Unsupervised Domain Adaptation with Adapter

Rongsheng Zhang[1][*][†]**, Yinhe Zheng**[2,3][*]**, Xiaoxi Mao**[1]**, Minlie Huang**[2]

[1] Fuxi AI Lab, NetEase Inc., Hangzhou, China

[2] Department of Computer Science and Technology, Institute for Artifical Intelligence, State Key Lab of Intelligent Technology and Systems, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China.

[3] Alibaba Group, Beijing, China

`zhangrongsheng@corp.netease.com, zhengyinhe.zyh@alibaba-inc.com`

Abstract

Unsupervised domain adaptation (UDA) with pre-trained language models (PrLM) has achieved promising results since these pre-trained models embed generic knowledge learned from various domains. However, fine-tuning all the parameters of the PrLM on a small domain-specific corpus distort the learned generic knowledge, and it is also expensive to deployment a whole fine-tuned PrLM for each domain. This paper explores an adapter-based fine-tuning approach for unsupervised domain adaptation. Specifically, several trainable adapter modules are inserted in a PrLM, and the embedded generic knowledge is preserved by fixing the parameters of the original PrLM at fine-tuning. A domain-fusion scheme is introduced to train these adapters using a mix-domain corpus to better capture transferable features. Elaborated experiments on two benchmark datasets are carried out, and the results demonstrate that our approach is effective with different tasks, dataset sizes, and domain similarities.

1   Introduction

Unsupervised domain adaption (UDA) is an essential task in the realm of deep learning since it mitigates the expensive burden of manual annotation by focusing on cheap unlabeled data from target domains [Ramponi and Plank, 2020]. Among all existing approaches for UDA, pre-trained language model (PrLM) based approaches become the de-facto standard [Gururangan et al., 2020, Ben-David et al., 2020, Yu et al., 2021, Karouzos et al., 2021] since these PrLMs are equipped with generic knowledge learned from large corpora [Howard and Ruder, 2018] and lead to promising results.

The primary focuses of UDA methods are to capture the transferable features for the target domain while reserving the knowledge learned from the source domain [Blitzer et al., 2006, Pan et al., 2010]. However, most existing pre-training-based UDA approaches are carried out by fine-tuning the entire set of model parameters on domain-specific corpora [Gururangan et al., 2020, Yu et al., 2021, Karouzos et al., 2021], which are usually of limited sizes. Such a setting may easily drift the PrLM to a specified domain and distort the generic knowledge embedded in the original PrLM weights [Pfeiffer et al., 2020, He et al., 2021]. This hinders the model from capturing transferable features between different domains and leads to sub-optimal performance for UDA tasks [Karouzos et al., 2021]. Moreover, it is also expensive to fine-tune and deploy a large model for every single domain [Houlsby et al., 2019].

We observe that the intuition of preserving learned knowledge coincides with recently developed adapter-based tuning methods [Houlsby et al., 2019, Rebuffi et al., 2017], in which several trainable

[*]   Equal contribution.
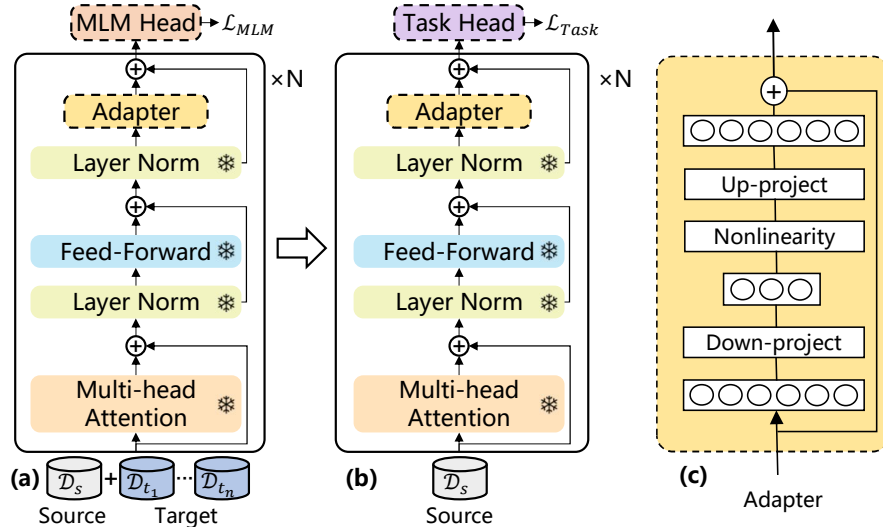[†]   Corresponding author.

Figure 1: UDA with the adapter. a) Domain-fusion training: train the adapter and MLM head using the MLM loss on a mixed corpus covering both source and target domains; b) Task fine-tuning: fine-tune the adapter and a task-specified head using the task specified loss on source domain samples. Note that all the parameters for the underlying pre-trained transformer model (i.e., sub-modules marked with ❅) are fixed. c) The architecture of the adapter module.

adapter modules are introduced between layers of a pre-trained language model (LM) while parameters from the original LM are fixed. This setting helps preserve the knowledge embedded in the PrLM and alleviates the distortion of features for different domains since the original PrLM remains intact [He et al., 2021, Pfeiffer et al., 2020,, Houlsby et al., 2019]. However, few studies are performed to extend this effective method to tackle UDA tasks.

In this paper, we explore to introduce adapter modules in pre-training-based UDA approaches. Specifically, several bottle-necked adapter modules are inserted in a transformer-based PrLM [Vaswani et al., 2017]. These adapters are learned following a two-step process: 1) The **domain-fusion training** step trains adapters with the Masked-Language-Model (MLM) loss [Devlin et al., 2019] on a mixed corpus containing data from both the source and all the target domains. This step facilitates the capture and fusion of transferable knowledge between different domains; 2) The **task fine-tuning** step fine-tunes adapters with the task-specific loss on the source domain corpus. Note that parameters of the underlying pre-trained LM are fixed throughout the two learning processes. This helps prevent the drifting of learned generic knowledge and facilitates more effective domain knowledge transferring [Pfeiffer et al., 2020]. In the testing phase, we apply the resulted model to data sampled from the target domain. The results on two benchmark datasets indicate that our method outperforms competitive baselines and is effective in improving the performance of downstream UDA tasks [3].

Our contributions can be summarized as:

1. We apply adapter modules in the pre-training-based UDA approaches. Specifically, trainable adapters are introduced in a PrLM, and a two-step process is introduced to facilitate the learning of these adapters.

2. Elaborated experiments on two benchmark datasets show that our approach outperforms competitive baselines and is more effective to improve the performance of downstream UDA tasks.

2 Related Work

Unsupervised Domain Adaption: Existing UDA approaches can be generally classified into two categories: 1) The *model-based* methods target at augmenting the feature spaces [Glorot et al., 2011, Chen et al., 2012, Ziser and Reichart, 2019, Ben-David et al., 2020], designing new losses [Ganin

---

[3]Our code is available in Appendix C

and Lempitsky, 2015, Ganin et al., 2016] or refining model structures [Bousmalis et al., 2016]; 2) The *data-based* methods aim to utilize pseudo-labels [Ruder and Plank, 2018, Lim et al., 2020] and develop better data selection schemes [Han and Eisenstein, 2019, Ma et al., 2019]. Some works also try to tackle UDA tasks utilizing large PrLM [Li et al., 2019, Gururangan et al., 2020, Yu et al., 2021, Karouzos et al., 2021], which are becoming the de-facto standard for various NLP tasks. Although promising results are reported, fine-tuning the whole model on a small amount of domain-specific data may distort underlying PrLM and lead to sub-optimal performances.

Adapters: In NLP studies, adapter modules are primarily used for parameter-efficient fine-tuning of large PrLMs [Lauscher et al., 2020, Wang et al., 2020, Lin et al., 2021, Poth et al., 2021, Han et al., 2021, Mahabadi et al., 2021]. The most similar works comparing to our study are the models for zero-shot cross-lingual transfer tasks [Pfeiffer et al., 2020, Vidoni et al., 2020]. However, these models aim to separate language-specific knowledge using adapters, while our UDA task tries to capture common and transferable features across different domains.

3 Method

3.1 Task Formulation

The UDA task investigated in this study aims to improve the model performance with the help of unlabeled data. Specifically, the training data consists of two parts: 1) labeled dataset $\mathcal{D}_s = \{(x_j^s, y_j)\}$ collected from a single source domain $\mathcal{S}$; 2) n unlabeled datasets $\mathcal{D}_{t_i} = \{x_j^{t_i}\}$ $(i = 1, ..., n)$ collected from n target domains $\mathcal{T}_i$ $(i = 1, ..., n)$. x_j^s and $x_j^{t_i}$ represents data drawn from $\mathcal{S}$ and $\mathcal{T}_i$, respectively, and $y_j \in \mathcal{Y}$ is the label associated with $x_j^{t_i}$, where $\mathcal{Y}$ is the label space. In the testing phase, we examine the effect of UDA on the labeled dataset $\bar{\mathcal{D}}_{t_i} = \{(\bar{x}_j^{t_i}, \bar{y}_j)\}$ collected for each target domain $\mathcal{T}_i$, in which $\bar{y}_j \in \mathcal{Y}$.

3.2 Adapters Architecture

Figure 1 shows an overview of our adapter-based UDA approach. Specifically, a transformer model is first initialized using a set of pre-trained weights, and a trainable adapter module is inserted into each transformer layer. Here we apply a variant of the efficient lightweight-adapter [Pfeiffer et al., 2020] that only adds one bottle-necked MLP after the feed-forward sublayer of each transformer block. The bottle-necked MLP first projects the H-dimensional input representations into a smaller dimension m $(m < H)$, then applies a nonlinearity (GELU [Hendrycks and Gimpel, 2016] in our case), and finally projects back to H dimensions. A residual connection is applied across the adapter.

3.3 Two-step Adaption

In the training process, we fix the parameters initialized from the PrLM and propose to learn the parameters of the randomly initialized adapter modules using a two-step process:

The first step is **domain-fusion training**, in which we mix the training instances from both the source and target domains, $\mathcal{D}_s \bigcup \mathcal{D}_{t_i} ... \bigcup \mathcal{D}_{t_n}$, and train the adapters with the MLM loss $\mathcal{L}_{MLM}$ on these instances. This setting enables the adapter to capture transferable features among all the domains, which enriches the knowledge embedded in the pre-trained model.

The second step, i.e., **task fine-tuning**, learns a task head using the task-specific loss $\mathcal{L}_{Task}$ (for example, the cross-entropy loss for classification tasks) on the labeled dataset $\mathcal{D}_s$. In this step, we also learn the adapters' parameters to allocate more modeling capacity to fit the task distribution. Note that different from the adaptive pre-training method [Gururangan et al., 2020] that directly learns the task head on labeled target domain data, our study follows the UDA setting that learns the task head on the source domain and tests it on the target domain. The transferable features captured in adapters help generalize the learned task knowledge and improve the performance of the resulting model on target domains.

4 Experiments

4.1 Dataset

Our UDA approach is evaluated on two benchmark datasets with different tasks:

1. **SDA:** Sentiment Domain Adaptation dataset [Blitzer et al., 2007] that contains Amazon product reviews for four different product types (i.e., domains): Books ($\mathcal{B}$), Dvds ($\mathcal{D}$), Electronics ($\mathcal{E}$) and Kitchen appliances ($\mathcal{K}$). 2.0K reviews with binary sentiment labels are available for each domain and we split these data into Train and Dev set with a ratio of 8:2. Four adaption schemes are attempted by regarding each of these four domains as the source domain (src) and the rest domains as the target domain (tgt). Test set of each domain contains 1.6K labeled reviews with binary sentiment labels.

2. **XNLI:** Cross-lingual Natural Language Inference dataset [Conneau et al., 2018] that focuses on the NLI three-way classification task. This dataset involves texts with 15 languages, and the Train, Dev, and Test set of each language contain 392.70K, 2.49K, and 5.01K samples, respectively. In this study, we regard each language as a domain and only use English as the source domain.

4.2 Implementation Details

Model and Training: For experiments on the SDA and XNLI dataset, the pre-trained RoBERTa$_{base}$ [Liu et al., 2019] and XLM-R$_{base}$ [Conneau et al., 2020] model is used as the initialization PrLM, respectively, and the adapter size m is set to 128, and 256, respectively. The Adam optimizer [Kingma and Ba, 2014] is used with a learning rate of 5e-5. The domain-fusion training process lasts for 10 epochs and the best models are selected based on the Dev set performance. See Appendix A for more training details.

Baselines: Three baselines are tested in our study. The first two baselines do not utilize the adapter module: 1) **Full-FT** Fine-Tunes all transformer parameters using $\mathcal{L}_{Task}$ on $\mathcal{D}_S$. The domain-fusion training is not performed in this baseline; 2) **Full-TSA** adjusts the work of [Gururangan et al., 2020] to UDA tasks, i.e., the Two-Step Adaption process introduced in Section 3.3 is applied on all transformer parameters. The third baseline in our study, 3) **Ada-FT**, incorporates the adapter module but do not perform the domain-fusion training process. We denote our approach as **Ada-TSA**.

$src \rightarrow tgt$	Full-FT	Full-TSA	Ada-FT	Ada-TSA
$\mathcal{B} \rightarrow \mathcal{D}$	92.80 ± 0.30	92.80 ± 0.40	92.46 ± 0.42	$\mathbf{92.87} \pm 0.27$
$\mathcal{B} \rightarrow \mathcal{E}$	92.01 ± 0.67	$91.26^{\dagger} \pm 1.15$	$91.98^{\dagger} \pm 0.54$	$\mathbf{92.84} \pm 0.43$
$\mathcal{B} \rightarrow \mathcal{K}$	$93.70^{\dagger} \pm 0.09$	$\mathbf{94.36} \pm 0.59$	$92.91^{\dagger} \pm 0.88$	93.98 ± 0.16
$\mathcal{D} \rightarrow \mathcal{B}$	$92.25^{\dagger} \pm 0.81$	$\mathbf{93.74} \pm 0.66$	92.51 ± 1.65	93.30 ± 0.28
$\mathcal{D} \rightarrow \mathcal{E}$	$91.46^{\ddagger} \pm 0.40$	93.17 ± 0.54	$91.50^{\ddagger} \pm 0.47$	$\mathbf{93.72} \pm 0.17$
$\mathcal{D} \rightarrow \mathcal{K}$	$92.64^{\ddagger} \pm 0.68$	94.17 ± 0.20	93.41 ± 0.86	$\mathbf{94.39} \pm 0.32$
$\mathcal{E} \rightarrow \mathcal{B}$	$91.44^{\ddagger} \pm 0.26$	$91.41^{\ddagger} \pm 0.24$	$90.77^{\ddagger} \pm 0.31$	$\mathbf{92.25} \pm 0.29$
$\mathcal{E} \rightarrow \mathcal{D}$	$91.34^{\ddagger} \pm 0.12$	91.66 ± 0.29	91.57 ± 0.58	$\mathbf{92.15} \pm 0.33$
$\mathcal{E} \rightarrow \mathcal{K}$	$93.48^{\ddagger} \pm 0.19$	94.10 ± 0.16	$\mathbf{94.32} \pm 0.36$	94.14 ± 0.21
$\mathcal{K} \rightarrow \mathcal{B}$	$90.77^{\ddagger} \pm 0.21$	$91.89^{\ddagger} \pm 0.14$	$90.54^{\ddagger} \pm 0.45$	$\mathbf{93.20} \pm 0.17$
$\mathcal{K} \rightarrow \mathcal{D}$	$90.85^{\ddagger} \pm 0.31$	$92.54^{\ddagger} \pm 0.08$	$91.01^{\ddagger} \pm 0.51$	$\mathbf{93.26} \pm 0.07$
$\mathcal{K} \rightarrow \mathcal{E}$	$92.54^{\ddagger} \pm 0.22$	$\mathbf{93.65} \pm 0.13$	93.09 ± 0.81	93.63 ± 0.16
Avg.	$92.10^{\ddagger}$	$92.89^{\dagger}$	$92.17^{\ddagger}$	$\mathbf{93.31}$

Table 1: Sentiment classification accuracy and standard deviations of five independent runs on SDA. $\dagger$ and $\ddagger$ indicates significant difference (t-test) between our approach and the baseline with p-value < 0.05 and 0.01, respectively.

4.3 Results and Discussion

UDA Results: The results of our approach and baselines are summarized in Table 1 (for the sentiment classification task on SDA) and Table 2 (for the NLI task on XNLI). All results are averaged over five runs with different random seeds. Our approach achieves the best results in most adaptation settings.

tgt	Full-FT	Full-TSA	Ada-FT	Ada-TSA
en	$83.16^{\ddagger} \pm 0.30$	$83.14^{\ddagger} \pm 0.29$	83.98 ± 0.39	$\mathbf{84.57} \pm 0.41$
fr	$76.57^{\ddagger} \pm 0.52$	$76.22^{\ddagger} \pm 0.21$	77.11 ± 0.72	$\mathbf{77.91} \pm 0.51$
es	$77.24^{\dagger} \pm 0.39$	77.87 ± 0.32	78.11 ± 0.37	$\mathbf{78.17} \pm 0.58$
de	$75.06^{\dagger} \pm 0.32$	75.96 ± 0.44	$74.90^{\dagger} \pm 0.37$	$\mathbf{76.02} \pm 0.57$
el	$73.61^{\ddagger} \pm 0.25$	$74.63^{\ddagger} \pm 0.29$	74.40 ± 0.72	$\mathbf{75.21} \pm 0.34$
bg	$76.25^{\ddagger} \pm 0.13$	76.86 ± 0.42	77.30 ± 0.42	$\mathbf{77.43} \pm 0.46$
ru	$73.89^{\ddagger} \pm 0.21$	$74.45^{\dagger} \pm 0.39$	75.11 ± 0.30	$\mathbf{75.39} \pm 0.47$
tr	$70.57^{\ddagger} \pm 0.21$	$71.13^{\dagger} \pm 0.13$	71.96 ± 0.39	$\mathbf{71.97} \pm 0.56$
ar	$69.64^{\ddagger} \pm 0.17$	$70.22^{\ddagger} \pm 0.46$	$70.49^{\dagger} \pm 0.51$	$\mathbf{71.31} \pm 0.37$
vi	73.37 ± 0.50	73.59 ± 0.33	$\mathbf{74.15} \pm 0.53$	73.86 ± 0.49
th	70.33 ± 0.55	$71.20^{\ddagger} \pm 0.29$	$\mathbf{71.29}^{\dagger} \pm 0.79$	69.99 ± 0.62
zh	72.18 ± 0.20	$\mathbf{74.47}^{\ddagger} \pm 0.29$	72.48 ± 0.58	72.98 ± 0.51
hi	68.74 ± 0.52	69.36 ± 0.52	$\mathbf{69.36} \pm 0.37$	68.69 ± 0.62
sw	$60.98^{\ddagger} \pm 0.34$	$\mathbf{65.45} \pm 0.42$	$63.31^{\ddagger} \pm 0.21$	64.88 ± 0.54
ur	64.34 ± 0.22	$62.96^{\dagger} \pm 0.63$	$\mathbf{64.95} \pm 0.52$	64.15 ± 0.54
Avg.	$71.62^{\ddagger}$	72.45	72.49	$\mathbf{72.71}$

Table 2: NLI accuracy and standard deviations of five independent runs on XNLI dataset. $\dagger$ and $\ddagger$ carry the same means as in Table 1.

Moreover, Figure 2 shows the averaged performance of each model across all adaption settings. It can be seen that applying the adapter module and using the two-step adaption approach helps to improve the UDA performance. Specifically, for the averaged performance, our method yields an absolute improvement of 1.21% and 1.09% over the Full-FT baseline on the SDA and XNLI dataset, respectively. Results on more datasets in Appendix B also support this conclusion.
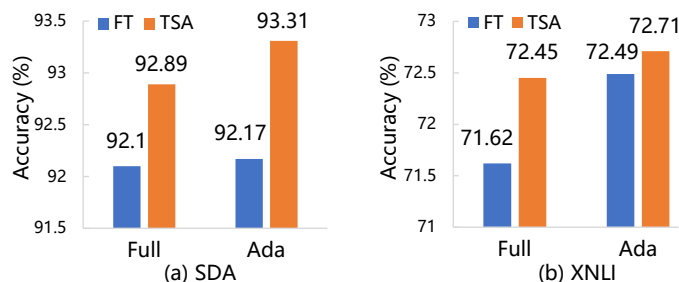

Figure 2: Averaged performance on both datasets.

Adapter for Different Data Sizes: Since the SDA and XNLI datasets have different sizes, the results reported in Table 1 and 2 also reveal the effect of data size on UDA performances:

1) When $\mathcal{D}_S$ is small, the adapter module is less effective without the domain-fusion step. Specifically, if we do not apply the domain-fusion training process, the adapter module only improves the averaged performance from 92.10% (Full-FT) to 92.17% (Ada-FT) on the relatively small dataset SDA (i.e., an absolute improvement of 0.07%). However, on the XNLI dataset, which contains more data, the performance gain brought by the adapter module is much larger, i.e., from 71.62% (Full-FT) to 72.49% (Ada-FT). This may be because a small $\mathcal{D}_S$ is not enough to properly learn the adapter module from random initialization.

2) Our two-step adaption process improves the effectiveness of the adapter on small $\mathcal{D}_S$. Specifically, when the two-step adaption process is applied, the adapter module becomes more effective on the SDA dataset that an absolute improvement of 0.42% (i.e., from 92.89% of Full-TSA to 93.31% of Ada-TSA) is observed. That is, our two-step adaption process upgrades the effectiveness of the adapter about 6 times on SDA.

Domain Fusion with Different Similarities: In this part, we study the effect of domain fusion training with different domain similarities. We first analyzed the similarities between each domain using the vocabulary overlaps [Gururangan et al., 2020]. Specifically, we first build the vocabulary

on the corpus of each domain. Then for any two domains, we calculate the overlap of their top 10k frequent vocabulary words as their domain similarity. The domain similarities of the SDA and XNLI dataset are separately shown in 3a and 3b. The results confirm our intuition that the gaps between each domain in XNLI are much larger than SDA. This conclusion makes sense because all texts in UDA are English Amazon reviews, while texts in XNLI are in different languages.

	Books	Dvd	Elec.	Kitc.
Books	100	56.4	39.0	39.2
Dvd	56.4	100	38.4	38.2
Elec.	39.0	38.4	100	41.5
Kitc.	39.2	38.2	41.5	100

(a) SDA

en	fr	es	de	el
100	17.4	9.3	12.6	2.5

bg	ru	tr	ar	vi
1.6	1.9	7.5	1.7	13.2

th	zh	hi	sw	ur
4.8	2.1	3.1	22.5	5.9

(b) XNLI

Figure 3: The domain similarity of the SDA and XNLI dataset. (a) SDA: each element of the matrix represents the vocabulary overlap of source (rows) and target domain (columns). (b) XNLI: the similarity between the English domain and all other domains.

(a1) No domain fusion   (a2) Domain fusion with full model   (a3) Domain fusion with adapter

(a) SDA dataset

(b1) No domain fusion   (b2) Domain fusion with full model   (b3) Domain fusion with adapter
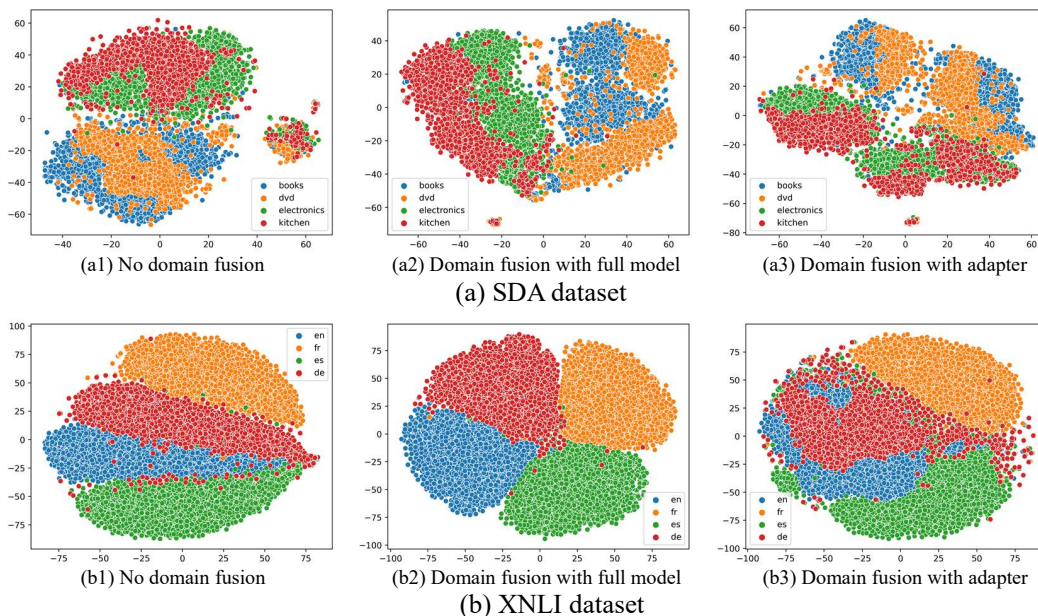
(b) XNLI dataset

Figure 4: Hidden representation of different domain fusion (DF) ways. For both SDA and XNLI (only show four languages for clarity), we plot the figures for three cases, (1) No domain fusion: the original pretrained model without DF. (2) Domain fusion with full model: perform DF on the original pretrained model. (3) Domain fusion with adapter: perform DF on the adapter-based model.

The results in Figure 2 suggest that our domain-fusion training step is more effective at improving the UDA performance when the data used in this step come from similar domains. Specifically, the domain-fusion process brings an absolute gain of 1.14% on the SDA dataset (i.e., from 92.17% of Ada-FT to 93.31% of Ada-TSA). In contrast, this gain drops to 0.22% on the XNLI dataset (i.e., from 72.49% of Ada-FT to 72.71% of Ada-TSA), which has larger domain gaps. To get a deeper insight into different ways of domain fusion training, we compute their hidden representation of the last layer and reduce the dimension using the t-SNE algorithm [Van der Maaten and Hinton, 2008], which are shown in Figure 4. We can see that the dataset with smaller domain similarity (XNLI) brings less changes in hidden representation than the one with larger domain similarity (SDA). And it is more effective to perform the domain fusion training with the adapter module than with the full model.

5 Conclusion

This paper presents an adapter-based fine-tuning approach for unsupervised domain adaptation. Trainable adapter modules are inserted in a pre-trained LM, and a two-step training process is introduced to learn the parameters of these adapters. We demonstrate our method on two datasets with different sizes and domain similarities. The results show that the introduced adapter modules facilitate the adaption process, and our two-step training approach helps to further improve the UDA performance. As future works, we will explore more effective architectures of the adapter module.

References

[1] E. Ben-David, C. Rabinovitz, and R. Reichart. Perl: Pivot-based domain adaptation for pre-trained deep contextualized embedding models. *Transactions of the Association for Computational Linguistics*, 8:504–521, 2020.

[2] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128, 2006.

[3] J. Blitzer, M. Dredze, and F. Pereira. 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, 2007.

[4] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, 2016.

[5] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

[6] A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, 2018.

[7] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/ 2020.acl-main.747. URL https://www.aclweb.org/anthology/2020.acl-main.747.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[9] Y. Ganin and V. S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[10] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[11] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

[12] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://www.aclweb.org/anthology/2020.acl-main.740.

[13] W. Han, B. Pang, and Y. Wu. Robust transfer learning with pretrained language models through adapters. *arXiv preprint arXiv:2108.02340*, 2021.

[14] X. Han and J. Eisenstein. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4229–4239, 2019.

[15] R. He, L. Liu, H. Ye, Q. Tan, B. Ding, L. Cheng, J.-W. Low, L. Bing, and L. Si. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*, 2021.

[16] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[17] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[18] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[19] C. Karouzos, G. Paraskevopoulos, and A. Potamianos. Udalm: Unsupervised domain adaptation through language modeling. *arXiv preprint arXiv:2104.07078*, 2021.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] A. Lauscher, O. Majewska, L. F. Ribeiro, I. Gurevych, N. Rozanov, and G. Glavaš. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, 2020.

[22] Z. Li, X. Peng, M. Zhang, R. Wang, and L. Si. Semi-supervised domain adaptation for dependency parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2386–2395, 2019.

[23] K. Lim, J. Y. Lee, J. Carbonell, and T. Poibeau. Semi-supervised learning on meta structure: Multi-task tagging and parsing in low-resource scenarios. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8344–8351, 2020.

[24] Z. Lin, A. Madotto, Y. Bang, and P. Fung. The adapter-bot: All-in-one controllable conversational model. 2021.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[26] X. Ma, P. Xu, Z. Wang, R. Nallapati, and B. Xiang. Domain adaptation with bert-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 76–83, 2019.

[27] R. K. Mahabadi, S. Ruder, M. Dehghani, and J. Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021.

[28] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760, 2010.

[29] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[30] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, 2020.

[31] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, 2020.

[32] C. Poth, J. Pfeiffer, A. Rücklé, and I. Gurevych. What to pre-train on? efficient intermediate task selection. *arXiv preprint arXiv:2104.08247*, 2021.

[33] A. Ramponi and B. Plank. Neural unsupervised domain adaptation in nlp—a survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6838–6855, 2020.

[34] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. *arXiv preprint arXiv:1705.08045*, 2017.

[35] S. Ruder and B. Plank. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054, 2018.

[36] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.

[38] M. Vidoni, I. Vulić, and G. Glavaš. Orthogonal language and task adapters in zero-shot cross-lingual transfer. *arXiv preprint arXiv:2012.06460*, 2020.

[39] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, C. Cao, D. Jiang, M. Zhou, et al. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*, 2020.

[40] T. Yu, Z. Liu, and P. Fung. Adaptsum: Towards low-resource domain adaptation for abstractive summarization. *arXiv preprint arXiv:2103.11332*, 2021.

[41] Y. Ziser and R. Reichart. Task refinement learning for improved accuracy and stability of unsupervised domain adaptation. In *proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 5895–5906, 2019.

A   Model and Training Details

In this appendix, we provide more details about the model and training configuration.

The hyper-parameter settings of different models and fine-tuning process on the SDA and XNLI dataset are provided in Table 3. In the domain-fusion training process, the corpus used to optimize the MLM loss contains data sampled from the source domain and all target domains. The learning rate of domain-fusion training is set to 2e-5 for the model with full-parameter update (i.e., Full-TSA) and 5e-5 for the model with adapter-parameter update (i.e., Ada-TSA). Additionally, the training of all models starts with a warm-up step of 1K, and the learning rate decreases linearly with the number of training steps. The Adam optimizer is used with the parameter $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All the models are trained on the GTX2080Ti GPU.

B   Supplementary Experiments

This appendix provides the experiment results on a larger multi-domain sentiment analysis dataset to support the analysis in our paper. Specifically, this dataset is collected from Kaggle[4], and it contains reviews from multiple sources (i.e., domains) including Amazon ($\mathcal{A}$), IMDB ($\mathcal{I}$), TripAdvisor ($\mathcal{T}$) and Yelp ($\mathcal{Y}$). We denote this dataset as **MSS** (Multi-Source Sentiment dataset). The statistic of MSS is summarized in Table 4 and the experimental results are shown in Table 5.

---

[4]https://www.kaggle.com

Dataset	Model	AdaS	LR	BZ	TrE	ToP	TrP	MaxL
SDA	Full-FT	-	2e-5	16	10	125M	125M	512
	Full-TSA	-	2e-5	16	10	125M	125M	512
	Ada-FT	128	5e-5	16	10	127M	2.9M	512
	Ada-TSA	128	5e-5	16	10	127M	2.9M	512
XNLI	Full-FT	-	2e-5	32	10	278M	278M	128
	Full-TSA	-	2e-5	32	10	278M	278M	128
	Ada-FT	256	5e-5	32	10	283M	5.3M	128
	Ada-TSA	256	5e-5	32	10	283M	5.3M	128

Table 3: Detailed configurations of the models and training process. Abbreviation description: LR: learning rate, BZ: batch size, TE: training epoch, ToP: total parameter, TrP: trainable parameter, AdaS: adapter size, MaxL: max sequence length.

Source	Train	Dev	Test
Amazon	20K	5K	5K
IMDB	20K	5K	5K
TripAdvisor	8K	1K	1K
Yelp	20K	5K	5K

Table 4: Statistics of the MSS dataset. Note that the Train, Dev, and Test set are all balanced.

$src \rightarrow tgt$	Full-FT	Full-TSA	Ada-FT	Ada-TSA
$\mathcal{A} \rightarrow \mathcal{I}$	93.30 ± 0.16	93.54 ± 0.15	$\mathbf{93.79} \pm 0.24$	93.72 ± 0.23
$\mathcal{A} \rightarrow \mathcal{T}$	82.72 ± 1.51	$\mathbf{86.58} \pm 0.65$	86.22 ± 1.79	86.10 ± 1.52
$\mathcal{A} \rightarrow \mathcal{Y}$	95.43 ± 0.51	96.14 ± 0.27	$\mathbf{96.25} \pm 0.15$	95.88 ± 0.42
$\mathcal{I} \rightarrow \mathcal{A}$	90.62 ± 1.30	89.24 ± 1.66	92.60 ± 0.27	$\mathbf{93.24} \pm 0.37$
$\mathcal{I} \rightarrow \mathcal{T}$	85.10 ± 1.50	82.28 ± 2.37	86.66 ± 0.15	$\mathbf{89.30} \pm 0.63$
$\mathcal{I} \rightarrow \mathcal{Y}$	92.82 ± 0.90	92.59 ± 1.03	95.19 ± 0.14	$\mathbf{95.48} \pm 0.09$
$\mathcal{T} \rightarrow \mathcal{A}$	92.47 ± 0.26	$\mathbf{93.71} \pm 0.12$	92.13 ± 0.30	92.10 ± 0.29
$\mathcal{T} \rightarrow \mathcal{I}$	88.64 ± 0.87	$\mathbf{89.64} \pm 0.06$	87.60 ± 0.81	88.81 ± 0.33
$\mathcal{T} \rightarrow \mathcal{Y}$	93.15 ± 0.40	$\mathbf{94.58} \pm 0.09$	94.57 ± 0.12	93.53 ± 0.39
$\mathcal{Y} \rightarrow \mathcal{A}$	94.50 ± 0.30	95.16 ± 0.07	95.00 ± 0.12	$\mathbf{95.40} \pm 0.10$
$\mathcal{Y} \rightarrow \mathcal{I}$	91.99 ± 0.07	$\mathbf{93.24} \pm 0.19$	92.49 ± 0.08	92.82 ± 0.11
$\mathcal{Y} \rightarrow \mathcal{T}$	86.36 ± 1.02	89.16 ± 0.54	85.20 ± 0.32	$\mathbf{89.24} \pm 0.53$
Avg.	90.59	91.32	91.47	**92.13**

Table 5: The accuracy on the MSS dataset and standard deviations of five independent runs.

Discussion: Experiments on the MSS dataset corroborate the discussion about the effect of data size and domain similarities in Section 4.3 of the main text:

1) Comparing the results of SDA and MSS, which have the same task (i.e., sentiment analysis) but different data sizes, a larger gain is obtained with the injection of the adapter module when the data size is larger. Specifically, the adapter module improves the average accuracy from 90.59% of Full-FT to 91.47% of Ada-FT on the MSS dataset (an absolute improvement of 0.88%) while only 92.10% of Full-FT to 92.17% of Ada-FT on the SDA dataset (an absolute improvement of 0.07%). This proves that more data facilitates to learn the parameters of the adapter module from random initialization.

2) Comparing the results of MSS and XNLI, which both have enough corpus for their corresponding task, the domain-fusion training is more effective on the MSS dataset than on the XNLI dataset. Specifically, the averaged accuracy improves from 91.47% of Ada-FT to 92.13% of Ada-TSA (the gain is 0.66%) on the MSS dataset. However, this performance improvement drops to 0.22% on the XNLI dataset. (from 72.49% of Ada-FT to 72.71% of Ada-TSA). This proves that the domain-fusion training brings a larger gain when the source and target domains are more similar.

C   Link of Our Code

Our experiments are implemented utilizing the Transformers[5] framework of Huggingface. To support the reproducibility of our work, we provide an anonymous download link [6] of our code.

---

[5]https://github.com/huggingface/transformers
[6]https://drive.google.com/file/d/1fHl4w1DmbRDon-3OxWsYz6AeQ6FmweWP/view?usp=sharing