets every epoch (MUNCHABLES-iteration model).

We compare the proposed MUNCHABLES models with standard multi-task learning and single auxiliary learning on eight chemical/ biomedical/ scientific domain NER tasks. As for our proposed models, seven training datasets are used as auxiliary training data in each task. The experiment results show that the F1-scores of the proposed models are higher than those of the baselines on average and NER performance can be improved by using multiple auxiliary training datasets. In addition, the proposed models achieve state-of-the-art performance in chemical/biomedical/scientific NER.

2 Existing Multi-Task Learning

This section describes existing multi-task learning methods which use training data of a different task other than the target task. We first outline the NER model used as the base model, and then describe an extension of the NER model to multi-task learning, where multiple tasks are trained simultaneously. In this multi-task learning, the target task and the other tasks are treated equally. Then, we explain an existing auxiliary learning model, which uses training data for a different task from the target task as auxiliary training data.

2.1 Multi-Task Learning Model

In this study, we use the BiLSTM-CRF model proposed by Huang et al. (2015) as our baseline NER model. The BiLSTM-CRF model is a sequence labeling model composed of bi-directional LSTM and CRF.

The BiLSTM-CRM model first computes the intermediate representation of each word in an input sentence using bidirectional LSTM. Let an input sentence be $\mathbf{w} = w_1, w_2, \cdots, w_N$ and the embedding vectors outputted by an embedding layer be $\mathbf{x} = \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$. The intermediate representation $\mathbf{e_i}$ of the word w_i is calculated as follows:

$$\overrightarrow{\mathbf{h_i}} = LSTM^{(f)}(\mathbf{x_i}, \overrightarrow{\mathbf{h_{i-1}}}), \tag{1}$$

$$\overleftarrow{\mathbf{h_i}} = LSTM^{(b)}(\mathbf{x_i}, \overleftarrow{\mathbf{h_{i+1}}}), \tag{2}$$

$$\mathbf{h_i} = [\overrightarrow{\mathbf{h_i}}; \overleftarrow{\mathbf{h_i}}], \tag{3}$$

$$\mathbf{e_i} = \mathbf{W^{(e)}} \mathbf{h_i}, \tag{4}$$

where $\rightarrow$ and $\leftarrow$ denote forward and backward directions, respectively, and $LSTM^{(f)}$ and $LSTM^{(b)}$ are forward and backward LSTMs, respectively. ";" denotes the concatenation of vectors. $\mathbf{W^{(e)}} \in R^{k \times d}$ is a weight matrix, d is the dimension of the hidden state vector $\mathbf{h_i}$, and k is the number of labels to be identified.

Then, the intermediate representations $\mathbf{e}$ computed by the bi-directional LSTM are fed to the CRF layer to obtain a label sequence. The score function for the label sequence $\mathbf{y} = (y_1, y_2, \cdots, y_N)$ is defined by using the score matrix $\mathbf{P} = (\mathbf{e_1}, \mathbf{e_2}, \cdots \mathbf{e_N})^T$, which is converted from the intermediate representations $\mathbf{e}$, and the transition score matrix $\mathbf{A}$ as follows:

$$\mathrm{s}(\mathbf{e}, \mathbf{y}) = \sum_{i=0}^{N} A_{y_i, y_{i+1}} + \sum_{i=1}^{N} P_{i, y_i}, \tag{5}$$

where $A_{i,j}$ represents the transition score from the label i to the label j. The output label sequence $\mathbf{y}^*$ is obtained by finding $\mathbf{y}$ that maximizes the score as follows:

$$\mathbf{y}^* = \arg\max_{\tilde{\mathbf{y}} \in \mathbf{Y_w}} \mathrm{s}(\mathbf{e}, \tilde{\mathbf{y}}), \tag{6}$$

where $\mathbf{Y_w}$ is the set of all possible label sequences for the input sentence $\mathbf{w}$.

Using the score function, the output probability of the label sequence $\mathbf{y}$ is defined by the softmax function as follows:

$$p(\mathbf{y}|\mathbf{w}) = \frac{\exp(\mathrm{s}(\mathbf{e}, \mathbf{y}))}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y_w}} \exp(\mathrm{s}(\mathbf{e}, \tilde{\mathbf{y}}))}. \tag{7}$$

In training, the parameters that minimize the following loss function are obtained:

$$L = -\sum_{(\mathbf{w}, \hat{\mathbf{y}}) \in D} \log(p(\hat{\mathbf{y}}|\mathbf{w})), \tag{8}$$

where D is a training dataset.

Figure 1 shows an overview of the BiLSTM-CRF model extended for multi-task learning. In the model, the word embedding layer and BiLSTM layer are shared by all the training datasets and the weights of these layers are the same on all the tasks. On the other hand, the CRF layer is prepared for each dataset and the weights of the CRF

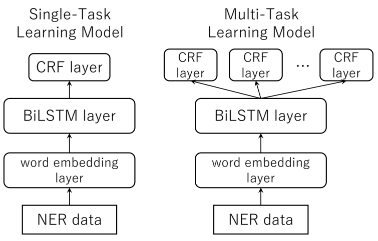

Figure 1: Overview of a multi-task learning model

layer are not shared. The objective function of the multi-task learning model is defined as follows:

$$Loss = \frac{1}{M}\sum_{i=1}^{M} L_i, \tag{9}$$

where L_i $(i = 1, 2, \cdots M)$ is the loss in the CRF layer for each training dataset (see Eq. 8), and M is the number of training datasets.

In the multi-task learning model, training data for the target task and that for the other tasks are treated equally, and thus an NER model common to all the tasks is learned. Larger datasets require more batches during training. In inference, NER is performed by using the CRF layer corresponding to the target task in the learned NER model.

2.2 Auxiliary Learning Model

Wang et al. (2019a) have proposed an auxiliary learning method, which is a multi-task learning method that distinguishes between training data for the target task (main training data) and that for the other task (auxiliary training data), and have improved NER performance for the target task. The auxiliary learning model is trained by using a main batch composed of main training data and an auxiliary batch composed of auxiliary training data. In each iteration, the model parameters are updated by the auxiliary batch first, and then by the main batch. This alternating updates by the main and auxiliary batches are repeated until the loss on the main training data converges.

Algorithm 1 shows the algorithm for the auxiliary learning method. In Algorithm 1, the subscripts denote the target task ($main$) and the auxiliary task (aux). $Epoch$ and $Iteration$ are the number of epochs and the number of iterations for the main task, respectively, and $BatchSize$ is the batch size. The number of iterations for each epoch is the total number of the main training data divided by the batch size (i.e., $Iteration = |D_{main}|/BatchSize$). The $extract$ function in lines 4 and 5 creates a batch by extracting $Batchsize$ data from the training dataset, and the $train$ function in lines 6 and 7 updates the parameters of the NER model $Model$ by using the batch data. The $is\_converge_{main}$ function in line 8 judges whether to stop training or not according to the loss on the target task.

Algorithm 1 Algorithm of an existing auxiliary learning method

```
Data: main training dataset $D_{main}$, auxiliary training dataset $D_{aux}$
1: for $i = 1$ to $EPOCH$ do
2:     for $j = 1$ to $ITERATION$ do
3:         $Batch_{main} = extract(D_{main}, BatchSize)$
4:         $Batch_{aux} = extract(D_{aux}, BatchSize)$
5:         $train(Model, Batch_{aux})$
6:         $train(Model, Batch_{main})$
7:     end for
8:     $is\_converge_{main}(Model)$
9: end for
```

Algorithm 2 Algorithm of the MUNCHABLES-concatenation model

```
Data: main training dataset $D_{main}$, $M$ auxiliary training datasets $D_{aux}^{(1)}, D_{aux}^{(2)}, \cdots, D_{aux}^{(M)}$
1: $D_{aux} = [D_{aux}^{(1)}; D_{aux}^{(2)}; \cdots; D_{aux}^{(M)}]$
2: for $i = 1$ to $EPOCH$ do
3:     for $j = 1$ to $ITERATION$ do
4:         $Batch_{main} = extract(D_{main}, BatchSize)$
5:         $Batch_{aux} = extract(D_{aux}, BatchSize)$
6:         $train(Model, Batch_{aux})$
7:         $train(Model, Batch_{main})$
8:     end for
9:     $is\_converge_{main}(Model)$
10: end for
```

3 MUNCHABLES: Multi-Auxiliary Learning

An existing auxiliary learning method uses only one auxiliary training dataset. In this section, we propose a new auxiliary learning paradigm, multi-auxiliary learning MUNCHABLES, that utilizes multiple training datasets as auxiliary training data. We first propose two MUNCHABLES models that use multiple auxiliary training datasets in single auxiliary learning (MUNCHABLES-concatenation model and MUNCHABLES-iteration model), and then propose a MUNCHABLES model that sequentially fine-tunes a main model by auxiliary learning with each auxiliary training dataset (MUNCHABLES-stack model).

3.1 MUNCHABLES-Concatenation Model

Algorithm 3 Algorithm of the MUNCHABLES-iteration model

```
Data: main training dataset $D_{main}$, $M$ auxiliary training
      datasets $D_{aux}^{(1)}, D_{aux}^{(2)}, \cdots, D_{aux}^{(M)}$
 1: for $i = 1$ to $EPOCH$ do
 2:     for $k = 1$ to $M$ do
 3:         for $j = 1$ to $ITERATION$ do
 4:             $Batch_{main} = extract(D_{main}, BatchSize)$
 5:             $Batch_{aux} = extract(D_{aux}^{(k)}, BatchSize)$
 6:             $train(Model, Batch_{aux})$
 7:             $train(Model, Batch_{main})$
 8:         end for
 9:     end for
10:     $is\_converge_{main}(Model)$
11: end for
```

Algorithm 4 Algorithm of the MUNCHABLES-stack model

```
Data: main training dataset $D_{main}$, $M$ auxiliary training
      datasets $D_{aux}^{(1)}, D_{aux}^{(2)}, \cdots, D_{aux}^{(M)}$
 1: for $k = 1$ to $M$ do
 2:     for $i = 1$ to $EPOCH$ do
 3:         for $j = 1$ to $ITERATION$ do
 4:             $Batch_{main} = extract(D_{main}, BatchSize)$
 5:             $Batch_{aux} = extract(D_{aux}^{(k)}, BatchSize)$
 6:             $train(Model, Batch_{aux})$
 7:             $train(Model, Batch_{main})$
 8:         end for
 9:         $is\_converge_{main}(Model)$
10:     end for
11: end for
```

The MUNCHABLES-concatenation model is a multi-auxiliary learning model that concatenates all the multiple auxiliary training datasets and treats the concatenated training data as one auxiliary training dataset in single auxiliary learning. Algorithm 2 shows the algorithm of the MUNCHABLES-concatenation model. Just like the existing single auxiliary learning model, the MUNCHABLES-concatenation model creates a main batch from the main training data and an auxiliary batch from the concatenated auxiliary training data. Then, the updates of model parameters with the auxiliary batch and with the main batch are repeated alternately until the loss on the main training dataset converges. The difference from the existing single auxiliary learning model is that an auxiliary batch is created from the concatenated data of multiple auxiliary training datasets, and thus an auxiliary batch can contain multiple types of auxiliary training data.

3.2 MUNCHABLES-Iteration Model

The MUNCHABLES-iteration model is a multi-auxiliary learning model which changes training datasets used as an auxiliary training dataset every epoch. Algorithm 3 shows algorithm of the MUNCHABLES-iteration model. The MUNCHABLES-iteration model alternately repeats parameter updates with the main batch created from the main training dataset and those with the auxiliary batch created from an auxiliary training dataset until the loss on the main training dataset converges as well as auxiliary learning models described so far. The difference from the MUNCHABLES-concatenation model is that an auxiliary batch in the MUNCHABLES-iteration model is created from a specific auxiliary training dataset and the source auxiliary training dataset is switched every epoch.

3.3 MUNCHABLES-Stack Model

The MUNCHABLES-stack model is a multi-auxiliary learning model that fine-tunes a main model as many as the number of auxiliary training datasets by sequential auxiliary learning with each auxiliary training dataset. Each auxiliary learning is performed by using a specific auxiliary training dataset as well as the existing single auxiliary learning. When the loss on the main training dataset converges, auxiliary data is switched to a new auxiliary training dataset and subsequently the main model is fine-tuned using the new auxiliary training dataset. Algorithm 4 shows the outline and algorithm of the MUNCHABLES-stack model, respectively. While, in the MUNCHABLES-concatenation model and MUNCHABLES-iteration model, a main model is trained only once (i.e., convergence is only once), in the MUNCHABLES-stack model, a main model is trained as many as the number of auxiliary training datasets.

4 Experiment

4.1 Experiment Settings

We evaluated our proposed models on eight chemical/biomedical/scientific domain NER tasks. Table 1 shows each NER dataset. We compared our three proposed models, the MUNCHABLES-concatenation model (*MUNCH.-Conc*), the MUNCHABLES-iteration model (*MUNCH.-Iter*), and the MUNCHABLES-stack model (*MUNCH.-Stack*), with three baseline models, the single task learning model (*SingleTask*), the standard multi-task learning model (*MultiTask*), and the existing single auxiliary learning model (*SingleAux*), which are described in Section 2. *MultiTask* learns one NER model from all the eight datasets. Our three MUNCHABLES models use all the datasets other than the target task (i.e., seven datasets) as auxiliary training data. *SingleAux* selected an auxiliary training dataset on the development data of the main task. Specifically, *SingleAux* used the model that achieved the best performance (i.e., F1-score) on the development data among seven models each of which is trained by single auxiliary learning with a training dataset for a task other than the target task, for testing. In *MUNCH.-Iter* and *MUNCH.-Stack*, the seven auxiliary training datasets were randomly sorted on condition that auxiliary datasets with the same NE type are not consecutive. We discuss the order of auxiliary training datasets in Section 5.1.

We implemented each NER model by extending the open framework FLAIR (Akbik et al., 2019). For word embeddings, we used Contextual String Embeddings (Akbik et al., 2018) and FastText (Bojanowski et al., 2017) provided by FLAIR, both of which were trained from the PubMed abstracts, a corpus of medical literature. The dimension of the BiLSTM layer was set to 256. We used the SGD optimizer, where a learning rate was adjusted by the following scheduling policy: the learning rate was reduced by a factor of two when the loss per epoch was not less than the minimum loss so far for four consecutive epochs, and training was terminated when the learning rate fell below 1e-4. We used the model at the end of training for testing. In hyperparameter tuning, we tried 0.1 and 0.05 as the initial learning rate and 16 and 32 as the batch size. Four models with these hyperparameter combinations were evaluated on the development data, and the hyperparameter set with the best performance was selected. In testing, we trained an NER model from the training data and the development data, and we reported and compared the performance on the test data. NER performance was evaluated by F1-score.

4.2 Experiment Results

Table 2 shows the experiment results. As can be seen in the table, *SingleAux* outperforms *SingleTask* and *MultiTask* on micro and macro average F1-scores. This suggests that auxiliary learning is more effective than the multi-task learning method where the training data for the target task and the other training data are equally treated. The observation is consistent with previously reported results. Table 2 also shows that *MUNCH.-Iter* and *MUNCH.-Stack* achieve higher performance than *SingleAux* on average and at least one of the MUNCHABLES models is better than *SingleAux* on all the tasks. These results experimentally demonstrate that NER performance can be improved by using multiple auxiliary training datasets in auxiliary learning as in the proposed models, which shows the effectiveness of the proposed auxiliary learning paradigm for NER.

In *MUNCH.-Iter* and *MUNCH.-Conc*, the main model only needs to be trained once, while *MUNCH.-Stack* requires fine-tuning on each auxiliary training dataset individually, so the training time for *MUNCH.-Stack* is longer than the other two MUNCHABLES models. Table 2 shows that *MUNCH.-Stack* achieves the best performance on two out of the eight tasks and its micro and macro average scores are the highest. This indicates the necessity of *MUNCH.-Stack* on some NER tasks even at longer training time.

5 Discussion

5.1 Discussion on the Order of Auxiliary Training Datasets

The performance of *MUNCH.-Iter* and *MUNCH.-Stack* might be affected by the order of auxiliary training datasets ($D_{aux}^{(1)}, D_{aux}^{(2)}, \cdots$ in Algorithms 3 and 4). This section discusses the impact of the order to NER performance.

In the experiments of Section 4, the auxiliary datasets were randomly sorted on condition that auxiliary datasets with the same NE type are not consecutive, in *MUNCH.-Iter* and *MUNCH.-Stack*. However, we conjecture that, in *MUNCH.-Iter* and *MUNCH.-Stack*, the auxiliary training dataset closer to the end of the training of the main model have a larger impact. Based on the conjecture, we sort the auxiliary training datasets in order of the degree of contribution to the performance improvement of the target task. Hereafter, the models are denoted as *MUNCH.-Iter (sort)* and *MUNCH.-Stack (sort)*. Specifically, we first evaluated the performance on the development data of the single auxiliary learning model with each auxiliary training dataset, and then sorted the auxiliary training datasets in ascending order of its single auxiliary learning models' performance and used the sorted training datasets in *MUNCH.-Iter* and *MUNCH.-Stack*.

We describe the order of auxiliary train-

Dataset	Type of NE	# of Sentences			# of Words			# of Annotations		
		Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
NCBI Disease (Doğan et al., 2014)	Disease	5,424	923	940	135,701	23,969	24,497	5,134	787	960
BC5CDR (Li et al., 2016)	Disease	4,560	4,581	4,797	118,170	117,453	124,750	4,182	4,246	4,424
BC5CDR (Li et al., 2016)	Drug/Chem	4,560	4,581	4,797	118,170	117,453	124,750	5,203	5,347	5,385
CHEMDNER (Krallinger et al., 2015)	Drug/Chem	30,682	30,639	26,364	893,685	887,805	767,636	29,478	29,486	25,346
BC2GM (Smith et al., 2008)	Gene/Protein	12,574	2,519	5,038	355,405	71,042	143,465	15,197	3,061	6,325
JNLPBA (Kim et al., 2004)	Gene/Protein	14,690	3,856	3,856	443,653	117,213	114,709	32,178	8,575	6,241
LINNAEUS (Gerner et al., 2010)	Species	11,935	4,078	7,142	281,273	93,877	165,095	2,119	711	1,433
s800 (Pafilis et al., 2013)	Species	5,733	830	1,630	147,291	22,217	42,298	2,557	384	767

Table 1: NER datasets

	NCBI-Disease	BC5CDR Disease	BC5CDR Chem	CHEMD NER	BC2GM	JNLPBA	LINN AEUS	s800	MA. AVG.	MI. AVG.
SingleTask	87.56	86.65	94.12	92.25	83.63	77.31	88.06	75.41	85.62	88.46
MultiTask	87.72	86.12	**94.53**	92.00	83.44	**77.86**	89.06	76.71	85.93	88.44
SingleAux	88.41	86.53	94.27	92.29	83.24	77.71	88.88	76.80	86.02	88.63
MUNCH.-Conc	**89.14**	86.73	94.23	**92.36**	82.57	77.48	**89.46**	76.42	86.05	88.49
MUNCH.-Iter	88.33	86.85	94.52	92.18	82.90	77.78	88.98	**77.20**	86.09	88.52
MUNCH.-Stack	87.69	**86.98**	94.33	92.32	**83.80**	77.62	89.42	76.65	**86.10**	**88.67**

Table 2: Experiment results (F1-score (%)). MA. AVG. and MI. AVG. indicate macro average and micro average, respectively. Each bold font value indicates the best result of each task.

Batch Size		16		32	
Learning Rate		0.05	0.1	0.05	0.1
Single Aux	NCBI-Disease	89.98	90.03	89.69	90.08
	BC5CDR Disease	89.95	90.08	89.87	89.93
	BC5CDR Chem	90.05	90.16	90.00	90.03
	BC2GM	89.95	89.92	89.88	89.91
	JNLPBA	89.90	89.93	89.88	89.89
	LINNAEUS	90.15	90.02	89.90	90.04
	s800	89.88	90.01	89.89	89.94
MUNCH.-Iter (sort)		90.04	90.11	90.03	**90.14**
MUNCH.-Stack (sort)		**90.24**	90.07	89.75	90.14

Table 3: Tuning of hyperparameters and the order of auxiliary training datasets of *MUNCH.-Iter (sort)* and *MUNCH.-Stack (sort)* on the CHEMDNER task (i.e., F1-score (%) on the CHEMDNER development data). In *MUNCH.-Iter (sort)* and *MUNCH.-Stack (sort)*, the best result is shown in bold font value.

Model	*MUNCH.-Iter*		*MUNCH.-Stack*	
Sort	w/o	w/	w/o	w/
NCBI-Disease	88.33	**88.50**	87.69	**87.90**
BC5CDR Disease	**86.85**	**86.85**	**86.98**	86.86
BC5CDR Chem	**94.52**	94.33	94.33	**94.47**
CHEMDNER	92.18	**92.39**	92.32	**92.35**
BC2GM	82.90	**83.59**	83.80	**83.84**
JNLPBA	**77.78**	77.28	**77.62**	77.21
LINNAEUS	**88.98**	88.82	**89.42**	88.87
s800	**77.20**	76.36	**76.65**	76.46
MA. AVG.	**86.09**	86.02	**86.10**	86.00
MI. AVG.	88.52	**88.64**	**88.67**	88.64

Table 4: Impact of the order of auxiliary training datasets. Each bold font value indicates the better result with or without sorting.

ing datasets in *MUNCH.-Iter (sort)* and *MUNCH.-Stack (sort)* for the CHEMDNER task as an example. Table 3 shows the performance on the CHEMDNER development data of *SingleAux* with each auxiliary NER training dataset for all combination of hyperparameters. Note that each model is trained from only training data to evaluate the performance on development data. As for *MUNCH.-Iter (sort)* and *MUNCH.-Stack (sort)*, the auxiliary training datasets are sorted on the basis of the performance of *SingleAux* with the same hyperparameter setting. From the table, for *MUNCH.-Iter (sort)*, the batch size and learning rate were set to 32 and 0.1, respectively, and the order of the auxiliary training datasets was set to "JNLPBA → BC2GM → BC5CDR-Disease → s800 → BC5CDR-Chem → LINNAEUS → NCBI-Disease." For *MUNCH.-Stack (sort)*, the batch size and learning rate were set to 16 and 0.05, respectively, and the order of the auxiliary training datasets was set to "s800 → JNLPBA → BC5CDR-Disease → BC2GM → NCBI-Disease → LINNAEUS → BC5CDR-Chem."

Table 4 shows the performance of *MUNCH.-Iter*,

	NCBI-Disease	BC5CDR Disease	BC5CDR Chem	CHEMDNER	BC2GM	JNLPBA	LINNAEUS	s800
SingleAux (reimpl)	88.41	86.53	94.27	92.29	83.24	77.71	88.88	76.80
BioBERT	**89.36**	86.56	93.44	91.41	84.4	77.59	**89.81**	75.31
HanPaNE	-	-	-	**92.57**	-	-	-	-
SciBERT	88.57	-	-	-	-	77.28	-	-
BioMegatron	87.0	**88.5**	92.5	-	-	-	-	-
SciFive	88.46	87.62	**94.61**	91.56	83.57	77.55	-	76.33
PubMedBERT (PubMed)	87.82	85.62	93.33	-	**84.52**	79.10	-	-
PubMedBERT (+PMC)	88.04	85.76	93.34	-	84.37	**79.16**	-	-
MUNCH.-Conc	89.14	86.73	94.23	92.36	82.57	77.48	89.46	76.42
MUNCH.-Iter	88.33	86.85	94.52	92.18	82.90	77.78	88.98	**77.20**
MUNCH.-Stack	87.69	86.98	94.33	92.32	83.80	77.62	89.42	76.65

Table 5: Comparison with previous results (F-measure (%)). These results are BioBERT (Lee et al., 2019), HanPaNE (Watanabe et al., 2019), SciBERT (Beltagy et al., 2019), BioMegatron (Shin et al., 2020), SciFive (Phan et al., 2021), and PubMedBERT (Gu et al., 2021). Each bold font value indicates the best result of each task.

MUNCH.-Iter (sort), *MUNCH.-Stack*, and *MUNCH.-Stack (sort)* on each test data. Table 4 shows that *MUNCH.-Iter (sort)* obtained a higher micro average than *MUNCH.-Iter* while the macro average of *MUNCH.-Iter (sort)* and the micro and macro averages of *MUNCH.-Stack (sort)* are worse than those of *MUNCH.-Iter* and *MUNCH.-Stack*, respectively. The results indicate that the performance of *MUNCH.-Iter* and *MUNCH.-Stack* are affected by the order of auxiliary training datasets and the performance could be improved by reordering auxiliary training datasets in ascending order of the performance on the development data of *SingleAux* on some NER tasks. We conjecture that sorting order of auxiliary training datasets might be affected by similarity of development data and test data. We will leave its further analysis for future work.

6 Related Work

Previous Methods		MUNCHABLES		*common*
v.s. Method	MA. AVG.	*Iter*	*Stack*	*tasks*
BioBERT	85.99	**86.09**	**86.10**	8
HanPaNE	92.57	92.18	92.32	1
SciBERT	82.93	**83.05**	82.66	2
BioMegatron	89.33	**89.90**	**89.67**	3
SciFive	85.67	**85.68**	85.63	7
PubMedBERT (PubMed)	86.08	86.08	86.08	5
PubMedBERT (+PMC)	86.13	86.08	86.08	5

Table 6: Summary of macro average F-measure (%). The *common tasks* indicates the number of tasks used by both of our MUNCHABLES and previous methods. We compared our MUNCHABLES methods with previous methods in terms of macro average F-measure on the common tasks. The bold font indicates that a MUNCHABLES model is better than the corresponding previous result.

6.1 Comparison with Previous Results

We compared our MUNCHABLES models with previous results[1] including state-of-the-art methods. Table 5 shows the results, and Table 6 shows a summary of the comparison, where we report macro average F-measure on the common tasks used by both of the MUNCHABLES and previous methods. As can be seen in Tables 5 and 6, in general, our MUNCHABLES models obtain competitive or better NER performance than previous results. These results show that our MUNCHABLES models achieve state-of-the-art performance on chemical/biomedical/scientific NER tasks. Another remarkable point is MUNCHABLES can be combined with the previous work. In other words, in order to improve the previous work, we can use MUNCHABLES in the previous work.

v.s. BioBERT BioBERT (Lee et al., 2019) is a BERT-based pre-training model trained with biomedical domain text. We compared BioBERT v1.0 with PubMed + PMC for its pre-training with our MUNCHABLES models. The macro average of BioBERT was 85.99 and those of *MUNCH.-Iter* and *MUNCH.-Stack* are 86.09 and 86.10. Our MUNCHABLES models obtained a higher performance than BioBERT.

v.s. HanPaNE HanPaNE (Watanabe et al., 2019) is a BiLSTM-CRF NER model that jointly learns an LSTM-based chemical compound paraphrase model through multi-task learning. HanPaNE showed 92.57 F-measure on CHEMDNER, which is the state-of-the-art performance on the

[1]If results obtained by different parameters were reported, we listed the results of the model that showed the best macro average F-measure on the NER datasets.

dataset. *MUNCH.-Stack* and *MUNCH.-Iter* are worse than HanPaNE. However, our MUNCHABLES model is compatible with HanPaNE and the models must complement each other. Therefore, we expect higher performance by combining MUNCHABLES with HanPaNE.

v.s. SciBERT SciBERT (Beltagy et al., 2019) is a BERT-based pre-training model trained with scientific domain text. SciBERT was evaluated on NCBI-Diseases and JNLPBA, and the macro average was 82.93. *MUNCH.-Iter* obtained a higher average (i.e., 83.05) than SciBERT.

v.s. BioMegatron BioMegatron (Shin et al., 2020) is a biomedical adaptation of a transformer model called Megatron-LM (Shoeybi et al., 2020). BioMegatron was evaluated on NCBI-Disease, BC5CDR Disease, and BC5CDR Chem. The macro average of BioMegatron with a 50k biomedical domain vocabularies and 345m parameters was 89.33, whereas *MUNCH.-Iter* and *MUNCH.-Stack* showed 89.90 and 89.67, which are higher than BioMegatron.

v.s. SciFive SciFive (Phan et al., 2021) is a domain-specific Text-to-Text Transfer Transformer (T5) (Raffel et al., 2020) model that has been pre-trained on large biomedical corpora. SciFive was evaluated on seven tasks out of the eight tasks except for LINNAEUS. The macro average F-measure of SciFive with PMC pre-training data was 85.67, whereas *MUNCH.-Iter* and *MUNCH.-Stack* were 85.68 and 85.63.

v.s. PubMedBERT PubMedBERT (Gu et al., 2021) is a BERT-based model trained with biomedical domain text from scratch. PubMedBERT (PubMed) was trained with only PubMed and PubMedBERT (+PMC) was trained with PubMed and PMC and these two models were evaluated on NCBI-Disease, BC5CDR Disease, BC5CDR Chem, BC2GM, and JNLPBA for NER. The macro average of PubMedBER (PubMed) was 86.08 and that of PubMedBERT (+PMC) was 86.13. *MUNCH.-Iter* and *MUNCH.-Stack* show the comparable accuracy as PubMedBERT (PubMed), however they showed lower accuracy than PubMedBERT (+PMC). We think that this difference was caused by the pretraining data size. The MUNCHABLES models were pretrained only with PubMed. Therefore, further improvement by increasing the amount of pretraining data is expected. Furthermore, the MUNCHABLES can be incorporated into PubMedBERT, therefore, we expect higher performance by enhancing PubMedBERT with MUNCHABLES.

6.2 Multi-Task Learning

Multi-task learning is employed to boost the performance of NLP systems (Liu et al., 2015; Luong et al., 2016; Dong et al., 2015; Hashimoto et al., 2017), including NER (Liu et al., 2018). Multi-task learning of sequence labeling with language models was proposed (Rei, 2017). Aguilar et al. (2018) and Cao et al. (2018) proposed multi-task learning of NER with word segmentation. Peng and Dredze (2017) proposed multi-task learning that leverages the performance of domain adaptation. Clark et al. (2018) proposed multi-task learning of NER with several NLP tasks such as POS tagging and parsing. Crichton et al. (2017b) and Wang et al. (2018) proposed multi-task learning of several tasks of biomedical NLP to increase NER performance. Watanabe et al. (2019) proposed multi-task learning of NER with chemical compound paraphrase.

Sampling methods for multi-task learning have also been proposed. Guo et al. (2019) is a two-stage mulit-task pipeline, where the first stage automatically selects the most useful auxiliary tasks via a Beta-Bernoulli multi-armed bandit with Thompson Sampling and the second stage learns the training mixing ratio of these selected auxiliary tasks. Kung et al. (2021) proposed a sampling method for training samples of auxiliary tasks based on the assumption that the more similar to the target task is, the more benefit is obtained for the target task.

7 Conclusion

This paper proposed a new auxiliary learning paradigm for NER, MUNCHABLES, that utilizes multiple training datasets as auxiliary training data for improving the performance of its target task. The experiments on eight chemical/biomedical/scientific domain NER datasets, showed that our proposed models achieved higher performance on average than conventional multi-task learning methods and an auxiliary learning method using only one auxiliary training dataset. Moreover, our proposed models achieved the state-of-the-art performance on chemical/biomedical/scientific NER tasks.

References

Gustavo Aguilar, Adrian Pastor López Monroy, Fabio González, and Thamar Solorio. 2018. Modeling noisiness to recognize named entities using multitask neural networks on social media. In *Proceedings of the 2018 Conference of the NAACL-HLT, Volum e 1 (Long Papers)*, pages 1401–1412.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 NAACL-HLT (Demonstrations)*, pages 54–59.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th COLING*, pages 1638–1649.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 EMNLP-IJCNLP*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 EMNLP*, pages 182–192.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 EMNLP*, pages 1914–1925. Association for Computational Linguistics.

Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017a. A neural network multi-task learning approach to biomedical named entity recognition. *BMC bioinformatics*, 18(1):1–14.

Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017b. A neural network multi-task learning approach to biomedical named entity recognition. *BMC Bioinformatics*, 18.

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

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd ACL and the 7th IJCNLP (Volume 1: Long Papers)*, pages 1723–1732.

Martin Gerner, Goran Nenadic, and Casey M Bergman. 2010. Linnaeus: a species name identification system for biomedical literature. *BMC bioinformatics*, 11(1):1–17.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Trans. Comput. Healthcare*, 3(1).

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2019. AutoSeM: Automatic task selection and mixing in multi-task learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3520–3531, Minneapolis, Minnesota. Association for Computational Linguistics.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 EMNLP*, pages 1923–1933.

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

Muhammad Raza Khan, Morteza Ziyadi, and Mohamed AbdelHady. 2020. Mt-bioner: Multi-task learning for biomedical named entity recognition using deep bidirectional transformers. *arXiv preprint arXiv:2001.08904*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the NLPBA/BioNLP*, pages 70–75. Citeseer.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(1):1–17.

Po-Nien Kung, Sheng-Siang Yin, Yi-Cheng Chen, Tse-Hsuan Yang, and Yun-Nung Chen. 2021. Efficient multi-task auxiliary learning: Selecting auxiliary data by feature similarity. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 416–428, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus:

a resource for chemical disease relation extraction. *Database*, 2016.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 NAACL-HLT*, pages 912–921.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 2016 ICLR*.

Tahir Mehmood, Alfonso E Gerevini, Alberto Lavelli, and Ivan Serina. 2020. Combining multi-task learning with transfer learning for biomedical named entity recognition. *Procedia Computer Science*, 176:848–857.

Evangelos Pafilis, Sune P Frankild, Lucia Fanini, Sarah Faulwetter, Christina Pavloudi, Aikaterini Vasileiadou, Christos Arvanitidis, and Lars Juhl Jensen. 2013. The species and organisms resources for fast and accurate identification of taxonomic names in text. *PloS one*, 8(6):e65390.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on RepL4NLP*, pages 91–100.

Long N. Phan, James T. Anibal, Hieu Tran, Shaurya Chanana, Erol Bahadroglu, Alec Peltekian, and Grégoire Altan-Bonnet. 2021. Scifive: a text-to-text transformer model for biomedical literature. *CoRR*, abs/2106.03598.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th ACL (Volume 1: Long Papers)*, pages 2121–2130. Association for Computational Linguistics.

Hoo-Chang Shin, Yang Zhang, Evelina Bakhturina, Raul Puri, Mostofa Patwary, Mohammad Shoeybi, and Raghav Mani. 2020. BioMegatron: Larger biomedical domain language model. In *Proceedings of the 2020 EMNLP*, pages 4700–4706, Online. Association for Computational Linguistics.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-lm: Training multi-billion parameter language models using model parallelism.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. 2008. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(2):1–19.

Xi Wang, Jiagao Lyu, Li Dong, and Ke Xu. 2019a. Multitask learning for biomedical named entity recognition with cross-sharing structure. *BMC bioinformatics*, 20(1):1–13.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, page bty869.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2019b. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, 35(10):1745–1752.

Taiki Watanabe, Akihiro Tamura, Takashi Ninomiya, Takuya Makino, and Tomoya Iwakura. 2019. Multi-task learning for chemical named entity recognition with chemical compound paraphrasing. In *Proceedings of the 2019 EMNLP-IJCNLP*, pages 6244–6249, Hong Kong, China. Association for Computational Linguistics.