Model Transfer with Explicit Knowledge of the Relation between Class Definitions

Hiyori Yoshikawa and **Tomoya Iwakura**
Fujitsu Laboratories Ltd., Kanagawa, Japan
{y.hiyori, iwakura.tomoya}@jp.fujitsu.com

Abstract

This paper investigates learning methods for multi-class classification using labeled data for the *target* classification scheme and another labeled data for a similar but different classification scheme (*support* scheme). We show that if we have prior knowledge about the relation between support and target classification schemes in the form of a *class correspondence table*, we can use it to improve the model performance further than the simple multi-task learning approach. Instead of learning the individual classification layers for the support and target schemes, the proposed method converts the class label of each example on the support scheme into a set of candidate class labels on the target scheme via the class correspondence table, and then uses the candidate labels to learn the classification layer for the target scheme. We evaluate the proposed method on two tasks in NLP. The experimental results show that our method effectively learns the target schemes especially for the classes that have a tight connection to certain support classes.

1 Introduction

Machine learning based methods have shown high performance in many NLP tasks, which typically are formulated as some kinds of classification problems. Although there has been a remarkable progress in methods utilizing unlabeled resources, many tasks still require a large amount (at least thousands, in some cases millions or billions) of high quality labeled data to achieve high accuracy.

For many tasks, however, classification schemes vary depending on fields of application or other factors, and large and high quality labeled data following a single scheme is insufficient. Named entity recognition (NER) (Nadeau and Sekine, 2007) and text classification (TC) (Joachims, 1998) are typical examples that allow variable classification schemes. For example, there are two kinds of NE type definitions for Japanese NER: IREX (Sekine and Isahara, 2000) with eight entity types and Sekine's extended NE (ENE) hierarchy with 200 entity types (Sekine et al., 2002). These two schemes are relevant but not in complete correspondence, *i.e.*, a class in ENE is not necessarily a proper subclass of a class in IREX and vice versa. For example, entities with `LOCATION` type in IREX are (a subtype of) `LOCATION` or `FACILITY` in ENE, while some entities with `FACILITY` type in ENE can also be `ORGANIZATION` in IREX. It is also the case that a classification scheme for an existing model is revised. In the case of news categolization, for example, a new category such as `world cup` would be added at a certain point of time; or the articles about eSports would newly categorised the existing `sports` category.

To obtain labeled data following the desired scheme, it is often required to create them almost from scratch or modify existing annotation because the existing data follow partly different schemes. However, the annotation processes to create such on-demand labeled data usually take too much cost to obtain enough data.

This paper addresses the methods to utilize the existing large amount of labeled data with a different classification scheme (*support scheme*) to learn a good model for the *target scheme* with a small amount of corresponding labeled data. One possible solution is the multi-task learning approach (Caruana, 1997) in which the model for each classification scheme is learned while sharing the model parameters for the input representation. A drawback of typical multi-task approaches is that they cannot exploit relation between two schemes directly, even if we know it in advance. The problem becomes critical when it is required to *preserve* the classification performance on the

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 541–550
Brussels, Belgium, October 31 - November 1, 2018.

classes that are tightly connected to those of the support scheme. This corresponds to the following practical situation. We have a model working on some systems, and are required to modify it to adapt to a new classification scheme given only a small amount of examples related to the change of the scheme. It is also required that the performance of the retrained model is almost unchanged for the input examples that is not related to the change of the scheme. In the simple multi-task learning, the classification layer for the target scheme is learned only from the small labeled data for the target scheme. Such small data are often insufficient to learn *existing* classes in spite of the shared input representation.

In this paper, we propose a method to exploit the relation between the two classification schemes which is given in the form of a class correspondence table described in Section 3. Instead of learning the individual classification layers for the support and target schemes, the proposed method converts the class label of each example on the support scheme into a set of candidate class labels on the target scheme via the class correspondence table, and then uses it to learn the classification layer for the target scheme using the *learning with multiple labels* framework (Jin and Ghahramani, 2002). The difference from the typical multi-task learning methods is that the large amount of labeled data on the support scheme are directly used to learn the classification layer for the target scheme. It enables the model to learn the target scheme while preserving the performance on those classes which are tightly connected to the support scheme effectively. We conduct experiments for two tasks in NLP to verify the effectiveness of our proposed method.

The contribution of this paper is as follows.

- We propose a method to utilize the known relation of the two classification schemes by using the relation as an explicit constraint.
- We evaluated the proposed method on two task with public data and original but reproducible classification schemes.

The proposed method has the following advantages.

- We can utilize the prior knowledge on the relation between the support and the target classification schemes to effectively constrain the model.
- The method can learn the classes existing in the support scheme, even when the target labeled data contain few or no examples on these classes.
- It can also be used for such tasks in which the output is structured and difficult to be separated, *e.g.*, NER.
- The proposed method can be applied to the most of current neural network based models which output a probability distribution and take loss to update parameters with learning method such as SGDs. There is no need to violate the original network architecture.

2 Preliminaries

2.1 Problem Settings

The goal is to learn a classification scheme (*target scheme*) $f_{\mathrm{T}} : \mathcal{X} \to \mathcal{Y}_{\mathrm{T}}$ for a certain input space $\mathcal{X}$ (*e.g.*, sentences) and a set of class labels $\mathcal{Y}_{\mathrm{T}}$. We assume that the model to learn takes an input $x \in \mathcal{X}$ and predicts a probability distribution $p_{\mathrm{T}}(y|x;\theta_{\mathrm{T}})$ over $\mathcal{Y}_{\mathrm{T}}$, where θ_{T} represents the model parameters to learn. We focus on the situation that we have only a small amount of labeled data $D_{\mathrm{T}} = \{(x_i, y_i)\}_{i=1}^{N_{\mathrm{T}}}$. Instead, we have a large amount of labeled data $D_{\mathrm{S}} = \{(x_i, y_i)\}_{i=1}^{N_{\mathrm{S}}}$, where x_i is from the same input space $\mathcal{X}$ and the same domain distribution but y_i is from a different set of class labels $\mathcal{Y}_{\mathrm{S}}$. We denote $f_{\mathrm{S}} : \mathcal{X} \to \mathcal{Y}_{\mathrm{S}}$ by the classification scheme (*support scheme*) followed by D_{S}. In addition, we have prior knowledge about the relation between these two schemes f_{S} and f_{T}. We introduce the relation formally in Section 3.

In general, we can assume multiple support and target schemes. But this paper describes the cases of a single support scheme and a single target scheme for simplicity. Note that the formulation in the following can be extended to multi-support and multi-target cases straightforwardly.

2.2 Multi-task Learning

We first review simple multi-task learning on two tasks, which we use as a baseline as well as the basis of the proposed method.

In multi-task learning, the probability ditributions on both $\mathcal{Y}_{\mathrm{S}}$ and $\mathcal{Y}_{\mathrm{T}}$ are learned simultaneously with sharing a part of their model parameters. Let θ_{R} denote the shared part, θ_{CT} the part specific to the target model , and θ_{CS} the parame-

ters specific to the support model. Then the probability distributions on the support and the target schemes can be written as $p_{\mathrm{S}}(y|x;\theta_{\mathrm{R}},\theta_{\mathrm{CS}})$ and $p_{\mathrm{T}}(y|x;\theta_{\mathrm{R}},\theta_{\mathrm{CT}})$, respectively.

For training, the following loss function is minimized:

$$L_{\mathrm{MT}} = L_{\mathrm{T}} + \lambda L_{\mathrm{S}}, \tag{1}$$

where

$$L_{\mathrm{T}} = -\frac{1}{N_{\mathrm{T}}} \sum_{(x,y)\in D_{\mathrm{T}}} \log p_{\mathrm{T}}(y|x;\theta_{\mathrm{R}},\theta_{\mathrm{CT}}), \tag{2}$$

$$L_{\mathrm{S}} = -\frac{1}{N_{\mathrm{S}}} \sum_{(x,y)\in D_{\mathrm{S}}} \log p_{\mathrm{S}}(y|x;\theta_{\mathrm{R}},\theta_{\mathrm{CS}}), \tag{3}$$

and λ is a real-valued hyperparameter for specifying the weight of the support loss.

2.3 Learning with Multiple Labels

In learning with multiple labels (LwML) framework, each training example (x, Y) consists of an input x and a set of candidate labels Y instead of a single true label. It is assumed that the only one label in Y is correct for x, and the objective is to learn a classifier that maps inputs to the correct labels. To deal with the problem, the loss fucntion consists of the likelihood for the predicted distribution to be high within the candidate label set:

$$l_{\mathrm{ML}}(x, Y, p) = \log \sum_{y\in Y} p(y|x). \tag{4}$$

3 Proposed Methods

In this section, we introduce relations between schemes in the form of *class correspondence table*, and how it is used for training a classifier for the target scheme.

3.1 Class Correspondence Table

We suppose that the two schemes introduced in Section 2.1 have a strong relation in that for an input x, the candidates for its class on the target scheme can be limited by its class on the support scheme. Here we give examples for two scheme sets introduced in Section 1. For Japanese NE type definitions, an entity with `LOCATION` type in IREX definition can be (a subtype of) `LOCATION` or `FACILITY` in ENE, but it cannot be other type such as `PERSON`, `DISEASE` or `COLOR`. For news categolization, `sports` category in the target scheme comes only from `sports` and the categories which include articles about eSports in the support scheme.

Formally, we consider the following *class correspondence table*. The class correspondence table $\mathcal{T}$ is a map from a class in $\mathcal{Y}_{\mathrm{S}}$ to a set of classes in $\mathcal{Y}_{\mathrm{T}}$. It functions as a constraint on the target scheme f_{T}. Namely, the class $y_{\mathrm{T}} = f_{\mathrm{T}}(x)$ must be a member of $\mathcal{T}(y_{\mathrm{S}})$, where $y_{\mathrm{S}} = f_{\mathrm{S}}(x)$.

There are some possible ways to construct a class correspondence table. One is to define it by hands. For example, if the ontology related to the classification scheme is known, it is straightforward to define the class correspondence table according to the ontology. Another way is to define it from data. First, we apply the model learned for the support scheme to the examples in the labeled data for the target scheme. Then, we obtain pairs of labels on the support and the target schemes. The class correspondence table can be defined by allowing the pair that appears in the dataset at a certain frequency. While the method can automatically define the relation, there is a risk to drop the possible relation that is not found in the given dataset, or because of the insufficient model accuracy. We propose a model to alleviate the problem in Section 3.3.

3.2 Multi-Task Learning with Multiple Candidate Labels

For training, the following loss function is minimized:

$$L_{\mathrm{CS}} = L_{\mathrm{T}} + \lambda L_{\mathrm{SCS}}, \tag{5}$$

where

$$L_{\mathrm{SCS}} = -\frac{1}{N_{\mathrm{S}}} \cdot \sum_{(x,y)\in D_{\mathrm{S}}} l_{\mathrm{ML}}(x, \mathcal{T}(y), p_{\mathrm{T}}(y|x;\theta_{\mathrm{R}},\theta_{\mathrm{CT}})), \tag{6}$$

with a real-valued hyperparameter λ. The first term corresponds to the loss from the target dataset, and the second for the support dataset in the form of LwML with candidate classes given by the class correspondence table. Compared with the simple multi-task learning, our method trains only θ_{T} (*i.e.* θ_{R} and θ_{CT}) and does not require the parameters specific to support scheme. We call this model as **Class Shift constraint** (**CS**) model.

To get an intuition, let us see the special cases. If the support scheme is equal to the target scheme and the class correspondence is identity, then the loss (5) behaves just like a single-task learning with labeled data consisting of the support and the target data. If the class correspondence table allows all class shift for all classes in the support scheme, then L_{SCS} is always zero and so the support dataset has completely no effect on training.

3.3 Combination with Simple Multi-Task Learning

While CS model can exploit the prior knowledge about class correspondence, it has a potential problem that the class correspondence table can work inadequately. For example, if we construct the class correspondence table from some data automatically, there can be some overlooked relation because they just do not exist in the given data.

To overcome this problem, we propose an extention of the CS model which relaxes the class shift constraint by combining it with the loss from the simple multi-task learning:

$$L_{\mathrm{MTCS}} = L_{\mathrm{T}} + \lambda \left\{ \mu L_{\mathrm{SCS}} + (1-\mu) L_{\mathrm{S}} \right\}, \quad (7)$$

with an additional hyperparameter $\mu \in [0, 1]$. We call this model as **Multi-Task with Class Shift constraint** (**MTCS**) model.

3.4 Training

For the following experiments with neural-based models, we adopt training by stochastic gradient descent (SGD) with mini-batches. For each iteration, we sample b examples from each of the support and the target labeled data, where b is the mini-batch size. Then the loss is calculated by (5) or (7) for the batch.

We suppose that the model is trained with a large amount of labeled data on the support scheme. When such labeled data is unavailable, however, it is possible to obtain pseudo-labeled data by applying the model for the support scheme to unlabeled data, and use them to train the model for the target scheme.

4 Experiments

We evaluated the proposed method on two tasks: named entity recognition (NER) and text classification (TC).

To examine the effectiveness of the proposed method, we adopted datasets that are not only large but also accompanied with well-organized ontology. We defined the target schemes following the exsiting shared tasks, while we defined different support schemes according to their ontology so that the labels correspond to the different level or granularity in the same ontology. By doing so, we can compare the proposed method with an ideal settings where all data is labeled according to the target scheme, which can be seen as an upper bound.

4.1 Named Entity Recognition (NER) task

We conducted NER task on GENIA corpus. GENIA corpus (Kim et al., 2003) was developed as a resource for text mining in biomedical literature. It contains annotated text for 2,000 Medline abstracts, and the annotated information includes term annotation for entities related to biological components such as proteins, genes and cells.

As described in (Kim et al., 2003), the entities are annotated according to hierarchical ontology, and have 36 types. BioNLP / JNLPBA shared task (Kim et al., 2004) is organized by GENIA project as well. The task is to extract named entities of 5 types, which is defined by integrating the above 36 types following the ontology.

We used the JNLPBA definition as the target scheme, and made another definition for the support scheme. We show the class correspondence in Table 1. Note that the `O` (no tag) class in the target scheme corresponds to `Nucleic_acid` and `O` classes in the support scheme. It means that the shift from the support scheme to the target scheme involves both class subdivision and integration. We also note that we follow BIOES (Collobert et al., 2011) representation to convert the NE tags into the word-level class labels. It means that each NE class (say `XXX`) except `O` corresponds to 4 word-level classes (`S-XXX`, `B-XXX`, `I-XXX`, and `E-XXX`). We construct the class correspondence table by associating the labels with same prefix for each corresponding label pair. For instance, if a class `XXX` in the support scheme corresponds to the label `YYY` in the target scheme, the word-level class label `S-XXX` corresponds to `S-YYY`, and so on. This setting is based on a strong assumption that the modification of the tagging scheme does not change the range of named entity mentions. As it is not always true, the relaxed formulation (7) is expected to work better.

We created the NER input for the support

Support scheme	Target scheme	Original GENIA ontology
Nucleic_acid	DNA	DNA (5 types)
	RNA	RNA (5 types)
	O (no tag)	polynucleotide, nucleotide
Protein	Protein	protein (7 types)
O (no tag)	Cell_type	cell_type
	Cell_line	cell_line
	O (no tag)	others (15 types) + no tag

Table 1: Class correspondence table for NER task.

Support				
Sentences		14838		
Types	Nucleic_acid	6954		
	Protein	26777		
Target		Train	Dev	Test
Sentences		2966	742	3856
Types	DNA	1868	412	1056
	RNA	206	50	118
	Protein	3634	1257	5067
	Cell_type	887	241	1921
	Cell_line	635	176	500

Table 2: Data statistics of NER task.

scheme from the original GENIA corpus. For tokenization we used NLTK (Bird et al., 2009), and then broke tokens at the start and the end of the entity mentions. Since the task does not allow overlap of entity mentions, we chose the shortest mentions and discarded longer ones when mentions are nested in the original corpus. We used JNLPBA dataset as the input for the target scheme. Table 2 shows the statistics of the dataset.

4.2 Text Classification (TC) task

For TC, we used DBpedia ontology classification dataset created by (Zhang et al., 2015). Each sample in the dataset consists of the description text and the class of a DBpedia (Lehmann et al., 2015) entry. The entries are chosen from 14 ontology classes. We use these classes as the target scheme, and defined the support scheme by integrating the categories into 5 classes. The class correspondence is shown in Table 3. Table 4 shows the statistics of the dataset.

4.3 Baseline Methods

We compare the following methods with the proposed methods CS and MTCS described in Section 3.

- **Target Only** trains a model with only labeled data on the target scheme.
- **Finetune** method first trains a model with labeled data on the support scheme. Then, the model for the target scheme is trained for another set of labeled data with the shared part θ_{R} of the parameters initilaized with the value trained on the support data (Razavian et al., 2014).
- **MT** is the multi-task learning method described in Section 2.2.
- **ALL Target** represents training on an ideal situation that all training examples are labeled according to the target scheme. The total number of training examples of each task is the sum of the number of labeled data for the support and the target schemes.

We also initialized model parameters for MT, CS and MTCS with the Finetune method.

4.4 Models and Training Settings

For NER, we used the model similar to the one described in (Ma and Hovy, 2016) with the same network parameters, except that we used the sum of word-level loss as in (Collobert and Weston, 2008) instead of the structural loss[1][2], mainly because of the computation time. As a result, the model is trained as a simple word-level label classification.

For TC, we used a simple softmax model which is similar to fastText (Grave et al., 2017) model with the same network parameters except we use pretrained word embeddings; we use simple softmax instead of hierarchical softmax; and only bag-of-words features are used to construct input representation.

[1]It is reported that its performance is competitive to the structural loss (Chiu and Nichols, 2016).

[2]The training of CRF with multiple label candidates can be found in (Tsuboi et al., 2008).

Support scheme	Target scheme
Org	Company, EducationalInstitution
Artist	Artist
Non-artist	Athlete, OfficeHolder
Work	Album, Film, WrittenWork
Other	MeanOfTransportation, Building, NaturalPlace, Village, Animal, Plant

Table 3: Class correspondence table for TC task.

Support		
Sentences		500000
Types	Org	70688
	Artist	35419
	Non_artist	70544
	Work	106162
	Other	212187

Target (14 classes)	Train	Dev	Test
Sentences	5000	60000	70000
Sentences / class (ave.)	357.1	4285.7	5000

Table 4: Data statistics of TC task. We show only the average number of sentences per class for target since the data is highly balanced.

Hyperparameter	**NER**	**TC**
dropout rate	0.5	0.5
batch size	10	10
initial learning rate	0.1	0.075
learning rate decay	0.1	0.05
gradient clipping	0.5	0.5
λ (MT)	1.0	1.25
λ (CS)	1.0	1.0
λ (MTCS)	1.5	1.25
μ (MTCS)	0.5	0.6

Table 5: Hyperparameter settings.

For both models, all parameters except the softmax layers on the top are shared. We implemented these models using DyNet (Neubig et al., 2017) library.

The models are trained by SGD with mini-batches as mentioned in Section 3.4, and some optimization techniques are used including dropout, learning rate decay and gradient clipping following (Ma and Hovy, 2016). The result of hyperparameter tuning on development data is described in Table 5.

We used pretrained word embeddings from

Method	**NER (F1)**	**TC (Acc.)**
Target Only	66.02	93.29
Finetune	68.29	94.71
MT	68.73	94.91
CS	**68.73**	**95.81**
MTCS	**69.12**	**95.67**
All Target	72.23	96.81

Table 6: Results for NER and TC task.

PMC open access subset (commercial use version)[3] for NER and from Wikipedia dump (2010/10/11)[4] for TC.

4.5 Experimental Results

All of the following results are averaged over five runs.

Table 6 shows the F1 scores for NER task and accuracy scores for TC task. For NER, the performance of CS is competitive to MT, but by combining them (MTCS) we had the improved performance. On the other hand, for TC task CS outperforms MT in a certain degree. The effect of their combination of (MTCS) is limited on this task.

We also evaluated the effect of the class shift constraint when the amount of labeled data for the target scheme is quite small. Figure 1 and Figure 2 show how the scores improve as the size of the labeled data for the target scheme increases. We can see that the advantage of CS and MTCS methods are significant especially when the size of the labeled data for the target scheme is very small.

Next, we evaluated how the models *preserve* the classification performance on the classes that are tightly connected to some classes in the support scheme. We first trained a model on the support scheme (*support model*) using the labeled data for the support scheme. Next we transfer the model for the target scheme with the labeled data

[3]https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/
[4]https://dumps.wikimedia.org/backup-index.html

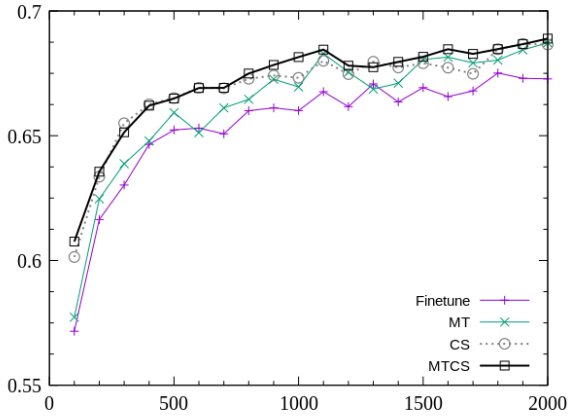

Figure 1: F1 scores with different data size on target scheme for NER task.

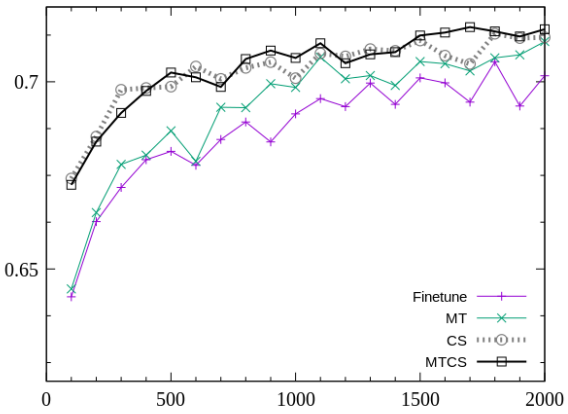

Figure 3: F1 scores with different data size on the target scheme for `Protein` class in NER task.

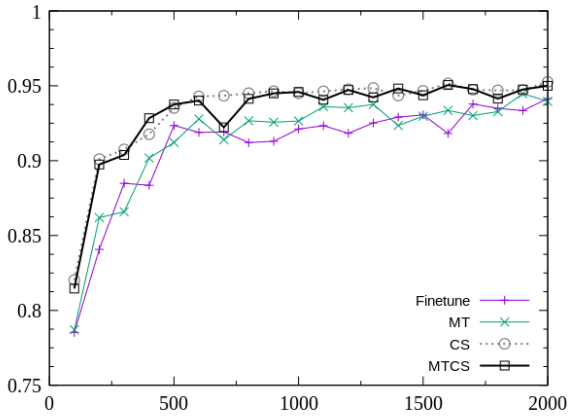

Figure 2: Accuracy scores with different data size on target scheme for TC task.

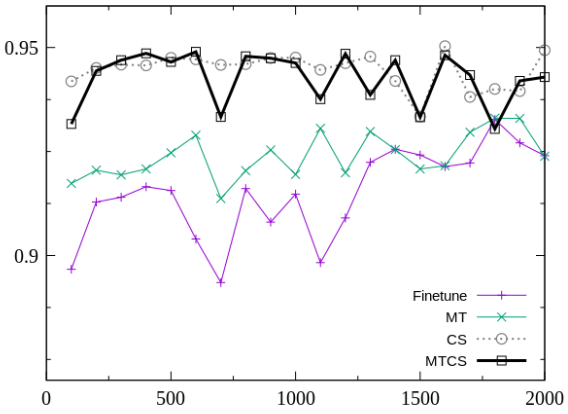

Figure 4: Accuracy scores with different data size on the target scheme for `Artist` class in TC task.

for the target scheme using Finetune, MT, CS, or MTCS. Then, test data are labeled by both the support model and the transferred model. By doing so, we can compare the classification performance of the support model and the transferred models on the *unchanged* classes for these tasks, namely `Protein` for NER and `Artist` for TC. We first check the performance of the transferred models on these classes with the small size of labeled data for the target schemes. Figure 3 and Figure 4 show the results. Compared to Finetune and MT, the performance of CS and MTCS is high even when the size of labeled data for the target scheme is very small. It suggests that the proposed methods effectively use the knowledge from support models for recognizing these classes. Table 7 shows the number of examples that are correctly classified by the transferred models out of those correctly classified by the support model. For NER task, the support model extracted 3193 out of 5067 `Protein` mentions correctly. For TC task, the support model categorized 4534 out of 5000 text with `Artist` labels correctly. We can see that the proposed methods succeed to prevent performance deterioration more effectively than Finetune and MT.

5 Related Work

Improving model performance with knowledge from other models or data sources is one of the central research topics in machine learning.

Domain adaptation methods utilize training data that have the same class definition but they come from different domains (Daumé III, 2007; Dai et al., 2007; Crammer and Mansour, 2012). These methods focus on the change of input distribution,

Method	NER-Protein	TC-Artist
Finetune	0.928 (2963)	0.966 (4380)
MT	0.933 (2979)	0.980 (4445)
CS	**0.956 (3053)**	**0.991 (4495)**
MTCS	**0.955 (3051)**	**0.989 (4486)**

Table 7: The proportion of performance preservation from the support model on unchanged classes. The numbers in the brackets represent the number of examples correctly classified by both the support and the target models.

not the classicication scheme.

Multi-task learning approaches with nerual networks often achieve this by sharing input representation among different tasks. The objectives of jointly learned tasks are often different and the mapping from the shared representation to the output for each task is learned independently (Liu et al., 2015; Hashimoto et al., 2017). Therefore, the relation between classification schemes is not directly considered.

Some studies focus on adding model capabilities to handle new tasks without storing all training data for old tasks. The central issue is to avoid catastrophic forgetting (Li and Hoiem, 2017), and several approaches have been explored (Lopez-Paz et al., 2017; Kirkpatrick et al., 2017; Triki et al., 2017). As with standard multi-task learning, many studies in this line assume different tasks with task-specific output models. iCaRL (Rebuffi et al., 2017) assumes a different problem named class-incremental learning. A stream of new class examples is observed and the model is required at any time to perform as a multi-class classifier on the classes observed so far. However, modifying the classification scheme on observed samples is not considered in this framework.

Knowledge distillation (Hinton et al., 2015) is another topic of knowledge transfer, which can be used to simplify a large complex classification model such as ensemble model by letting a simple model imitate the output distributions of the complex model instead of predicted labels. It is also used for preventing catastrophic forgetting in continual learning (Rebuffi et al., 2017; Shmelkov et al., 2017).

Learning with Hierarchy and Exclusion (HEX) graph (Deng et al., 2014) is a promising method utilizing pre-defined relationship between class labels. HEX graph can express exclusion and subsumption relations between class labels. Despite that it is originally used for a different kind of problems (multi-class classification which allows multiple labels for an input), it is possible to solve our problem setting in this framework. In fact, we can construct a HEX graph by assigning exclusive edges to all class pairs within the same scheme and to class pairs from different schemes which do not correspond in the class correspondence table. One of the main advantages of our method is the computational cost. Inference with HEX graph is sometimes computationally prohibitive depending on the graph structure, while inference with our model is not affected by the structure of the class correspondence table. In addition, HEX graph approach requires parameters of both support and target classes even at inference. Hence it is not suitable if the classification scheme can change many times.

Another related framework is semi-supervised learning, which use both labeled and unlabeled data. The approaches include use of classifiers trained with automatically generated training data from unlabeled data (Ando and Zhang, 2005), use of automatically labeled data (Suzuki and Isozaki, 2008), language model (Peters et al., 2017, 2018) trained from unlabeled data, and so on. In discriminative models, knowledge from unlabled data is often incorporated in the models as improved input representation or additional features. Since our method does not restrict input representation, such semi-supervised methods can be easily combined.

6 Conclusion

We have proposed a training method for the setting where we have only a small amount of labeled data for the target scheme, but have access to a large amount of labeled data for a related support scheme with the class correspondence table. The experimental results on a named entity recognition task and a text classification task showed that our proposed methods outperform finetune and simple multi-task learning methods.

Although the experiment for NER task showed that MTCS model has potential to work with a possibly incomplete class correspondence table, further experiments are necessary to verify its effectiveness on automatically generated class correspondence tables. Future work also includes applying our method to improve models learned from a single corpus by combining other corpora

with different schemes, experiments on multi-support and multi-target settings, and extensions to the case where input domain of labeled data for support and target schemes are different.

References

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proc. of ACL*, pages 1–9.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Koby Crammer and Yishay Mansour. 2012. Learning multiple tasks using shared hypotheses. In *Proc. of NIPS*, pages 1484–1492.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for transfer learning. In *Proc. of ICML*, pages 193–200.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*, pages 256–263.

Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. 2014. Large-scale object classification using label relation graphs. In *Proc. of ECCV*, pages 48–64.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *Proc. of EACL*, pages 427–431.

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

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Rong Jin and Zoubin Ghahramani. 2002. Learning with multiple labels. In *Proc. of NIPS*, pages 921–928.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of ECML*, pages 137–142.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Supplement 1):i180–i182.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences U.S.A*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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 *Proc. of NAACL*, pages 912–921.

David Lopez-Paz et al. 2017. Gradient episodic memory for continual learning. In *Proc. of NIPS*, pages 6467–6476.

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

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*, pages 1756–1765.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: An astounding baseline for recognition. In *Proc. of CVPR workshops*, pages 806–813.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. 2017. iCaRL: Incremental classifier and representation learning. pages 5533–5542.

Satoshi Sekine and Hitoshi Isahara. 2000. IREX: IR & IE evaluation project in japanese. In *Proc. of LREC*, pages 1977–1980.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. 2002. Extended named entity hierarchy. In *Proc. of LREC'02*.

Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. 2017. Incremental learning of object detectors without catastrophic forgetting. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3420–3429.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *Proc. of ACL*, pages 665–673.

Amal Rannen Triki, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. 2017. Encoder based lifelong learning. *Proc. of ICCV*, pages 1329–1337.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proc. of COLING*, pages 897–904.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*, pages 649–657.