HRKD: Hierarchical Relational Knowledge Distillation for Cross-domain Language Model Compression

Chenhe Dong[1], Yaliang Li[2], Ying Shen[1*], Minghui Qiu[2*]
[1] Sun Yat-sen University [2] Alibaba Group
dongchh@mail2.sysu.edu.cn, sheny76@mail.sysu.edu.cn
{yaliang.li, minghui.qmh}@alibaba-inc.com

Abstract

On many natural language processing tasks, large pre-trained language models (PLMs) have shown overwhelming performances compared with traditional neural network methods. Nevertheless, their huge model size and low inference speed have hindered the deployment on resource-limited devices in practice. In this paper, we target to compress PLMs with knowledge distillation, and propose a hierarchical relational knowledge distillation (HRKD) method to capture both hierarchical and domain relational information. Specifically, to enhance the model capability and transferability, we leverage the idea of meta-learning and set up domain-relational graphs to capture the relational information across different domains. And to dynamically select the most representative prototypes for each domain, we propose a hierarchical compare-aggregate mechanism to capture hierarchical relationships. Extensive experiments on public multi-domain datasets demonstrate the superior performance of our HRKD method as well as its strong few-shot learning ability. For reproducibility, we release the code at https://github.com/cheneydon/hrkd.

1 Introduction

Large pre-trained language models (PLMs) (e.g., BERT (Devlin et al., 2019)) have demonstrated their outperforming performances on a wide range of NLP tasks, such as machine translation (CONNEAU and Lample, 2019; Zhu et al., 2020), summarization (Zhang et al., 2019; Liu and Lapata, 2019), and dialogue generation (Bao et al., 2020; Zheng et al., 2020). However, their large size and slow inference speed have hindered practical deployments, such as deploying on resource-constrained devices.

To solve the above problem, many compression techniques for PLMs have been proposed, such as

quantization (Shen et al., 2020), weight pruning (Michel et al., 2019), and knowledge distillation (KD) (Sun et al., 2019; Jiao et al., 2020). Due to the plug-and-play feasibility of KD, it is the most commonly used method in practice, and we focus on it in this work. The purpose of KD is to transfer knowledge from a larger teacher model to a smaller student model (Hinton et al., 2015). Traditional KD methods only leverage single-domain knowledge, i.e., transferring the knowledge of the teacher model to the student model domain by domain. However, as stated in the purpose of transfer learning, the model performance on target domains can be improved by transferring the knowledge from different but related source domains (Lu et al., 2015), thus the cross-domain knowledge also plays an important role. In addition, several recent works have also proved the advantage of cross-domain knowledge, and many multi-domain KD methods have been proposed. For example, Peng et al. (2020); Yang et al. (2020) demonstrate the effectiveness of distilling knowledge from multiple teachers in different domains; Liu et al. (2019a,b) show that jointly distilling the student models of different domains can enhance the performance.

Nevertheless, these methods fail to capture the relational information across different domains and might have poor generalization ability. To enhance the transferability of the multi-domain KD framework, some researchers have recently adopted the idea of meta-learning. Some studies have pointed out that meta-learning can improve the transferability of models between different domains (Finn et al., 2017; Javed and White, 2019). For example, Meta-KD (Pan et al., 2020) introduces an instance-specific domain-expertise weighting technique to distill the knowledge from a meta-teacher trained across multiple domains to the student model. However, the Meta-KD framework trains student models in different domains separately, which is inconvenient in real-world applications and might not

---

*Corresponding author.

have enough capability to capture multi-domain correlations.

In this paper, we aim to simultaneously capture the relational information across different domains to make our framework more convenient and effective. Specifically, we set up several domain-relational graphs to adequately learn the relations of different domains and generate a set of domain-relational ratios to re-weight each domain during the KD process. Moreover, since different domains might have different preferences of layer prototypes, motivated by the Riesz representation theorem (Hartig, 1983), we first construct a set of reference prototypes for each domain, which is calculated by a self-attention mechanism to integrate the information of different domains. Then we introduce a hierarchical compare-aggregate mechanism to compare each layer prototype with the corresponding reference prototype and make an aggregation based on their similarities. The aggregated prototypes are finally sent to the corresponding domain-relational graphs. Our framework is referred to as hierarchical relational knowledge distillation (HRKD).

We evaluate the HRKD framework on two multi-domain NLP datasets, including the MNLI dataset (Williams et al., 2018) and the Amazon Reviews dataset (Blitzer et al., 2007). Experiments show that our HRKD method can achieve better performance compared with several multi-domain KD methods. We also evaluate our approach under the few-shot learning setting, and it can still achieve better results than the competing baselines.

2 Method

In this section, we detailedly describe the proposed HRKD framework. Our HRKD aims to simultaneously capture the relational information across different domains with both hierarchical and domain meta-knowledges. To achieve this goal, we introduce a hierarchical compare-aggregate mechanism to dynamically identify more representative prototypes for each domain, and construct a set of domain-relational graphs to generate re-weighting KD ratios. The overview of HRKD is shown in Figure 1. We first introduce the basic multi-domain KD method in Section 2.1, which is a naive framework lacking the ability of capturing cross-domain relations. Then we describe the domain-relational graph and compare-aggregate mechanism in Section 2.2 and 2.3, respectively, which are the primary

modules of our HRKD method to discover the relational information.

2.1 Multi-domain Knowledge Distillation

Similar to (Jiao et al., 2020), we jointly distill the embeddings, attention matrices, transformer layer outputs, and predicted logits between the teacher and student models. Inspired by (Liu et al., 2019c), we use a multi-task training strategy to perform multi-domain KD. Specifically, we share the weights of the embedding and transformer layers for all domains while assigning different prediction layers to different domains. Innovatively, we optimize models in different domains simultaneously rather than sequentially.

In detail, the embedding loss $\mathcal{L}_{embd}^{d}$ and prediction loss $\mathcal{L}_{pred}^{d}$ of d-th domain are formulated as:

$$\mathcal{L}_{embd}^{d} = \text{MSE}(\mathbf{E}^S \mathbf{W}^{embd}, \mathbf{E}_d^T), \quad (1)$$

$$\mathcal{L}_{pred}^{d} = \text{CE}(\mathbf{z}_d^S/t, \mathbf{z}_d^T/t), \quad (2)$$

where MSE and CE represent the mean square loss and cross-entropy loss, respectively. $\mathbf{E}^S$ and $\mathbf{E}_d^T$ represent the embeddings of student model and teacher model of d-th domain, respectively. $\mathbf{z}_d^S$ and $\mathbf{z}_d^T$ represent the predicted logits of student model and teacher model of d-th domain, respectively. $\mathbf{W}^{embd}$ is a learnable transformation matrix to align the student embedding dimension that mismatches with the teacher embedding dimension, and t is the temperature factor.

The attention loss $\mathcal{L}_{attn}^{m,d}$ and transformer layer output loss $\mathcal{L}_{hidn}^{m,d}$ at m-th student layer and d-th domain are formulated as:

$$\mathcal{L}_{attn}^{m,d} = \frac{1}{h} \sum_{i=1}^{h} \text{MSE}(\mathbf{A}_{i,m}^S, \mathbf{A}_{i,n,d}^T), \quad (3)$$

$$\mathcal{L}_{hidn}^{m,d} = \text{MSE}(\mathbf{H}_m^S \mathbf{W}_m^{hidn}, \mathbf{H}_{n,d}^T), \quad (4)$$

where h is the number of attention heads, $\mathbf{A}_{i,m}^S$ and $\mathbf{A}_{i,n,d}^T$ are the i-th head of attention matrices at m-th student layer and its matching n-th teacher layer of d-th domain, respectively. $\mathbf{H}_m^S$ and $\mathbf{H}_{n,d}^T$ are the transformer layer outputs at m-th student layer and n-th teacher layer of d-th domain, respectively. $\mathbf{W}_m^{hidn}$ is a transformation matrix to align the m-th layer of student output dimension that mismatches with the n-th layer of teacher output dimension. We use uniform strategy to match the layers between the student and teacher models.
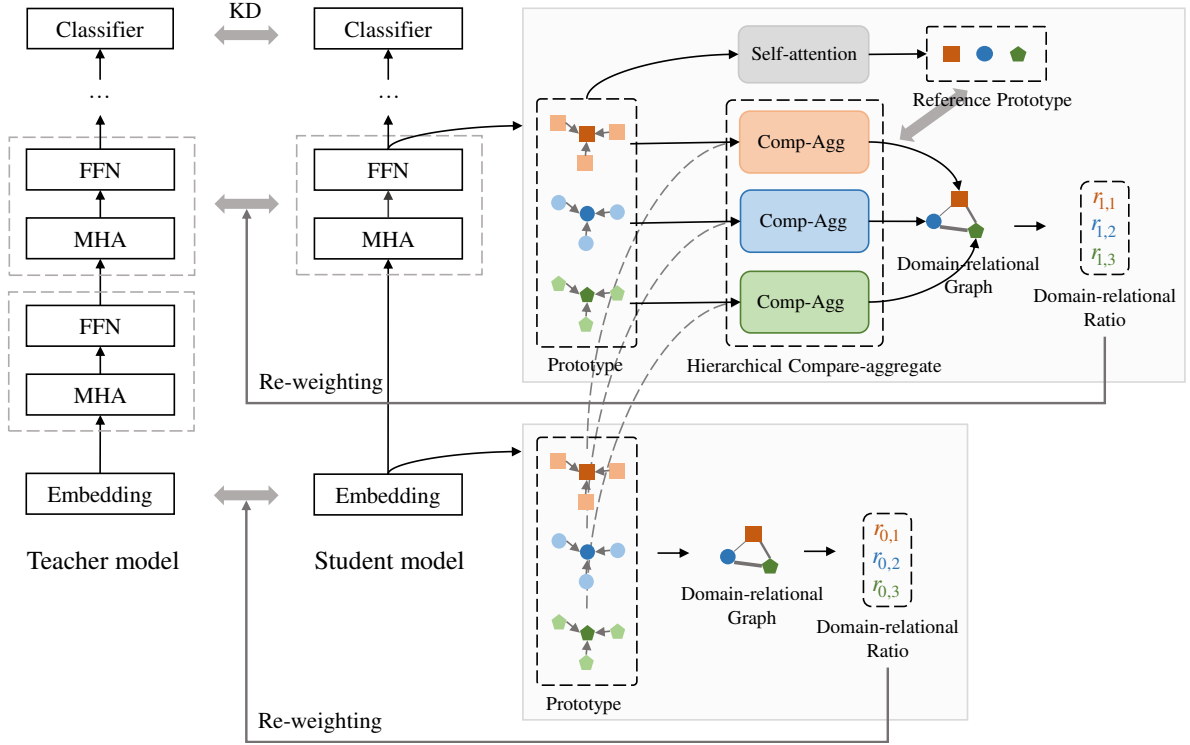
Figure 1: An overview of the proposed HRKD method. We use knowledge distillation (KD) to transfer the knowledge from the teacher model to the student model. During KD, we set up several domain-relational graphs to generate domain-relational ratios for re-weighting each domain. We then introduce a hierarchical compare-aggregate mechanism. The prototypes of different layers are dynamically aggregated based on the similarity ratios compared with the corresponding reference prototypes, which are then fed into the domain-relational graphs.

Finally, the overall KD loss is formulated as:

$$\mathcal{L}_{total} = \sum_{d=1}^{D} \left(L_{embd}^{d} + \sum_{m=1}^{M} (\mathcal{L}_{attn}^{m,d} + \mathcal{L}_{hidn}^{m,d}) + \gamma \mathcal{L}_{pred}^{d} \right),$$

(5)

where D is the total domain number, M is the number of transformer layers in the student model, γ is used to control the weight of the prediction loss $\mathcal{L}_{pred}$.

2.2 Prototype-based Domain-relational Graph

Although the basic multi-domain KD method described in Section 2.1 can distill the student models across different domains, the relational information between different domains is neglected, which is important for enhancing the model transferability as pointed out by previous studies (Finn et al., 2017; Javed and White, 2019). To solve the problem, we attempt to leverage meta-learning to enhance the performance and transferability of our student model. Inspired by the metric-based methods of meta-learning (Snell et al., 2017; Sung et al., 2018), we use prototype representations rather than raw samples to reflect the characteristics of each domain data. This helps to alleviate the negative impact of abnormal samples when there are few training samples (e.g., overfitting) and make the meta-learner easier to learn transferable cross-domain knowledge. Moreover, since we conduct KD over all of the student layers, we calculate different prototypes for different student layers to explicitly distinguish their characteristics. Specifically, the prototype $\mathbf{h}_{m,d}$ of m-th layer of the student model at d-th domain is calculated by:

$$\mathbf{h}_{m,d} = \begin{cases} \frac{1}{|\mathcal{D}_d|L} \sum_{i=1}^{|\mathcal{D}_d|} \sum_{l=1}^{L} \mathbf{E}_{i,l}^{S}, & m = 0 \\ \frac{1}{|\mathcal{D}_d|L} \sum_{i=1}^{|\mathcal{D}_d|} \sum_{l=1}^{L} \mathbf{H}_{m,i,l}^{S}, & 1 \le m \le M \end{cases}$$

(6)

where $\mathcal{D}_d$ refers to the training set of d-th domain, L refers to the sentence length (i.e., number of tokens), $\mathbf{E}_{i,l}^{S}$ represents l-th token of i-th sampled student embedding in $\mathcal{D}_d$, and $\mathbf{H}_{m,i,l}^{S}$ represents the l-th token output by the i-th sampled student transformer layer of the m-th student layer in $\mathcal{D}_d$. In practice, we calculate different prototypes for

different batches of training samples.

Afterward, these domain prototypes are leveraged to probe the relations across different domains. Although many multi-domain text mining methods have been proposed recently (Wang et al., 2020; Pan et al., 2020), they capture the relations separately for each given domain, which might be inconvenient and time-consuming in practice. Meanwhile, the learning process is not effective enough since the other domains cannot learn from each other when optimizing a specific domain. To solve this problem, we aim to simultaneously discover the cross-domain relations to make our framework more convenient and effective. To achieve the goal, we propose to use the graph attention network (GAT) (Veličković et al., 2018) to process the prototypes of all domains at the same time. To utilize GAT, each node in the graph represents a domain prototype, and each edge weight represents the similarity of the connected two prototypes. In this way, the relations across different domains can be captured simultaneously. In detail, we set up a two-layer *domain-relational graph* for each layer of the student model (except for the prediction layer). The input $\mathbf{h}_m$ of the m-th graph is a set of node features containing all of the domain prototypes at m-th student layer, i.e., $\mathbf{h}_m = \{\mathbf{h}_{m,1}, ..., \mathbf{h}_{m,D}\} \in \mathcal{R}^{D \times F}$, where D is the total domain number, F is the channel number of each prototype.

In the first-layer domain-relational graph of the m-th student layer, a shared weight matrix $\mathbf{W}_m \in \mathcal{R}^{F' \times F}$ is first applied to each node followed by a self-attention mechanism, where F' is the intermediate channel number. Then a multi-head concatenation mechanism with K heads is employed to stabilize the training process. Specifically, each input prototype $\mathbf{h}_{m,d}$ is first transformed by the weight matrix $\mathbf{W}_m$, then the attention coefficient $\alpha_{i,j,m}$ between two nodes i, j is calculated by applying a weight vector $\mathbf{a}_m \in \mathcal{R}^{2F' \times 1}$ to the concatenation of their transformed features followed by the LeakyReLU nonlinearity and softmax function, which can be formulated as:

$$s_{i,j,m} = \mathbf{a}_m^\top \left[\mathbf{W}_m \mathbf{h}_{m,i} \oplus \mathbf{W_m} \mathbf{h}_{m,j} \right], \quad (7)$$

$$\begin{aligned} \alpha_{i,j,m} &= \text{softmax} \left(\text{LeakyReLU}(s_{i,j,m}) \right) \\ &= \frac{\exp \left(\text{LeakyReLU}(s_{i,j,m}) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left(\text{LeakyReLU}(s_{i,k,m}) \right)}, \end{aligned}$$
$$(8)$$

where $\oplus$ represents the concatenation operation and $\mathcal{N}_i$ is all the first-order neighbors of node i (includ-

ing node i). Then the final output $\mathbf{h}'_{m,i} \in \mathcal{R}^{KF'}$ of node i can be obtained by the weighted sum of the transformed features of node i and its neighbors based on their attention coefficients followed by the ELU nonlinearity and a multi-head concatenation mechanism:

$$\mathbf{h}'_{m,i} = \oplus_{k=1}^K \text{ELU}(\sum_{j \in \mathcal{N}_i} \alpha_{i,j,m}^k \mathbf{W}_m^k \mathbf{h}_{m,j}), \quad (9)$$

where k represents the head index.

In the second-layer domain-relational graph of the m-th student layer, targeting at obtaining domain-relational ratios, we reformulate the parameters $\mathbf{W}_m, \mathbf{a}_m$ used in the first-layer graph as $\mathbf{W}'_m \in \mathcal{R}^{1 \times KF'}, \mathbf{a}'_m \in \mathcal{R}^{2 \times 1}$ respectively and do not apply the multi-head mechanism. We use the softmax operation to normalize the output and finally derive the domain-relational ratios $r_m \in \mathcal{R}^D$, formulated as below:

$$r_m = \text{softmax}(\text{ELU}(\sum_{j \in \mathcal{N}_i} \alpha'_{i,j,m} \mathbf{W}'_m \mathbf{h}'_{m,j})),$$
$$(10)$$

where $\alpha'_{i,j,m}$ is calculated by:

$$s'_{i,j,m} = \mathbf{a}_m'^\top \left[\mathbf{W}'_m \mathbf{h}'_{m,i} \oplus \mathbf{W}'_\mathbf{m} \mathbf{h}'_{m,j} \right], \quad (11)$$

$$\alpha'_{i,j,m} = \text{softmax} \left(\text{LeakyReLU}(s'_{i,j,m}) \right). \quad (12)$$

2.3 Hierarchical Compare-aggregate Mechanism

As different domains might have different preferences towards different layer prototypes, we propose a *hierarchical compare-aggregate mechanism* to dynamically select the most representative prototype for each domain. Our compare-aggregate mechanism is motivated by the Riesz representation theorem (Hartig, 1983), which indicates that an element can be evaluated by comparing it with a specific reference element and the quality of the element is the same as that of the selected reference element. Based on this, we establish a set of *reference prototypes* for each domain and hierarchically aggregate the current and previous layer prototypes based on their similarities with the corresponding reference prototypes.

Reference prototype. For each student layer, a simple way is to use the original domain prototypes of current layer as the reference prototypes for the current and previous layer prototypes. However, the information of other domains is not integrated,

which plays an important role to enhance the model transferability across different domains. To handle this, we introduce a self-attention mechanism over all of the domain prototypes in the same layer to inject the information of different domains. Specifically, the reference prototype $\mathbf{RP}_m \in \mathcal{R}^{D \times F}$ of m-th student layer is calculated by:

$$\mathbf{RP}_m = \boldsymbol{\alpha}_m^{\mathcal{D}} \cdot \mathbf{h}_m, \tag{13}$$

$$\boldsymbol{\alpha}_m^{\mathcal{D}} = \mathrm{softmax}(\mathbf{h}_m \cdot \mathbf{W}_m^{\mathcal{D}} \cdot \mathbf{h}_m^{\top}), \tag{14}$$

where $\boldsymbol{\alpha}_m^{\mathcal{D}} \in \mathcal{R}^{D \times D}$ refers to the attention matrix of m-th layer, $\mathbf{h}_m \in \mathcal{R}^{D \times F}$ refers to the prototypes of all domains at m-th layer, $\mathbf{W}_m^{\mathcal{D}} \in \mathcal{R}^{F \times F}$ refers to a learnable parameter matrix at m-th layer, and the softmax operation is performed over the last vector dimension.

Compare-aggregate mechanism. After obtaining the reference prototypes, we propose a compare-aggregate mechanism to hierarchically aggregate the layer prototypes by comparing them with the corresponding reference prototypes, which makes the model be aware to more representative layer prototypes for each domain. In detail, the aggregated prototype $\mathbf{AP}_{m,d} \in \mathcal{R}^F$ of m-th layer and d-th domain is formulated as:

$$\mathbf{AP}_{m,d} = \boldsymbol{\alpha}_{m,d}^{\mathcal{H}} \cdot \mathbf{h}_{\leq m,d}, \tag{15}$$

$$\boldsymbol{\alpha}_{m,d}^{\mathcal{H}} = \mathrm{softmax}(\mathbf{h}_{\leq m,d} \cdot \mathbf{W}_{m,d}^{\mathcal{H}} \cdot \mathbf{RP}_{m,d}), \tag{16}$$

where $\boldsymbol{\alpha}_{m,d}^{\mathcal{H}} \in \mathcal{R}^{m+1}$ represents the similarity ratios of m-th layer and d-th domain, $\mathbf{h}_{\leq m,d} \in \mathcal{R}^{(m+1) \times F}$ represents the prototypes of m-th layer and its previous layers at d-th domain, $\mathbf{W}_{m,d}^{\mathcal{H}} \in \mathcal{R}^{F \times F}$ is a learnable parameter matrix of m-th layer and d-th domain, and $\mathbf{RP}_{m,d} \in \mathcal{R}^F$ is the reference prototype of m-th layer and d-th domain. Then the aggregated prototype $\mathbf{AP}$ is sent to the domain-relational graphs to obtain the domain-relational ratios $r \in \mathcal{R}^{(M+1) \times D}$, as formulated by Equation (7)-(11).

Finally, the overall loss of our HRKD can be represented as:

$$\mathcal{L}_{total} = \sum_{d=1}^{D} \left(r_{0,d} \mathcal{L}_{embd}^d + \sum_{m=1}^{M} r_{m,d}(\mathcal{L}_{attn}^{m,d} + \mathcal{L}_{hidn}^{m,d}) \right.$$
$$\left. + \frac{\gamma}{D} \mathcal{L}_{pred}^d \right), \tag{17}$$

where $r_{m,d}$ is the domain-relational ratio at m-th student layer and d-th domain.

Table 1: Statistics of the MNLI and Reviews datasets.

Dataset	Domain	#Train	#Dev	#Test
MNLI	Fiction	69,613	7,735	1,973
	Government	69,615	7,735	1,945
	Slate	69,575	7,731	1,955
	Telephone	75,013	8,335	1,966
	Travel	69,615	7,735	1,976
Amazon Reviews	Books	1,631	170	199
	DVD	1,621	194	185
	Electronics	1,615	172	213
	Kitchen	1,613	184	203

3 Experiment

In this section, we conduct extensive experiments on two multi-domain datasets, namely MNLI and Amazon Reviews, to demonstrate the effectiveness of our HRKD method.

3.1 Datasets and Model Settings

We evaluate our method on two multi-domain datasets, including the multi-genre natural language inference (MNLI) dataset (Williams et al., 2018) and the Amazon Reviews dataset (Blitzer et al., 2007). In detail, MNLI is a natural language inference dataset with five domains for the task of entailment relation prediction between two sentences. In our setting, we randomly sample 10% of the original training data as our development set and use the original development set as our test set. Amazon Reviews is a sentiment analysis dataset with four domains for predicting whether the reviews are positive or negative. Following Pan et al. (2020), we randomly split the original data into train, development, and test sets. The statistics of these two datasets are listed in Table 1.

We use BERT$_{\mathrm{B}}$ (the number of layers N=12, the hidden size d'=768, the FFN intermediate hidden size d_i'=3072, the number of attention heads h=12, the number of parameters #params=109M) as the architecture of our teacher model, and BERT$_{\mathrm{S}}$ (M=4, d'=312, d_i'=1200, h=12, #params=14.5M) as our student model. Our teacher model HRKD-teacher is trained in a multi-domain manner as described in Section 2.1, and our student model BERT$_{\mathrm{S}}$ is initialized with the general distillation weights of TinyBERT[1].

Table 2: Results on MNLI in terms of accuracy (%) with standard deviations. $X \xrightarrow{A} Y$ denotes using teacher X to distill student Y with KD method of A. The bold and underlined numbers indicate the best and the second-best performance, respectively.

Method	Fiction	Government	Slate	Telephone	Travel	Average
BERT$_B$-single	82.2	84.2	76.7	82.4	84.2	81.9
BERT$_B$-mix	84.8	87.2	80.5	83.8	85.5	84.4
BERT$_B$-mtl	83.7	87.1	80.6	83.9	85.8	84.2
Meta-teacher	85.1	86.5	81.0	83.9	85.5	84.4
HRKD-teacher	83.8	87.6	80.4	83.5	85.4	84.2
BERT$_B$-single $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	78.8	83.2	73.6	78.8	81.9	79.3
BERT$_B$-mix $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	79.6	83.3	74.8	79.0	81.5	79.6
BERT$_B$-mtl $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	79.7	83.1	74.2	79.3	82.0	79.7
Meta-teacher $\xrightarrow{\text{Meta-distillation}}$ BERT$_S$	**80.5**	83.7	75.0	<u>80.5</u>	<u>82.1</u>	<u>80.4</u>
HRKD-teacher $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	80.1±0.22	<u>84.2±0.20</u>	<u>75.7±0.27</u>	80.0±0.23	81.9±0.17	<u>80.4</u>
HRKD-teacher $\xrightarrow{\text{HRKD}}$ BERT$_S$	<u>80.4±0.33</u>	**84.3±0.30**	**76.1±0.32**	**81.4±0.29**	**82.2±0.26**	**80.9**

Table 3: Results on Amazon Reviews in terms of accuracy (%) with standard deviations.

Method	Books	DVD	Electronics	Kitchen	Average
BERT$_B$-single	87.9	83.8	89.2	90.6	87.9
BERT$_B$-mix	89.9	85.9	90.1	92.1	89.5
BERT$_B$-mtl	90.5	86.5	91.1	91.1	89.8
Meta-teacher	92.5	87.0	91.1	89.2	89.9
HRKD-teacher	88.4	89.2	92.5	91.1	90.3
BERT$_B$-single $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	83.4	83.2	89.2	<u>91.1</u>	86.7
BERT$_B$-mix $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	88.4	81.6	89.7	89.7	87.3
BERT$_B$-mtl $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	<u>90.5</u>	81.6	88.7	90.1	87.7
Meta-teacher $\xrightarrow{\text{Meta-distillation}}$ BERT$_S$	**91.5**	86.5	90.1	89.7	<u>89.4</u>
HRKD-teacher $\xrightarrow{\text{TinyBERT-KD}}$ BERT$_S$	84.6±0.93	<u>87.8±0.55</u>	<u>91.3±0.23</u>	88.1±2.98	87.9
HRKD-teacher $\xrightarrow{\text{HRKD}}$ BERT$_S$	87.4±0.90	**90.5±1.76**	**91.8±1.25**	**92.2±0.48**	**90.5**

3.2 Baselines

We mainly compare our KD method with several KD baseline methods distilled from four teacher models, including BERT$_B$-single, BERT$_B$-mix, BERT$_B$-mtl, and Meta-teacher in Meta-KD (Pan et al., 2020). Specifically, BERT$_B$-single trains the teacher model of each domain separately with the single-domain dataset; BERT$_B$-mix trains a single teacher model with the combined dataset of all domains; BERT$_B$-mtl adopts the multi-task training method proposed by Liu et al. (2019c) to train the teacher model; Meta-teacher trains the teacher model with several meta-learning strategies including prototype-based instance weighting and domain corruption.

3.3 Implementation Details

For the teacher model, we train the HRKD-teacher for three epochs with a learning rate of 5e-5. For the student model, we train it for ten epochs with a learning rate of 1e-3 and 5e-4 on MNLI and Amazon Reviews, respectively. γ is set to 1, and t is 1. For few-shot learning, the learning rate for the student model is 5e-5, while other hyper-parameters are kept the same. The few-shot training data is selected from the front of our original training set with different sample ratios, while the dev and test data are the same as our original dev and test sets without sampling to make a fair comparison. In all the experiments, the sequence length is set to 128, and the batch size is 32. The hyper-parameters are tuned on the development set, and the results are averaged over five runs. Our experiments are conducted on 4 GeForce RTX 3090 GPUs.

3.4 General Experimental Results

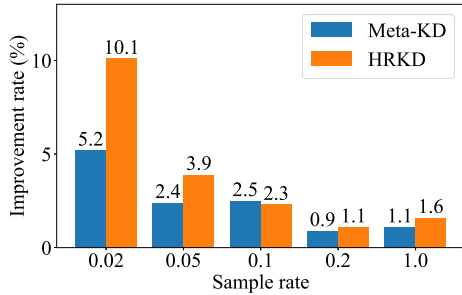The experimental results of our method are shown in Table 2 and 3. On the MNLI dataset, our

Figure 2: Comparison results of few-shot learning between Meta-KD and HRKD.

Table 4: Ablation studies on Amazon Reviews dataset. The accuracy values (%) with standard deviations are reported.

KD Method	Books	DVD	Elec.	Kitchen	Average
HRKD	**87.4**	90.5	91.8	**92.2**	**90.5**
- Self-attention	87.3	89.9	**92.5**	91.6	90.3
- Comp-Agg	86.3	**90.6**	90.9	91.8	89.9
- Hierarchical Rel.	85.9	89.6	91.1	91.6	89.5
- Domain Rel.	84.6	87.8	91.3	88.1	87.9

teacher model HRKD-teacher has similar performances with other baseline teacher models, but the performance of the student model distilled with the HRKD method (HRKD-teacher $\xrightarrow{\text{HRKD}}$ BERT_S) is significantly better than the base TinyBERT-KD method (HRKD-teacher $\xrightarrow{\text{TinyBERT-KD}}$ BERT_S) as well as its counterpart Meta-KD (Meta-teacher $\xrightarrow{\text{Meta-distillation}}$ BERT_S), which demonstrate the superior performance of our method. Specifically, with the HRKD method, the average score of the student model is both 0.5% higher than that of the model with the base TinyBERT-KD method and its counterpart Meta-KD method (see Table 2). It can also be observed that the improvement of our HRKD method on the Telephone domain is the most significant, which is probably caused by the amount of training data. From Table 1, we can see that the Telephone domain has much more training data than other domains, indicating that the Telephone domain can derive more relationship information from other domains and lead to higher improvement. Meanwhile, as shown in the results on the Amazon Reviews dataset in Table 3, the performance of the HRKD-teacher model is slightly better than that of other teacher models, but the student model distilled by the HRKD method largely outperforms the models distilled by the TinyBERT-KD and Meta-KD methods with average gains of 2.6% and 1.1% respectively, which prove the excellent performance of our method again. Note that our HRKD method significantly outperforms the base TinyBERT-KD method on both MNLI and Amazon Reviews datasets (t-test with p < 0.1). And since the performances of the Meta-teacher and our HRKD-teacher are similar on both datasets, the impact of the teacher is negligible, making the comparison between our HRKD and its counterpart Meta-KD relatively fair.

3.5 Few-shot Learning Results

As a large amount of training data is hard to collect in reality, the few-shot learning ability of our method is worth being evaluated, where both the teacher and student models are trained with few training data in each domain. We randomly sample a part of the training data in the MNLI dataset to make an evaluation, where the chosen sample rates are 2%, 5%, 10%, and 20%. We mainly compare the performance improvements between two methods: distilling from BERT_B-single to BERT_S with TinyBERT-KD (BERT_B-single $\xrightarrow{\text{TinyBERT-KD}}$ BERT_S) and our HRKD method (HRKD-teacher $\xrightarrow{\text{HRKD}}$ BERT_S). From the results in Figure 2, we can observe that the improvement gets more prominent when the training data gets fewer, and the average improvement rate is the largest of 10.1% when there is only 2% MNLI training data. In addition, we can see that the improvement rates of our method are higher than those of Meta-KD under most of the sample rates, especially when there are only 2% training data. These results demonstrate the strong learning ability of our HRKD method under the few-shot setting.

3.6 Ablation Studies

In this section, we progressively remove each module of our KD method to evaluate the effect of each module.

The results are shown in Table 4. We first remove the self-attention mechanism across different domain prototypes (- Self-attention), and the average score on Amazon Reviews drops by 0.2%, which proves its effectiveness. Next, we replace the hierarchical compare-aggregate mechanism with a simple average operation (- Comp-Agg), and the average score drops by 0.4%, which demonstrates the effectiveness of the compare-aggregate mechanism. Then we remove the hierarchical graph structure (- Hierarchical Rel.), where the input of each domain-relational graph comes from a single

Table 5: Case study on Amazon Reviews across four domains with three positive samples and one negative sample. Positive samples are colored in gray.

Domain	Label	Review Text
Books	POS	...leading, or molding young people today would benefit from reading this book...
DVD	POS	...The plot wasn't horrible, it was actually pretty good for a fright flick...
Electronics	NEG	...I returned the camera and bought a Panasonic and never looked back!
Kitchen	POS	This is great for making poached eggs on toast. My family has enjoyed using it...

Domain-relational Ratio	Hierarchical Similarity Ratio
[0.24, 0.29, 0.25, 0.23, 0.24]	[0.42, 0.58], [0.38, 0.30, 0.32], [0.24, 0.27, 0.25, 0.24], [0.16, 0.19, 0.21, 0.20, 0.24]
[0.24, 0.29, 0.25, 0.23, 0.24]	[0.52, 0.48], [0.32, 0.38, 0.30], [0.27, 0.24, 0.27, 0.22], [0.16, 0.17, 0.19, 0.24, 0.24]
[0.27, 0.17, 0.25, 0.25, 0.26]	[0.62, 0.38], [0.22, 0.21, 0.58], [0.19, 0.22, 0.30, 0.29], [0.14, 0.15, 0.24, 0.21, 0.26]
[0.24, 0.25, 0.25, 0.28, 0.27]	[0.46, 0.54], [0.30, 0.43, 0.28], [0.30, 0.25, 0.26, 0.20], [0.29, 0.19, 0.15, 0.12, 0.26]

Table 6: Case study on Amazon Reviews across four domains with two positive samples and two negative samples. Positive samples are colored in gray.

Domain	Label	Review Text
Books	NEG	...In this book, his "hard-evidence" is flimsey and suspicious...
DVD	POS	...If you have a child who loves John Deere, then this is a perfect DVD for them.
Electronics	POS	...Movies are amazing! My music collection never sounded so good...
Kitchen	NEG	This is the worst blender I've ever used...It's also loud and it moves a lot...

Domain-relational Ratio	Hierarchical Similarity Ratio
[0.26, 0.25, 0.25, 0.24, 0.27]	[0.43, 0.57], [0.40, 0.31, 0.29], [0.25, 0.26, 0.25, 0.23], [0.13, 0.17, 0.20, 0.21, 0.30]
[0.24, 0.25, 0.25, 0.26, 0.22]	[0.52, 0.48], [0.35, 0.32, 0.33], [0.22, 0.25, 0.27, 0.25], [0.18, 0.22, 0.18, 0.20, 0.22]
[0.24, 0.25, 0.25, 0.26, 0.25]	[0.54, 0.46], [0.31, 0.33, 0.36], [0.22, 0.24, 0.29, 0.25], [0.21, 0.20, 0.18, 0.18, 0.23]
[0.26, 0.25, 0.25, 0.24, 0.27]	[0.49, 0.51], [0.36, 0.33, 0.31], [0.26, 0.25, 0.26, 0.23], [0.22, 0.20, 0.19, 0.18, 0.21]

student layer. As can be seen, the average score drops by 0.4%, which proves the importance of the hierarchical relationship. Finally, we remove the domain-relational graph in each layer (- Domain Rel.), and the performance significantly drops by 1.6%, which strongly demonstrates the advantage of the domain relationship.

3.7 Case Studies

We further provide some case studies to intuitively explain the effectiveness of the domain-relational ratios and hierarchical similarity ratios calculated by our HRKD method (see Table 5 and 6).

In Table 5 and 6, we use the label to denote the categories of sampled domain examples, and we assume that if the learned domain-relational ratios and hierarchical similarity ratios are similar for domain examples with same category while different for those with different categories, then the model has relatively correctly captured the cross-domain and hierarchical relational information. We select two typical types of cases from Amazon Reviews across four domains, in which we adjust the number of domains in each category under two settings: (i) three same categories (i.e., POS) with another one category (i.e., NEG) as in Table 5, and (ii) two same categories (i.e., POS) with another two same categories (i.e., NEG) as in Table 6.

We find the results are intuitive, as we observe that the review texts with the same labels have similar domain-relational ratios and hierarchical similarity ratios, while different layers indeed have different domain weighting preferences and different preferences of layer prototypes for graph input. For example, in Table 5 and 6, positive samples tend to have higher domain-relational ratios in the middle layers (i.e., 2-4), while negative samples have higher ratios in the marginal layers (i.e., 1, 5). Meanwhile, in the second and third layers of Table 5 as well as the first layer of Table 6, lower positive layer prototypes tend to have higher similarity ratios, and the higher positive layer prototypes in the third layer of Table 6 also tend to have higher similarity ratios; while those of the negative layer prototypes are just the opposite. The results show that HRKD method has distinctively and correctly captured the hierarchical and domain meta-knowledges, leading to better performance.

4 Related Work

Pre-trained Language Model (PLM) Compression. Due to the large size and slow inference speed, PLMs are hard to be deployed on edge devices for practical usage. To solve this problem, many PLM compression methods have been proposed, including quantization (Shen et al., 2020), weight pruning (Michel et al., 2019), and knowledge distillation (KD) (Sun et al., 2019; Jiao et al., 2020). Among them, KD (Hinton et al., 2015) has been widely adopted due to its plug-and-play feasibility, aiming to distill the knowledge from a larger teacher model to a smaller student model without decreasing too much performance. For example, BERT-PKD (Sun et al., 2019) distills both intermediate and output layers on fine-tuning. TinyBERT (Jiao et al., 2020) additionally distills the embedding layer and attention matrices during pre-training and fine-tuning. Meta-KD (Pan et al., 2020) proposes to distill knowledge from a cross-domain meta-teacher through an instance-specific domain-expertise weighting technique.

In this paper, we propose a novel cross-domain KD framework that captures the relational information across different domains with both domain and hierarchical meta-knowledges, which has a better capability for capturing multi-domain correlations.

Transfer Learning and Meta-learning. Transfer learning focuses on transferring the knowledge from source domains to boost the model performance on the target domain. Among the methods in transfer learning, the shared-private architecture (Liu et al., 2017, 2019c) is most commonly applied in NLP tasks, which consists of a shared network to store domain-invariant knowledge and a private network to capture domain-specific information. There are also many works applying adversarial training strategies (Shen et al., 2018; Li et al., 2019; Zhou et al., 2019), which introduce domain adversarial classifiers to learn the domain-invariant features. Besides, the research of multi-domain learning has gained more and more attention recently, which is a particular case of transfer learning targeting transferring knowledge across different domains to comprehensively enhance the model performance (Cai and Wan, 2019; Wang et al., 2020). Unlike transfer learning, the goal of meta-learning is to train a meta-learner that can easily adapt to a new task with a few training data and iterations (Finn et al., 2017). Traditional meta-learning typically contains three categories of methods: metric-based (Snell et al., 2017; Sung et al., 2018), model-based (Santoro et al., 2016; Munkhdalai and Yu, 2017), and optimization-based (Ravi and Larochelle, 2017; Finn et al., 2017). In addition, the meta-learning technique can benefit the multi-domain learning task by learning the relationship information among different domains (Franceschi et al., 2017).

In this paper, we leverage meta-learning to solve the multi-domain learning task, where we consider cross-domain KD to simultaneously capture the correlation between different domains, aiming to train a better student meta-learner.

5 Conclusion

In this paper, we present a hierarchical relational knowledge distillation (HRKD) framework to simultaneously capture the cross-domain relational information. We build several domain-relational graphs to capture domain meta-knowledge and introduce a hierarchical compare-aggregate mechanism to capture hierarchical meta-knowledge. The learnt domain-relational ratios are leveraged to measure domain importance during the KD process. Extensive experiments on public datasets demonstrate the superior performance and solid few-shot learning ability of our HRKD method.

Acknowledgements

References

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

Yitao Cai and Xiaojun Wan. 2019. Multi-domain sentiment classification based on domain-aware embedding and attention. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 4904–4910.

Alexis CONNEAU and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. 2017. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1165–1173.

Donald G. Hartig. 1983. The riesz representation theorem revisited. *The American Mathematical Monthly*, pages 277–280.

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

Khurram Javed and Martha White. 2019. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Zheng Li, Xin Li, Ying Wei, Lidong Bing, Yu Zhang, and Qiang Yang. 2019. Transferable end-to-end aspect-based sentiment analysis with selective adversarial learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4590–4600.

Linqing Liu, Huan Wang, Jimmy Lin, Richard Socher, and Caiming Xiong. 2019a. Mkd: a multi-task knowledge distillation approach for pretrained language models. *arXiv preprint arXiv:1911.03588*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1–10.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019c. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3730–3740.

Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. 2015. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, pages 14–23.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2554–2563.

Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. 2020. Meta-kd: A meta knowledge distillation framework for language model compression across domains. *arXiv preprint arXiv:2012.01266*.

Shuke Peng, Feng Ji, Zehao Lin, Shaobo Cui, Haiqing Chen, and Yin Zhang. 2020. Mtss: Learn from multiple domain teachers and become a multi-domain dialogue expert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8608–8615.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1842–1850.

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4058–4065.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. *Proceedings of the AAAI*

Conference on Artificial Intelligence, pages 8815–8821.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4323–4332.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1199–1208.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Chengyu Wang, Minghui Qiu, Jun Huang, and Xiaofeng He. 2020. Meta fine-tuning neural language models for multi-domain text mining. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3094–3104.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1112–1122.

Ze Yang, Linjun Shou, Ming Gong, Wutao Lin, and Daxin Jiang. 2020. Model compression with two-stage multi-teacher knowledge distillation for web question answering system. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, page 690–698.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HI-BERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069.

Yinhe Zheng, Rongsheng Zhang, Minlie Huang, and Xiaoxi Mao. 2020. A pre-training based personalized dialogue generation model with persona-sparse data. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9693–9700.

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.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.