Self-Adaptation for Unsupervised Domain Adaptation

Xia Cui and **Danushka Bollegala**
Department of Computer Science
University of Liverpool
Ashton Street, Liverpool L69 3BX, United Kingdom
{xia.cui, danushka.bollegala}@liverpool.ac.uk

Abstract

Lack of labelled data in the target domain for training is a common problem in domain adaptation. To overcome this problem, we propose a novel unsupervised domain adaptation method that combines projection and self-training based approaches. Using the labelled data from the source domain, we first learn a projection that maximises the distance among the nearest neighbours with opposite labels in the source domain. Next, we project the source domain labelled data using the learnt projection and train a classifier for the target class prediction. We then use the trained classifier to predict pseudo labels for the target domain unlabelled data. Finally, we learn a projection for the target domain as we did for the source domain using the pseudo-labelled target domain data, where we maximise the distance between nearest neighbours having opposite pseudo labels. Experiments on a standard benchmark dataset for domain adaptation show that the proposed method consistently outperforms numerous baselines and returns competitive results comparable to that of SOTA including self-training, tri-training, and neural adaptations.

1 Introduction

A machine learning model trained using data from one domain (source domain) might not necessarily perform well on a different (target) domain when their distributions are different. Domain adaptation (DA) considers the problem of adapting a machine learning model such as a classifier that is trained using a source domain to a target domain. In particular, in Unsupervised Domain Adaptation (UDA) (Blitzer et al., 2006, 2007; Pan et al., 2010) we do not assume the availability of any labelled instances from the target domain but a set of labelled instances from the source domain and unlabelled instances from both source and the target domains.

Two main approaches for UDA can be identified from prior work: **projection-based** and **self-training**.

Projection[1]-based methods for UDA learn an embedding space where the distribution of features in the source and the target domains become closer to each other than they were in the original feature spaces (Blitzer et al., 2006). For this purpose, the union of the source and target feature spaces is split into domain-independent (often referred to as *pivots*) and domain-specific features using heuristics such as mutual information or frequency of a feature in a domain. A projection is then learnt between those two feature spaces and used to adapt a classifier trained from the source domain labelled data. For example, methods based on different approaches such as graph-decomposition *spectral feature alignment* (Pan et al., 2010) or autoencoders (Louizos et al., 2015) have been proposed for this purpose.

Self-training (Yarowsky, 1995; Abney, 2007) is a technique to iteratively increase a set of labelled instances by training a classifier using current labelled instances and applying the trained classifier to predict pseudo-labels for unlabelled instances. High confident predictions are then appended to the current labelled dataset, thereby increasing the number of labelled instances. The process is iterated until no additional pseudo-labelled instances can be found. Self-training provides a direct solution to the lack of labelled data in the target do-

[1] we consider the terms *project* and *embed* as synonymous in this paper

Proceedings of Recent Advances in Natural Language Processing, pages 213–222,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_025

main in UDA (McClosky et al., 2006; Reichart and Rappoport, 2007; Drury et al., 2011). Specifically, the source domain's labelled instances are used to initialise the self-training process and during subsequent iterations labels are inferred for the target domain's unlabelled instances, which can be used to train a classifier for the task of interest.

So far in UDA projection-learning and self-trained approaches have been explored separately. An interesting research question we ask and answer positively in this paper is whether *can we improve the performance of projection-based methods in UDA using self-training?* In particular, recent work on UDA (Morerio et al., 2018) has shown that minimising the entropy of a classifier on its predictions in the source and target domains is equivalent to learning a projection space that maximises the correlation between source and target instances. Motivated by these developments, we propose **Self-Adapt**, a method that combines the complementary strengths of projection-based methods and self-training methods for UDA.

Our proposed method consists of three steps.

- First, using labelled instances from the source domain we learn a projection ($\mathcal{S}_{\text{prj}}$) that maximises the distance between each source domain labelled instance and its nearest neighbours with opposite labels. Intuitively, this process will learn a projected feature space in the source domain where the margin between the opposite labelled nearest neighbours is maximised, thereby minimising the risk of misclassifications. We project the source domain's labelled instances using $\mathcal{S}_{\text{prj}}$ for the purpose of training a classifier for predicting the target task labels such as positive/negative sentiment in cross-domain sentiment classification or part-of-speech tags in cross-domain part-of-speech tagging.
- Second, we use the classifier trained in the previous step to assign pseudo labels for the (unlabelled) target domain instances. Different strategies can be used for this label inference process such as selecting instances with the highest classifier confidence as in self-training or checking the agreement among multiple classifier as in tri-training.
- Third, we use the pseudo-labelled target domain instances to learn a projection for the target domain ($\mathcal{T}_{\text{prj}}$) following the same procedure used to learn $\mathcal{S}_{\text{prj}}$. Specifically, we learn a projected feature space in the target domain where the margin between the opposite pseudo-labelled nearest neighbours is maximised. We project labelled instances in the source domain and pseudo-labelled instances in the target domain respectively using $\mathcal{S}_{\text{prj}}$ and $\mathcal{T}_{\text{prj}}$, and use those projected instances to learn a classifier for the target task.

As an evaluation task, we perform cross-domain sentiment classification on the Amazon multi-domain sentiment dataset (Blitzer et al., 2007). Although most prior work on UDA have used this dataset as a standard evaluation benchmark, the evaluations have been limited to the four domains *books*, *dvds*, *electronic appliances* and *kitchen appliances*. We too report performances on those four domains for the ease of comparison against prior work. However, to reliably estimate the generalisability of the proposed method, we perform an additional extensive evaluation using 16 other domains included in the original version of the Amazon multi-domain sentiment dataset.

Results from the cross-domain sentiment classification reveal several interesting facts. A baseline that uses $\mathcal{S}_{\text{prj}}$ alone would still outperform a baseline that uses a classifier trained using only the source domain's labelled instances on the target domain test instances, without performing any adaptations. This result shows that it is useful to consider the label distribution available in the source domain to learn a projection even though it might be different to that in the target domain.

On the other hand, training a classifier using the pseudo-labelled target domain instances alone, without learning $\mathcal{T}_{\text{prj}}$ further improves performance. This result shows that pseudo labels inferred for the target domain unlabelled instances can be used to overcome the issue of lack of labelled instances in the target domain.

Moreover, if we further use the pseudo-labelled instances to learn $\mathcal{T}_{\text{prj}}$, then we see a significant improvement of performance across all domain pairs, suggesting that UDA can benefit from both projection learning and self-training.

These experimental results support our claim that it is beneficial to combine projection-based and self-training-based UDA approaches. Moreover, our proposed method outperforms all self-training based domain adaptation methods such as tri-training (Zhou and Li, 2005; Søgaard, 2010) and is competitive against neural domain adapta-

tion methods (Louizos et al., 2015; Ganin et al., 2016; Saito et al., 2017; Ruder and Plank, 2018).

2 Related Work

Self-training (Yarowsky, 1995) has been adapted to various cross-domain NLP tasks such as document classification (Raina et al., 2007), POS tagging (McClosky et al., 2006; Reichart and Rappoport, 2007) and sentiment classification (Drury et al., 2011). Although different variants of self-training algorithms have been proposed (Abney, 2007; Yu and Kübler, 2011) a common recipe can be recognised involving the following three steps: (a) Initialise the training dataset, $\mathcal{L} = \mathcal{S}_L$, to the labelled data in the source domain, and train a classifier for the target task using $\mathcal{L}$, (b) apply the classifier trained in step (a) to the unlabelled data in the target domain $\mathcal{T}_U$, and append the most confident predictions as identified by the classifier (e.g. higher than a pre-defined confidence threshold τ) to the labelled dataset $\mathcal{L}$, (c) repeat the two steps (a) and (b) until we cannot append additional high confidence predictions to $\mathcal{L}$.

Another popular approach for inferring labels for the target domain is co-training (Blum and Mitchell, 1998), where the availability of multiple views of the feature space is assumed. In the simplest case where there are two views available for the instances, a separate classifier is trained using the source domain's labelled instances that involve features from a particular view only. Next, the two classifiers are used to predict pseudo labels for the target domain unlabelled instances. If the two classifiers agree on the label for a particular unlabelled instances, then that label is assigned to that instance. Co-training has been applied to UDA (Yu and Kübler, 2011; Chen et al., 2011), where the feature spaces in the source and target domains were considered as the multiple views. The performance of co-training will depend on the complementarity of the information captured by the different feature spaces. Therefore, it is an important to carefully design multiple feature spaces when performing UDA. In contrast, our proposed method does not require such multiple views and does not require training multiple classifiers for the purpose of assigning pseudo labels for the target domain unlabelled instances, which makes the proposed method easy to implement.

Tri-training (Zhou and Li, 2005) relaxes the requirement of co-training for the feature spaces to be sufficient and redundant views. Specifically, in tri-training, as the name implies three separate classifiers are trained using bootstrapped subsets of instances sampled from the labelled instances. If at least two out of the three classifiers agree upon a label for an unlabelled instance, that label is then assigned to the unlabelled instance. Søgaard (2010) proposed a variation of tri-training (i.e. tri-training with diversification) that diversifies the sampling process and reduces the number of additional instances, where they require exactly two out of the three classifiers to agree upon a label and the third classifier to disagree. It has been shown that the classic tri-training algorithm when applied to UDA acts as a strong baseline that outperforms even more complex SoTA neural adaptation methods (Ruder and Plank, 2018). As later shown in our experiments, the proposed **Self-Adapt** method consistently outperforms self-training, tri-training and tri-training with diversification across most of the domain pairs considered.

Projection-based approaches for UDA learn a (possibly lower-dimensional) projection where the difference between the source and target feature spaces is reduced. For example, Structural Correspondence Learning (SCL) (Blitzer et al., 2006, 2007) learns a projection using a set of domain invariant common features called *pivots*. Different strategies have been proposed in the literature for finding pivots for different tasks such as the frequency of a feature in a domain for cross-domain POS tagging (Blitzer et al., 2006; Cui et al., 2017a), mutual information (Blitzer et al., 2007) and pointwise mutual information (Bollegala et al., 2011, 2015) for cross-domain sentiment classification. Cui et al. (2017b) proposed a method for learning the appropriateness of a feature as a pivot (*pivothood*) from the data during training, without requiring any heuristics. Although we use projections in the proposed method, unlike prior work on projection-based UDA, we *do not* require splitting the feature space into domain independent and domain specific features. Moreover, we learn two separate projections for each of the source and target domain, which gives us more flexibility to address the domain-specific constrains in the learnt projections.

Neural adaptation methods have recently reported SoTA for UDA. Louizos et al. (2015) proposed a Variational Fair Autoencoder (VFAE) to learn an invariant representation for a domain.

They used Maximum Mean Discrepancy (MMD) (Gretton et al., 2006) for further promoting the invariant projected feature space. Ganin et al. (2016) proposed Domain Adversarial Neural Network (DANN) to learn features that combine the discriminative power of a classifier and the domain-invariance of the projection space to simultaneously learn adaptable and discriminative projections. Saito et al. (2017) proposed a deep tri-training method with three neural networks, two for pseudo labelling the target unlabelled data and another one for learning discriminator using the inferred pseudo labels for the target domain. Ruder and Plank (2018) proposed Multi-task Tri-training (MT-Tri) based on tri-training and Bi-LSTM. They show that tri-training is a competitive baseline and rivals more complex neural adaptation methods. Although MT-Tri does not outperform SoTA on cross-domain sentiment classification tasks, their proposal reduces the time and space complexity required by the classical tri-training.

As stated above, our proposed method **Self-Adapt**, differs from the prior work discussed above in that it (a) does not require pivots, (b) does not require multiple feature views, (c) learns two different projections for the source and target domains and (d) combines a projection and a self-training step in a non-iterative manner to improve the performance in UDA.

3 Self-Adaptation (Self-Adapt)

In UDA, we are given a set of positively ($\mathcal{S}_L^+$) and negatively ($\mathcal{S}_L^-$) labelled instances for a source domain $\mathcal{S}$ ($\mathcal{S}_L = \mathcal{S}_L^+ \cup \mathcal{S}_L^-$), and sets of unlabelled instances $\mathcal{S}_U$ and $\mathcal{T}_U$ respectively for the source and target domain $\mathcal{T}$. Given a dataset $\mathcal{D}$, we are required to learn a classifier $f(\boldsymbol{x}, y; \mathcal{D})$ that returns the probability of a test instance $\boldsymbol{x}$ taking a label y. For simplicity, we consider the pairwise adaptation from a single source to single target, and binary ($y \in \{-1, 1\}$) classification as the target task. However, self-adapt can be easily extended to multi-domain and multi-class UDA.

We represent an instance (document/review) x by a bag-of-n-gram (BonG) embedding (Arora et al., 2018), where we add the pre-trained d-dimensional word embeddings $\boldsymbol{w} \in \mathbb{R}^d$ for the words $w \in x$ to create a d-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^d$ representing x. Self-adapt consists of three steps: (a) learning a source projection using $\mathcal{S}_L$ (Section 3.1), (b) pseudo labelling $\mathcal{T}_U$ using a classifier trained on the projected $\mathcal{S}_L$ (Section 3.2); (c) learning a target projection using the pseudo-labelled target instances, and then learning a classifier f for the target task (Section 3.3).

3.1 Source Projection Learning ($\mathcal{S}_{\text{prj}}$)

In UDA, the adaptation task does not vary between the source and target domains. Therefore, we can use $\mathcal{S}_L$ to learn a projection for the source domain $\mathcal{S}_{\text{prj}}$ where the separation between an instance $x \in \mathcal{S}_L$ and its opposite-labelled nearest neighbours is maximised. Specifically, for an instance x we represent the set of k of its nearest neighbours $\text{NN}(\boldsymbol{x}, \mathcal{D}, k)$ selected from a set $\mathcal{D}$ by a vector $\phi(\boldsymbol{x}, \mathcal{D}, k)$ as the sum of the word embeddings of the neighbours given by (1).

$$\phi(\boldsymbol{x}, \mathcal{D}, k) = \sum_{u \in \text{NN}(\boldsymbol{x}, \mathcal{D}, k)} \theta(x, u)\boldsymbol{u}. \quad (1)$$

Here, the weight $\theta(x, u)$ is computed using the cosine similarity between $\boldsymbol{u}$ and $\boldsymbol{x}$, and is normalised s.t. $\sum_{u \in \text{NN}(\boldsymbol{x}, \mathcal{D}, k)} \theta(x, u) = 1$. Other similarity measures can also be used instead of cosine, for example, Euclidean distance (Van Asch and Daelemans, 2016). Then, $\mathcal{S}_{\text{prj}}$ is defined by the projection matrices $\mathbf{A}_+ \in \mathbb{R}^{d \times d}$ and $\mathbf{A}_- \in \mathbb{R}^{d \times d}$ and is learnt by maximising the objective O_L given by (2).

$$O_L(\mathbf{A}_+, \mathbf{A}_-) = \sum_{\boldsymbol{x} \in \mathcal{S}_L^+} ||\mathbf{A}_+\boldsymbol{x} - \mathbf{A}_-\phi(\boldsymbol{x}, \mathcal{S}_L^-, k)||_2^2 + \sum_{\boldsymbol{x} \in \mathcal{S}_L^-} ||\mathbf{A}_-\boldsymbol{x} - \mathbf{A}_+\phi(\boldsymbol{x}, \mathcal{S}_L^+, k)||_2^2 \quad (2)$$

We initialise $\mathbf{A}_+$ and $\mathbf{A}_-$ to the identify matrix $\mathbf{I} \in \mathbb{R}^{d \times d}$ and apply Adam (Kingma and Ba, 2014) to find their optimal values denoted respectively by $\mathbf{A}_+^*$ and $\mathbf{A}_-^*$. Finally, we project $\mathcal{S}_L$ using the learnt $\mathcal{S}_{\text{prj}}$ to obtain a projected set of source domain labelled instances $\mathcal{S}_L^* = \mathbf{A}_+^* \circ \mathcal{S}_L^+ \cup \mathbf{A}_-^* \circ \mathcal{S}_L^-$. Here, we use the notation $\mathbf{A} \circ \mathcal{D} = \{\mathbf{A}\boldsymbol{x} | \boldsymbol{x} \in \mathcal{D}\}$ to indicate the application of a projection matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ on elements $\boldsymbol{x} \in \mathbb{R}^d$ in a dataset $\mathcal{D}$.

3.2 Pseudo Label Generation (PL)

In UDA, we do not have labelled data for the target domain. To overcome this issue, inspired by prior work on self-training approaches to UDA, we train a classifier $f(\boldsymbol{x}, y; \mathcal{S}_L^*)$ on $\mathcal{S}_L^*$ first and then use this classifier to assign pseudo labels for the target domain's unlabelled data $\mathcal{T}_U$, if the classifier

Algorithm 1 Pseudo Label Generation

Input: source domain positively labelled data $\mathcal{S}_L^+$,
source domain negatively labelled data $\mathcal{S}_L^-$,
source domain positive transformation matrix $\mathbf{A}_+$,
source domain negative transformation matrix $\mathbf{A}_-$,
target domain unlabelled data $\mathcal{T}_U$,
a set of target classes $Y = \{+1, -1\}$,
classification confidence threshold τ.

Output: target domain pseudo-labelled data $\mathcal{T}'_L$

$\mathcal{S}_L^* \leftarrow \mathbf{A}_+^* \mathcal{S}_L^+ \cup \mathbf{A}_-^* \mathcal{S}_L^-$
$\mathcal{T}'_L \leftarrow \emptyset$
for $\boldsymbol{x} \in \mathcal{T}_U$ **do**
  $y \in Y$, $p(t = y|\boldsymbol{x}) = f(\boldsymbol{x}, y; \mathcal{S}_L^*)$
  {probability of $\boldsymbol{x}$ belongs to class y}
  if $p(t = y|\boldsymbol{x})) > \tau$ **then**
    $\mathcal{T}'_L \leftarrow \mathcal{T}'_L \cup \{(\boldsymbol{x}, y)\}$
  end if
end for
return $\mathcal{T}'_L$

is more confident than a pre-defined threshold τ. Algorithm 1 returns a pseudo-labelled dataset $\mathcal{T}'_L$ for the target domain. According to the classical self-training (Yarowsky, 1995; Abney, 2007), $\mathcal{T}'_L$ will be appended to $\mathcal{S}_L^*$ and the classifier is retrained on this extended labelled dataset. The process is repeated until no further unlabelled instances can be assigned labels with confidence higher than τ. However, in our preliminary experiments, we found that this process does not improve the performance in UDA beyond the first iteration. Therefore, we limit the number of iterations to one as shown in Algorithm 1. Doing so also speeds up the training process over classical self-training, which retrains the classifier and iterates.

3.3 Target Projection Learning ($\mathcal{T}_{\text{prj}}$)

Armed with the pseudo-labelled data generated via Algorithm 1, we can now learn a projection for the target domain, $\mathcal{T}_{\text{prj}}$, following the same procedure we proposed for learning $\mathcal{S}_{\text{prj}}$ in Section 3.1. Specifically, $\mathcal{T}_{\text{prj}}$ is defined by the two target-domain projection matrices $\mathbf{B}_+ \in \mathbb{R}^{d \times d}$ and $\mathbf{B}_- \in \mathbb{R}^{d \times d}$ that maximises the distance between each pseudo-labelled target instance x and its k opposite labelled nearest neighbours selected from positively ($\mathcal{T}_L'^+$) and negatively ($\mathcal{T}_L'^-$) pseudo-labelled instances. The objective O'_L for this optimisation problem is given by (3).

$$O'_L(\mathbf{B}_+, \mathbf{B}_-) = \sum_{\boldsymbol{x} \in \mathcal{T}_L'^+} \left|\left|\mathbf{B}_+ \boldsymbol{x} - \mathbf{B}_- \phi(\boldsymbol{x}, \mathcal{T}_L'^-, k)\right|\right|_2^2 + \sum_{\boldsymbol{x} \in \mathcal{T}_L'^-} \left|\left|\mathbf{B}_- \boldsymbol{x} - \mathbf{B}_+ \phi(\boldsymbol{x}, \mathcal{T}_L'^+, k)\right|\right|_2^2 \quad (3)$$

Likewise with $\mathcal{S}_{\text{prj}}$, $\mathbf{B}_+$ and $\mathbf{B}_-$ are initialised to the identify matrix $\mathbf{I} \in \mathbb{R}^{d \times d}$, and Adam is used to find their minimisers denoted respectively by $\mathbf{B}_+^*$ and $\mathbf{B}_-^*$. We project the target domain pseudo-labelled data using $\mathcal{T}_{\text{prj}}$ to obtain $\mathcal{T}_L'^* = \mathbf{B}_+^* \circ \mathcal{T}_L'^+ \cup \mathbf{B}_-^* \circ \mathcal{T}_L'^-$. Finally, we train a classifier $f(\boldsymbol{x}, y; \mathcal{S}_L^* \cup \mathcal{T}_L'^*)$ for the target task using both source and target projected labelled instances $\mathcal{S}_L^* \cup \mathcal{T}_L'^*$. Any binary classifier can be used for this purpose. In our experiments, we use ℓ_2 regularised logistic regression following prior work in UDA (Blitzer et al., 2006; Pan et al., 2010; Bollegala et al., 2013). Moreover, by using a simple linear classifier, we can decouple the projection learning step from the target classification task, thereby more directly evaluate the performance of the former.

4 Experiments

Our proposed method *does not* assume any information about the target task and can be in principle applied for any domain adaptation task. We use cross-domain sentiment classification as an evaluation task in this paper because it has been used extensively in prior work on UDA, thereby enabling us to directly compare the performance of our proposed method against previously proposed UDA methods. In particular, we use the Amazon multi-domain sentiment dataset, originally created by Blitzer et al. (2007), as a benchmark dataset in our experiments. This dataset includes Amazon Product Reviews from four categories: Books (**B**), DVDs (**D**), Electronic Appliances (**E**) and Kitchen Appliances (**K**). Considering each category as a domain[2], we can generate $\binom{4}{2} = 12$ pair-wise adaptation tasks involving a single source and a single target domain.

An Amazon product review is assigned 1-5 star rating and product reviews with 4 or 5 stars are labelled as positive, whereas 1 or 2 star reviews

[2]Multiple reviews might exist for the same product within the same domain. Products are not shared across domains.

are labelled as negative. 3 star reviews are ignored because of their ambiguity. In addition to the labelled reviews, the Amazon multi-domain dataset contains a large number of unlabelled reviews for each domain. We use the official balanced train and test dataset splits, which has 800 (pos), 800 (neg) training instances and 200 (pos), 200 (neg) test instances for each domain. We name this dataset as the Multi-domain Adaptation Dataset (**MAD**).

One issue that is often raised with **MAD** is that it contains only four domains. In order to robustly evaluate the performance of an UDA method we must evaluate on multiple domains. Therefore, in addition to **MAD**, we also evaluate on an extended dataset that contains 16 domains. We name this dataset as the Extended Multi-domain Adaptation Dataset (**EMAD**). The reviews for the 16 domains contained in **EMAD** were also collected by Blitzer et al. (2007), but were not used in the evaluations. The same star-based procedure used in **MAD** is used to label the reviews in **EMAD**. We randomly select 20% of the available labelled reviews as test data and construct a balanced training dataset from the rest of the labelled reviews (i.e. for each domain we have equal number of positive and negative labelled instances in the train datasets). Likewise in **MAD**, we generate $\binom{16}{2} = 240$ pair-wise domain adaptation tasks from **EMAD**.

We train an ℓ_2 regularised logistic regression as the target (sentiment) classifier, in which we tune the regularisation coefficient using validation data. We randomly select 10% from the target domain labelled data, which is separate from the train or test data. We tune regularisation coefficient in $[0.001, 0.01, 0.1, 1]$. We use 300 dimensional pre-trained GloVe word embeddings (Pennington et al., 2014) to create BonG embeddings for uni and bigrams. We found that a maximum of 100 epochs to be sufficient to reach convergence in all projection learning tasks for all domains. The source code implementation of **self-adapt** will be made publicly available upon paper acceptance.

4.1 Effect of Projection Learning and Pseudo-Labelling

Our proposed method consists of 3 main steps as described in Section 3: source projection learning, pseudo labelling and target projection learning. Using **MAD**, in Table 1, we compare the relative effectiveness of these three steps towards the overall performance in UDA using $k = 1$ for all the steps. Specifically, we consider the following baselines.

NA: No-adaptation. Learn a classifier from $\mathcal{S}_L$ and simply use it to classify sentiment on target domain test instances, without performing any domain adaptation.

$\mathcal{S}_{\text{prj}}$: Learn a source projection $\mathcal{S}_{\text{prj}}$ and apply it to project $\mathcal{S}_L$ to obtain $\mathcal{S}_L^* = \mathbf{A}_+^* \circ \mathcal{S}_L^+ \cup \mathbf{A}_-^* \circ \mathcal{S}_L^-$. Train a sentiment classifier using $\mathcal{S}_L^*$ and use it to classify target domain test instances.

$\mathcal{S}_{\text{prj}}$ +PL: Use the classifier trained using $\mathcal{S}_L^*$ on target domain unlabelled data to create a pseudo-labelled dataset $\mathcal{T}_L'$. Train a sentiment classifier on $\mathcal{S}_L^* \cup \mathcal{T}_L'$ and use it to classify target domain test instances.

$\mathcal{S}_{\text{prj}}$ +$\mathcal{T}_{\text{prj}}$: This is the proposed method including all three steps. A target projection $\mathcal{T}_{\text{prj}}$ is learnt using $\mathcal{T}_L'$ and is applied to obtain $\mathcal{T}_L'^* = \mathbf{B}_+^* \circ \mathcal{T}_L'^+ \cup \mathbf{B}_-^* \circ \mathcal{T}_L'^-$. Finally, a sentiment classifier is trained using $\mathcal{S}_L^* \cup \mathcal{T}_L'^*$ and used to classify target domain test instances.

With all methods, we keep $k = 1$ in the nearest neighbour feature representation in (1) for the ease of comparisons. Confidence threshold τ is tuned in the range $[0.5, 0.9]$ using cross-validation. From Table 1 we see that $\mathcal{S}_{\text{prj}}$ consistently outperforms **NA**, showing that even without using any information from the target domain, it is still useful to learn source domain projections that discriminates instances with opposite labels. When we perform pseudo labelling on top of source projection learning ($\mathcal{S}_{\text{prj}}$ **+PL**) we see a slight but consistent improvement in all domain-pairs. However, when we use the pseudo labelled instances to learn a target projection ($\mathcal{S}_{\text{prj}}$ +$\mathcal{T}_{\text{prj}}$) we obtain the best performance in all domain-pairs. Moreover, the obtained results are significantly better under the stricter $p < 0.001$ level over the **NA** baseline in 7 out of 12 domain-pairs.

Table 3 shows the classification accuracy for the **EMAD**. Due to the limited availability of space, we show the average classification accuracy for adapting to the same target domain instead of showing all 240 domain-pairs for **EMAD**. Likewise in **MAD**, we see in **EMAD** we obtain the best results when we use both source and target projections. Interestingly, we see that the proposed method adapting well even to the domains

$\mathcal{S}-\mathcal{T}$	**NA**	$\mathcal{S}_{\text{prj}}$	$\mathcal{S}_{\text{prj}}$ **+PL**	$\mathcal{S}_{\text{prj}}$ **+**$\mathcal{T}_{\text{prj}}$
B-D	73.50	74.25	74.50	**76.25**
B-E	64.00	73.25**	73.50**	**77.50****
B-K	68.50	75.25*	76.00*	**78.75****
D-B	74.00	**75.50**	**75.50**	**75.50**
D-E	64.75	71.25*	71.50*	**74.25****
D-K	71.50	75.00	76.25	**79.75****
E-B	67.25	74.50*	75.25**	**76.25****
E-D	66.75	67.75	68.00	**69.50**
E-K	76.00	**81.00**	**81.00**	**81.00**
K-B	63.00	73.75**	73.75**	**75.00****
K-D	71.25	71.25	71.00	**72.50**
K-E	69.00	77.50**	78.00**	**78.25****

Table 1: Target domain test data classification accuracy for the different steps in the proposed method ($k = 1$). $\mathcal{S}-\mathcal{T}$ denotes adapting from a source $\mathcal{S}$ to a target $\mathcal{T}$ domain. The best result for each domain-pair is bolded. Statistically significant improvements over **NA** according to the binomial exact test are shown by "*" and "**" respectively at $p = 0.01$ and $p = 0.001$ levels.

with smaller numbers of unlabelled data such as **gourmet_food** (168 labelled and 267 unlabelled train instances). This is encouraging because it shows that the proposed method can overcome the lack of labelled instances via pseudo labelling and projection learning.

4.2 Comparisons against Self-Training

Ruder and Plank (2018) evaluated classical general-purpose semi-supervised learning methods proposed for inducing pseudo labels for unlabelled instances using a seed set of labelled instances in the context of UDA. They found that tri-training to outperform more complex neural SoTA UDA methods. Considering the fact that **Self-Adapt** is performing pseudo-labelling, similar to other self-training methods, it is interesting to see how well it compares against classical self-training methods for inducing labels (Yarowsky, 1995; Abney, 2007; Zhou and Li, 2005; Søgaard, 2010) when applied to UDA. Specifically, we consider the classical self-training (Yarowsky, 1995; Abney, 2007) (**Self**), Tri-training (Zhou and Li, 2005) (**Tri**) and Tri-training with diversification (Søgaard, 2010) (**Tri-D**). For each of those methods, we use the labelled data in the source domain as seeds and induce labels for the unlabelled data in the target domain. Table 2 reports the results on **MAD**.

We re-implement the classical self-training methods considered by Ruder and Plank (2018) and evaluated them against the proposed self-adapt on the same datasets, feature representations and settings to conduct a fair comparison. All classical self-training methods were trained using the source domain labelled instances $\mathcal{S}_L$ as seed data. As discussed in Section 3.2, similar to **Self-Adapt**, we observed that the performance did not significantly increase beyond the first iteration for any of the classical self-training methods in UDA. Consequently, we compare all classical self-training methods for their peak performance, obtained after the first iteration. We tune the confidence threshold τ for each method using validation data and found the optimal value of τ to fall in the range $[0.6, 0.9]$. k is a hyperparameter selected using validation dataset for **Self-Adapt** in comparison.

Experimental results on **MAD** and **EMAD** are shown respectively in Tables 2 and 4. From those Tables, we see that **Self-Adapt** for most of the domain pairs performs similarly or slightly worse than **NA**. Although **Tri** and **Tri-D** outperform **NA** on all cases, we found that those two methods are highly sensitive to the seed instances used to initialise the pseudo-labelling process. We find the proposed **Self-Adapt** to outperform all classical self-training based methods in 11 out of 12 domain pairs in **MAD** and in all 16 target domains in **EMAD**, showing a strong and robust improvement over classical self-training methods. This result shows that by combining source and target domain projections with self-training, we can obtain superior performance in UDA in comparison to using classical self-training methods alone.

S-T	**NA**	**Self**	**Tri**	**Tri-D**	**Self-Adapt**
B-D	73.50	74.25	74.75	**77.25**	**77.25**
B-E	64.00	65.00	73.25**	72.00**	**78.50****
B-K	68.50	70.75	73.25	73.75	**79.00****
D-B	74.00	73.00	77.00	76.00	**81.00***
D-E	64.75	65.25	73.75**	70.50*	**75.50****
D-K	71.50	71.75	75.75	74.00	**79.75****
E-B	67.25	68.25	68.50	74.25*	**76.25****
E-D	66.75	66.25	68.25	**73.50***	69.50
E-K	76.00	76.50	**82.50***	81.00	**82.50***
K-B	63.00	63.00	70.00*	73.00**	**75.00****
K-D	71.25	71.00	71.25	72.00	**72.75**
K-E	69.00	68.75	73.00	75.50*	**79.00****

Table 2: Target domain test data classification accuracy of classical self-training methods when applied to UDA.

4.3 Comparisons against Neural UDA

In Table 5, we compare **Self-Adapt** against the following neural UDA methods on **MAD**: Variational Fair Autoencoder (Louizos et al., 2015) (**VFAE**), Domain-adversarial Neural Networks (Ganin et al., 2016) (**DANN**), Asymmet-

Target Domain	NA	$\mathcal{S}_{\text{prj}}$	$\mathcal{S}_{\text{prj}}$ +PL	$\mathcal{S}_{\text{prj}}$ +$\mathcal{T}_{\text{prj}}$
apparel	71.39	75.31	75.66	**76.68***
baby	68.00	71.06	71.37	**72.63**
beauty	69.66	73.20	73.18	**73.70**
camera_and_photo	68.46	73.46	74.02	**74.54***
computer_and_video_games	61.78	67.38	67.85	**68.11***
gourmet_food	72.34	81.13**	82.89**	**83.15****
grocery	71.01	76.90*	77.57 *	**78.14***
health_and_personal_care	65.85	68.38	68.67	**69.24**
jewelry_and_watches	68.90	77.86**	79.11**	**79.79****
magazines	65.28	71.54*	71.45*	**72.16***
music	66.27	70.69	70.89	**71.41**
outdoor_living	71.97	76.77	78.01*	**78.93***
software	65.72	69.56	69.60	**70.31**
sports_and_outdoors	66.56	70.18	70.67	**71.02**
toys_and_games	69.57	72.82	73.22	**73.65**
video	64.19	67.59	68.37	**68.97**

Table 3: Average classification accuracy on each target domain in **EMAD** for the steps in the proposed method ($k = 1$).

Target Domain	NA	Self	Tri	Tri-D	Self-Adapt
apparel	71.39	71.38	73.96	73.89	**76.68**
baby	68.00	67.96	69.71	69.84	**72.63**
beauty	69.66	70.54	71.73	70.49	**73.70**
camera_and_photo	68.46	68.44	71.31	71.28	**74.54***
computer_and_video_games	61.78	62.28	64.65	64.93	**68.11***
gourmet_food	72.34	74.10	75.39	77.88	**83.15****
grocery	71.01	71.83	75.22	74.29	**78.14***
health_and_personal_care	65.85	66.08	67.10	67.07	**69.24**
jewelry_and_watches	68.90	71.04	76.67**	76.21**	**79.79****
magazines	65.28	65.34	67.58	67.47	**72.16***
music	66.27	66.22	68.82	68.27	**71.41**
outdoor_living	71.97	74.01	76.21	74.79	**78.93***
software	65.72	65.47	67.36	67.53	**70.31**
sports_and_outdoors	66.56	66.87	69.22	68.05	**71.02**
toys_and_games	69.57	70.03	71.76	72.32	**73.65**
video	64.19	64.88	66.34	67.49	**68.97**

Table 4: Average classification accuracy on each target domain in **EMAD** of classical self-training based methods when applied to UDA.

ric Tri-training (Saito et al., 2017) (**Asy-Tri**), and Multi-task Tri-training (Ruder and Plank, 2018) (**MT-Tri**). We select these methods as they are the current SoTA for UDA on **MAD**, and report the results from the original publications in Table 5.

Although only in 3 out of 12 domain-pairs **Self-Adapt** is obtaining the best performance, the difference of performance between **DANN** and **Self-Adapt** is not statistically significant. Although **MT-Tri** is outperforming **Self-Adapt** in 8 domain-pairs, it is noteworthy that **MT-Tri** is using a larger feature space than that of **Self-Adapt**. Specifically, **MT-Tri** is using 5000 dimensional tf-idf weighted unigram and bigram vectors for representing reviews, whereas we **Self-Adapt** uses a 300 dimensional BonG representation computed using pre-trained GloVe vectors. Moreover, prior work on neural UDA have not used the entire unlabelled datasets and have sampled a smaller subset due to computational feasibility. For example, **MT-Tri** uses only 2000 unlabelled instances for each domain despite the fact that the original unlabelled datasets contain much larger numbers of reviews. This is not only a waste of available data but it is also non-obvious as how to subsample unlabelled data for training. Our preliminary experiments revealed that the performance of neural UDA methods to be sensitive to the unlabelled datasets used.[3] On the other hand, **Self-Adapt** does not require sub-sampling of unlabelled data and uses all the available unlabelled data for UDA. During the pseudo-labelling step, **Self-Adapt** automatically selects a subset of unlabelled target instances that are determined to be confident by the classifier more than a pre-defined threshold τ. The ability to operate on a lower-dimensional feature space and obviating the need to subsample unlabelled data are properties of the proposed method that are attractive when applying UDA methods on large datasets and across multiple domains.

[3]Unfortunately, the source code for MT-Tri was not available for us to run this method with the same set of features and unlabelled dataset that we used. Therefore, we report the results from the original publication.

S-T	VFAE	DANN	Asy-Tri	MT-Tri	Self-Adapt
B-D	79.90	78.40	80.70	**81.47**	77.25
B-E	79.20	73.30	**79.80**	78.62	78.50
B-K	81.60	77.90	**82.50**	78.09	79.00
D-B	75.50	72.30	73.20	77.49	**81.00****
D-E	78.60	75.40	77.00	**79.66**	75.50
D-K	82.20	78.30	**82.50**	81.23	79.75
E-B	72.70	71.10	73.20	73.43	**76.25**
E-D	**76.50**	73.80	72.90	75.05	69.50
E-K	85.00	85.40	86.90	**87.07**	82.50
K-B	72.00	70.90	72.50	73.60	**75.00**
K-D	73.30	74.00	74.90	**77.41**	72.75
K-E	83.80	84.30	84.60	**86.06**	79.00

Table 5: Classification accuracy compared with neural adaptation methods. The best result is bolded. Statistically significant improvements over **DANN** according to the binomial exact test are shown by "*" and "**" respectively at $p = 0.01$ and $p = 0.001$ levels.

5 Conclusions

We proposed **Self-Adapt**, an UDA method that combines projection learning and self-training. Our experimental results on two datasets for cross-domain sentiment classification show that projection learning and self-training have complementary strengths and jointly contribute to improve UDA performance. In future, we plan to apply **Self-Adapt** to other UDA tasks in NLP such as cross-domain POS tagging and NER.

References

Steven Abney. 2007. *Semisupervised Learning for Computational Linguistics*, 1st edition. Chapman & Hall/CRC.

Sanjeev Arora, Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and LSTMs. In *International Conference on Learning Representations*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*, pages 120–128.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Danushka Bollegala, Tingting Mu, and Yannis Goulermas. 2015. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28(2):398–410.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *ACL/HLT'11*, pages 132 – 141.

Danushka Bollegala, David Weir, and John Carroll. 2013. Cross-domain sentiment classification using a sentiment sensitive thesaurus. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1719 – 1731.

Minmim Chen, Kilian Q. Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NIPS'11*.

Xia Cui, Frans Coenen, and Danushka Bollegala. 2017a. Effect of data imbalance on unsupervised domain adaptation of part-of-speech tagging and pivot selection strategies. In *Proc. of the Wokshop on Learning With Imbalanced Domains: Theory and Applications (LIDTA) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 103–115.

Xia Cui, Frans Coenen, and Danushka Bollegala. 2017b. TSP: Learning task-specific pivots for unsupervised domain adaptation. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 754–771.

Brett Drury, Luís Torgo, and Jose Joao Almeida. 2011. Guided self training for sentiment classification. In *Proceedings of Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, pages 9–16.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. 2006. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. 2015. The variational fair autoencoder. *CoRR*, abs/1511.00830.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344. Association for Computational Linguistics.

Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. 2018. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. In *International Conference on Learning Representations*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proc. of WWW*, pages 751–760.

Jeffery Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: global vectors for word representation. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: Transfer learning from unlabeled data. In *ICML'07*.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL 2007*, pages 616 – 623.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. pages 1044–1054.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 205–208, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vincent Van Asch and Walter Daelemans. 2016. Predicting the effectiveness of self-training: Application to sentiment classification. *arXiv preprint arXiv:1601.03288*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ning Yu and Sandra Kübler. 2011. Filling the gap: Semi-supervised learning for opinion detection across domains. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 200–209. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.