Real-World Deep Domain Adaptation through Prediction Propagation

by

Michiel Anton Bakker

B.Sc, Delft University of Technology (2012)
M.Sc, Delft University of Technology (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2019

Signature redacted

Certified by .....
Alex P. Pentland
Professor of Media Arts and Sciences
Thesis Supervisor

Signature redacted

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Real-World Deep Domain Adaptation through Prediction Propagation

by

Michiel Anton Bakker

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Deep learning-based natural language processing classifiers often have difficulty classifying texts that stem from a different domain than the labeled training data. Many domain adaptation methods have been proposed to train classifiers using only labeled texts from a single domain and unlabeled texts from other domains. Nevertheless, we find that the state-of-the-art methods all lack one or more desirable properties for real-world modeling. In particular, we find that the many methods using a domain-adversarial loss are unable to model domains with different label distributions. Motivated by these limitations, we are developing a new method, Prediction Propagation, that can classify texts from different domains without using an adversarial loss. Our method will use the label prediction for reconstructing the input text and backpropagates through the prediction as a way to learn label-related information for the new domain. Our method has the desirable properties for real-world modeling while not compromising on performance.

Thesis Supervisor: Alex P. Pentland
Title: Professor of Media Arts and Sciences

Acknowledgments

Foremost, I want to thank my advisor Sandy. During most of my previous research projects, the scale and costs of doing a quantum physics experiment demanded long-term group-level planning of resources limiting the freedom of a junior researcher like me. Coming from an environment like that, the freedom that Sandy gave me in his group was a welcome surprise but could also feel overwhelming and confusing. In hindsight, however, it led to the wide variety of collaborations, research directions, and projects that I am now proud of and that helped me explore what really excites me as a researcher. For Sandy, no direction is out of scope and no idea is too wild. This, together with his unconditional support, made me appreciate even more how unique this opportunity really is and how thankful I am to be a graduate student in his group and at MIT at large.

I am grateful to everyone in Human Dynamics and the Media Lab for the help, discussions, collaborations, adventures, and laughs. Abhi and Bjarke, thanks for the fun and productive time while working together on the project that led to this thesis (and hopefully soon an impactful conference submission). Alejandro and Kevin, it is truly a joy to have worked on so many other projects with two of my best friends.

Huge thanks also to Janet and Leslie. Even though I spent most of my time in the Media Lab, you still made me feel part of the EECS community. I am amazed by how smoothly you are running MIT's largest department while still being so approachable, responsive, and caring.

I want to thank Irwin Mark Jacobs and Joan Klein Jacobs for supporting my first year of graduate studies.

Finally, I want to thank my friends, family, Jacques and Maya for supporting me, adventuring with me, and inspiring me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern natural language processing (NLP) models rely heavily on machine learning, which has enabled impressive results, but has also made the learned models closely tied with the training data [12]. For NLP, there are large amounts of unlabeled data available, and it is thus useful to exploit this data to make classifiers more robust to data coming from other domains than that of the training data.

The task of learning models that can exploit unlabeled data sources from other domains to successfully classify observations in those domains has been formalized in [10] under the name of domain adaptation. Many methods have been proposed with some prominent approaches leveraging unsupervised learning to learn a shared representation [5, 37, 7] and others using adversarial loss to learn domain-invariant representations [1, 15, 40].

We find that many of the state-of-the-art methods rely on assumptions about the datasets that are often not upheld for real-world applications. Motivated by these limitations, we propose a new method, Prediction Propagation, that uses the label prediction for each piece of text, together with the surrounding words, to separately reconstruct each word in the text, see Figure 1-1. When backpropagating through the label classifier, the reconstruction loss forces the model to improve its label-related information for domains without labeled data. A new neighborhood encoding architecture and multi-phase training is used to ensure that the label prediction remains a good estimate of the label even when updating the model parameters only

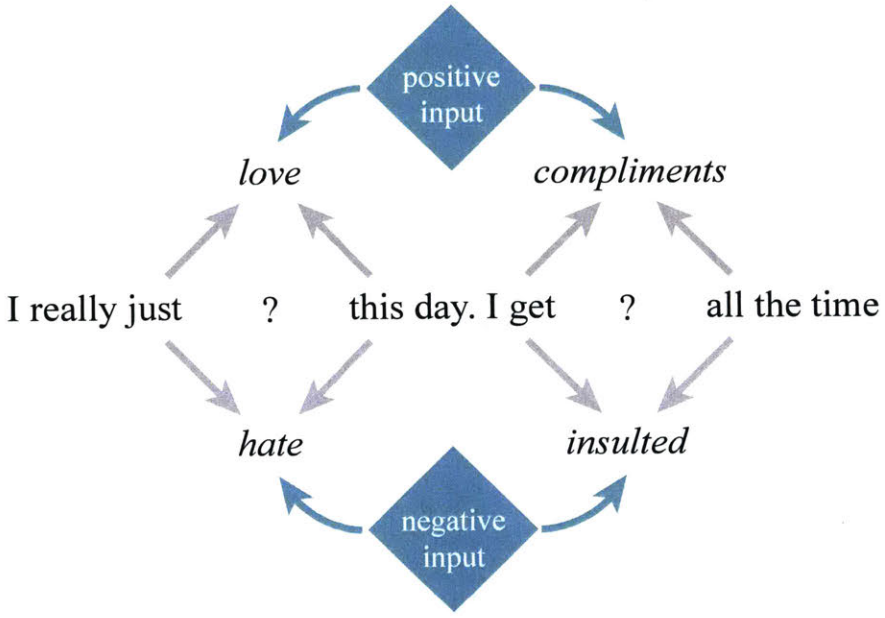

Figure 1-1: Simplified illustration of how our method works. Text-level sentiment prediction feeds into a word-level reconstruction model.

using a reconstruction loss. Our method has the desirable properties for real-world modeling and is able to obtain state-of-the-art performance.

Chapter 2

Desirable Properties for Domain Adaptation

2.1 Related Work

Structured correspondence learning (SCL) [5], stacked denoising autoencoders (SDA) [37], and marginalized SDA [7] have all been used for domain adaptation within NLP. These methods first use an unsupervised method to learn a low-dimensional representation and then train a classifier using the source domain labels on this representation. In this way, they rely on the unsupervised projection to a low-dimensional representation being able to capture the information relevant for label classification, which is a limitation that the current state-of-the-art methods do not have.

A simple approach to learn domain-invariant representations is to use a pretrained deep neural network and align the feature spaces across domains. This approach is used in DAN [26] and Deep CORAL [32]. While these methods can be effective, they require that a pretrained network is available. In the vision domain, these methods have relied on models that have been pretrained on large-scale annotated datasets such as ImageNet [11]. In NLP, it is not clear which pretrained models could be used. An approach to avoid this issue of not having a pretrained model is to create a new pretrained model by first training using $\mathcal{D}_S$. However, as we show in the Results chapter, this approach is not able to achieve state-of-the-art accuracy.

There are domain adaptation methods that have been used successfully within computer vision, but are not easily applicable to NLP. For instance, one approach in computer vision has been to use a generative adversarial network (GAN) [17] to transform the inputs, thereby improving the accuracy of the source classifier [25]. While there has been a large amount of interest in making GANs work for text [18, 39] with promising initial results, there has to our knowledge not been any papers that's successfully used these methods for improving domain adaptation for NLP tasks. Similarly, the state-of-the-art method within domain adaptation for computer vision uses a teacher-student model to achieve domain-specific input space invariances such as color and rotation invariance [14] through extensive data augmentation. As it is not clear how to adapt these methods for NLP tasks, they are not included in the comparisons in this thesis.

The domain-adversarial neural network (DANN) [1, 15] uses an adversarially trained domain classifier to align the feature spaces across domains in addition to a traditional classifier on the source domain. A gradient reversal layer (GRL) ensures that the encoder's parameters are updated such that the domain classification loss increases. While training, the model simultaneously minimizes the label prediction loss while maximizing the domain classifier's cross entropy loss. Many of the recent state-of-the-art methods build on this DANN framework. For instance, [35] proposed the Deep Domain Confusion (Deep DC) method that instead of maximizing the domain classifier's cross-entropy loss minimize the maximum mean discrepancy. Other differences between methods are the model architecture with methods like ADDA and DSN using separate encoders for each domain [36, 6]. The DSN method reconstructs the input as part of the training process, thereby learning which parts of the feature space to share across domains and which to keep private [6]. Similarly, the aspect-augmented adversarial network (AAN) was recently proposed, which stabilizes the adversarial training for NLP applications through the use of an additional word-level autoencoder loss [40].

Our method's reconstruction of words in the context of neighboring words bear resemblance to window-based tagging [8] and continuous bag-of-words (CBOW) used

to learn word vectors [28]. However, to our knowledge, no approach uses the label prediction for the text reconstruction as our method does.

2.2 Desirable properties

Due to the many ways of designing domain adaptation methods, we aim at identifying some desirable properties for these methods that can put our method in the context of previous work. We identify three such desirable properties.

First, it is desirable that the method can be trained from scratch on the dataset without any pretrained parameters because there are many important real-world tasks for which there are no pretrained classifiers available to adapt. While most of the state-of-the-art domain adaptation methods are able to do this, the DAN [26] and Deep CORAL [32] rely on aligning features of pretrained neural network models and are thus restricted in the tasks they can be applied to.

Secondly, it is desirable that the method can classify an observation without knowledge of the domain from which that observation originated. Some domain adaptation methods require such information for each observation due to e.g. the use of separate encoders for different domains [6, 36]. While this issue could potentially be solved by having a domain classifier that is used as part of the preprocessing, it is not trivial to determine the domain of individual observations in NLP. To demonstrate the difficulty of this task, we have trained LSTM models using two sets of standard NLP datasets to predict the domain (or dataset) that each example comes from. Amazon Reviews $\leftrightarrow$ Twitter has an error rate of 11.4% and AG $\leftrightarrow$ Yahoo of 43% (random: 50%). For comparison, we also train a convolutional neural network (CNN) that predicts the domains for two sets of computer vision datasets; MNIST $\leftrightarrow$ SVHN has an error rate of 1% and Amazon Pictures $\leftrightarrow$ DSLR of 3.7%. For NLP as well as for computer vision domains the error rate naturally depends a lot on the homogeneity of the specific domains, but these experiments do show the importance of being able to classify observations without domain information, especially for NLP classifiers. In addition to classifying observations from known domains, NLP models for real-world

Table 2.1: Properties of various NLP domain adaptation methods.

Method	Learn w/o pretrained params	Classify w/o domain info	Train on different label distributions
DAN [26]	no	yes	yes
Deep CORAL [32]	no	yes	yes
ADDA [36]	yes	no	no
DSN [6]	yes	no	suboptimal[1]
Deep DC [35]	yes	yes	no
DANN [1]	yes	yes	no
AAN [40]	yes	yes	no
Prediction Propagation (ours)	yes	yes	yes

applications might also need to classify new observations that cannot be clearly defined as being part of any of the existing domains used for training. It is not clear how these methods with separate layers for different domains would handle such observations.

Thirdly, it is desirable that a method is able to model domains with different label distributions as many real-world domains vary in their label distributions. For instance, Amazon reviews have a proportion of positive reviews ranging from 78% to 93% depending on the product category (see Table 2.2). All the methods that are rooted in the gradient reversal technique from DANN (i.e. ADDA, DSN, Deep DC, DANN, AAN) suffer from an inability to handle such differences in label distributions across domains. For these methods, the domain classifier exploits the label information encoded in the feature vector to predict the domain, causing the reversed gradient to remove information pertinent to the classification task. This issue is formalized in Appendix A and experimentally confirmed in the Results chapter.

Table 2.1 summarizes the properties of the state-of-the-art methods within NLP (see the Related Work section for other methods). Prediction Propagation stands out as the only method with all three desirable properties. With this thesis's focus on domain adaptation for real-world applications, the adversarial methods, DANN, Deep DC and AAN methods are relevant baselines due to their ability to learn without pretrained

[1]DSN's similarity loss is defined as the adversarial loss from either DANN or Deep DC, both of which have difficulty training on domains with different label distributions. Table 3 in the DSN paper shows that the model works without this loss, but achieves results below state of the art for 2 out of 4 benchmark datasets.

parameters and classify observations without domain information. Additionally, in case one has access to pretrained parameters, DAN and Deep CORAL are relevant baselines. Recent work has shown that DANN performs comparable or better than Deep DC [36, 27] and that Deep CORAL outperforms DAN [32], which is why we only use DANN, AAN and Deep CORAL as baseline comparisons in the Results chapter.

Table 2.2: Label distributions for the top 15 Amazon categories in the dataset after filtering to only include reviews with 30 tokens or less. The positivity percentage is the proportion of positive reviews out of all the reviews.

Category	Positivity
Books	93%
Movies And Tv	91%
Grocery And Gourmet Food	90%
Toys And Games	90%
Sports And Outdoors	90%
Automotive	90%
Home And Kitchen	89%
Tools And Home·Improvement	89%
Office Products	87%
Beauty	87%
Electronics	87%
Clothing, Shoes And Jewelry	86%
Apps For Android	85%
Health And Personal Care	85%
Cell Phones And Accessories	78%

Chapter 3

Methods

Consider the example in Figure 3-1, where our model is reconstructing the word *sweetiepie* using only the two neighboring words and the text-level label prediction. Given the two neighboring words, a simple model will be able to predict that the reconstructed word should be a noun, but without the text-level label prediction it is unclear if the reconstructed word should be a positive noun (e.g. 'sweetiepie') or a negative noun (e.g. 'moron'). Our method focuses precisely on how to use such reconstruction difficulties to learn label-related information about the target domain.

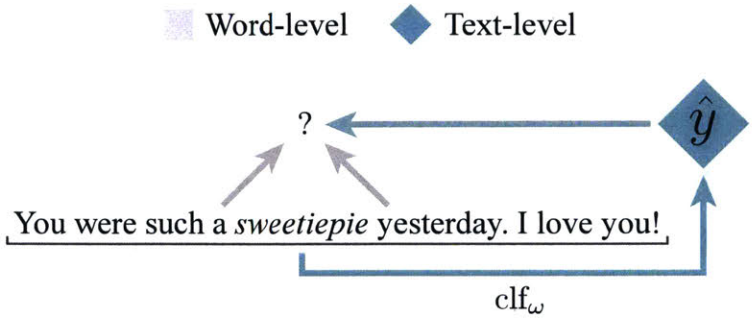

Figure 3-1: Simplified illustration of our model reconstructing each word using its two neighbors and the text-level label prediction.

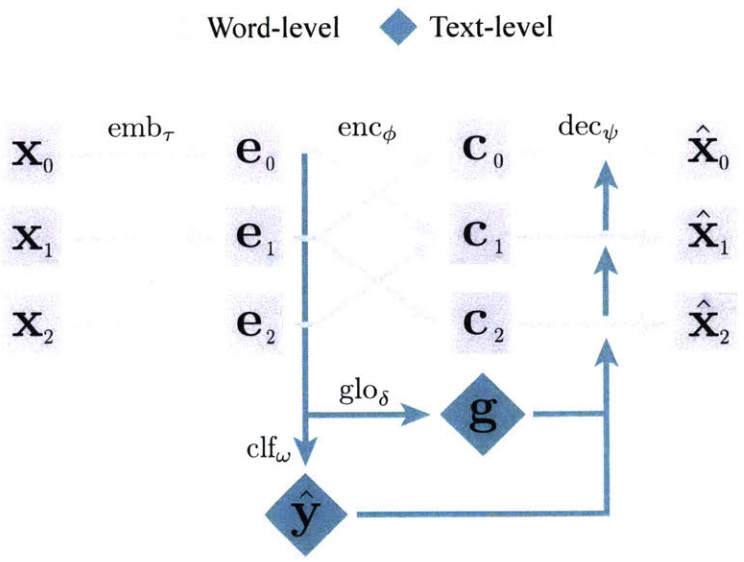

Figure 3-2: Illustration of the architecture for our method with $T = 3$ and $k = 1$. Each word is encoded based on that word's two neighboring words. The predicted distribution over the labels, $\hat{\mathbf{y}}$, is computed for the entire text, which is then concatenated to each encoded word as part of the decoding. Similarly, a global representation $\mathbf{g}$ is computed and concatenated. The decoder has thus access to word-level neighborhood encodings, the text-level prediction, and the text-level global representation.

3.1 Prediction Propagation

For the unsupervised domain adaptation task involving a source domain $\mathcal{D}_S$ and a target domain $\mathcal{D}_T$, we aim to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that obtains a low generalization error on the target domain $\mathcal{D}_T$. We have N_S labeled training samples, $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_{N_S}, y_{N_S})\}$, from $\mathcal{D}_S$ and N_T unlabeled training samples, $\{(\mathbf{x}_1), ..., (\mathbf{x}_{N_T})\}$, from $\mathcal{D}_T$. The task focus is how to make use of the N_T unlabeled training samples to teach the model about $\mathcal{D}_T$ in order to reduce the generalization error.

It is easy to obtain general semantic information about words in $\mathcal{D}_T$ using an unsupervised method like CBOW [28], but it is non-trivial to learn label-related information for each word. For instance, CBOW would likely have difficulty learning that *sweetiepie* is a positive word in the example in Figure 3-1. The goal of our *Prediction Propagation* method is thus to enrich our word embedding function $\text{emb}(\cdot)$ with label-related information about words in $\mathcal{D}_T$ and to enrich our classifier function $\text{clf}(\cdot)$ to account for patterns specific to $\mathcal{D}_T$.

We represent each text of T words as $\mathbf{x} = \{(\mathbf{x}_0), ..., (\mathbf{x}_{T-1})\}$ and assume that the probability of each word $\mathbf{x}_i$ is a function only of its k-adjacent neighbors for some $k > 0$ (i.e. it has the k-Markovian property [16]) and a text-level prediction of the probability distribution over the classes, $\hat{\mathbf{y}}$:

$$p(\mathbf{x}_i|\mathbf{x}_0, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ..., \mathbf{x}_{T-1}, \mathbf{y}) = p(\mathbf{x}_i|\mathbf{x}_{i-k}, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ..., \mathbf{x}_{i+k}, \hat{\mathbf{y}}) \tag{3.1}$$

The central idea of this thesis is to use the predicted distribution $\hat{\mathbf{y}}$ as part of a generative model, thereby learning label-related information for $\mathcal{D}_T$ by backpropagating through the classifier. To do this properly requires a special architecture and training procedure that is presented in the remainder of this chapter.

3.2 Neighborhood Encoding Architecture

In parallel to the neighborhood encoding of each word, a classifier function predicts the probability distribution over the labels from the embedding representations, i.e. $\hat{\mathbf{y}} = \text{clf}_\omega(\mathbf{e})$, where $\mathbf{e}$ contains all embedded words $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{T-1}$. This text-level prediction is used as part of the input to decoding. Similarly, we compute a global text-level representation $\mathbf{g} = \text{glo}_\delta(\mathbf{e})$ of dimensionality l. The global representation is added to help $\hat{\mathbf{y}}$ remain a good estimate of y by capturing the main global patterns that are relevant for reconstruction, thus preventing the gradients from strongly pushing $\hat{\mathbf{y}}$ towards capturing this information. The benefit of the global representation is confirmed experimentally in the Results chapter.

The decoder function is applied to each word to reconstruct that part of the input, i.e. $\hat{\mathbf{x}}_i = \text{dec}_\psi(\mathbf{c}_i, \hat{\mathbf{y}}, \mathbf{g})$. As opposed to a classic autoencoder [37], the entire input is not compressed to a single vector representation, but rather each word x_i is represented independently by its word-level neighborhood representation c_i, the text-level predicted distribution $\hat{\mathbf{y}}$, and the global representation $\mathbf{g}$. Figure 3-2 illustrates this network architecture for an input of size $T = 3$.

The key to our model's success is the connection from the text-level prediction $\hat{\mathbf{y}}$ to each word as it is being decoded. From an implementation perspective, the prediction is simply repeated with an identical prediction value being concatenated to each encoded word $\mathbf{c}_0, \mathbf{c}_1 \ldots \mathbf{c}_{T-1}$. The same goes for the global representation $\mathbf{g}$. We tie weights ψ with the weights τ for the embedding function, thereby reducing overfitting and the numbers of parameters in the model [22, 30].

Implementation

The embedding function emb_τ consists of a single embedding layer with an embedding size of $m + n + l$, where $m = 64$ and $l = 8$ are hyperparameters and n is the number of classes. The embedding weights, τ, is consequently a matrix of size $V \times (m + n)$, where V is the vocabulary size.

The classifier function clf_ω has an attention layer that averages the representations across words by weighing each word based on their attention importance score as done in [13] and a bidirectional long short-term memory [20, 31] module with 512 dimensions in each direction for detecting sequential patterns. These two layers are combined using a fully-connected layers with 1024 units and a Softmax layer that computes a probability distribution over the n classes. The global representation function glo_δ uses the same layers as the classifier function, but has a final layer of dimensionality $l = 8$ with a tangents hyperbolic activation rather than the Softmax layer.

The neighborhood encoding function enc_ϕ uses a *center-masked* convolutional layer with a filter size of 3 to encode each word using its two neighboring words. We define a center-masked convolution as a standard convolution that is applied across a feature map, where the central neuron is masked by multiplying it with zero to have no influence on the filters (see enc_ϕ in Figure 3-2). The center-masked convolutional layer has 2048 filters and is followed by a time-distributed fully-connected layer that projects each word independently down to 64 dimensions. Both layers use the ReLU activation.

The decoder function multiplies the concatenated input from enc_ϕ and clf_ω with a

matrix of size $(m+n) \times V$. A bias is added for each word and the softmax activation function is applied, thereby for each word producing a probability distribution over all the words in the vocabulary.

3.3 Multi-Phase Training

In order to learn label-related information about words and patterns occurring in $\mathcal{D}_T$ by backpropagating the gradients from a reconstruction loss through the classifier, it is crucial that $\hat{\mathbf{y}}$ remains a good estimate of y throughout training. This is non-trivial as the gradients from the reconstruction loss will attempt to move the prediction $\hat{\mathbf{y}}$ away from being a good estimate of y and instead towards something more useful for reconstructing the words. One could imagine trying to make $\hat{\mathbf{y}}$ remain a good estimate by using a classification loss based on the labeled samples from $\mathcal{D}_S$, but this would introduce a dataset-dependent hyperparameter related to the trade-off between the reconstruction loss and classification loss.

We ensure that $\hat{\mathbf{y}}$ remains a good estimate by training the model in five consecutive phases. Each phase involves minimizing one or more of the model's three losses with respect to a subset of the parameters τ, ϕ, ψ and ω. The three losses are the classification loss on the source dataset, $\mathcal{L}_{clf}$, and the two reconstructions losses, $\mathcal{L}_{source}$ and $\mathcal{L}_{target}$, making the total training loss:

$$\mathcal{L} = \lambda_{clf}\,\mathcal{L}_{clf} + \lambda_{source}\,\mathcal{L}_{source} + \lambda_{target}\,\mathcal{L}_{target} \tag{3.2}$$

While this training scheme may seem excessive, it is unlike most other domain adaptation methods free of hyperparameters that need to be tuned for each new dataset. Moreover, the main difference between phases is which parameters are trainable and which losses are used, making the training easy to implement in any modern deep learning library. The loss multipliers, λ_{clf}, λ_{source} and λ_{target} are always binary, meaning that the loss is simply on or off for any given phase. Table 3.1 details which parameters of the model are trainable in which phase and which losses are used. The first three phases teaches the model how to reconstruct the words $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}$

and predict $\hat{\mathbf{y}}$ for $\mathcal{D}_S$, whereas the last two phases make use of the trained model to learn label-related information for $\mathcal{D}_T$. In particular, the fourth phase updates the word embeddings parameters τ to learn the meanings of previously unknown words and new meanings of existing words. Similarly, the fifth phase updates ω to capture patterns specific to $\mathcal{D}_T$. As phase 1 does not yet have a trained classifier to provide $\hat{\mathbf{y}}$, the labels are provided instead. Table 4.5 in the Results chapter shows how the model is improving in each phase.

Table 3.1: Parameters being updated in each phase using which losses. Loss multipliers are binary.

Phase	Parameters	λ_{clf}	λ_{source}	λ_{target}
1	τ, ϕ, ψ, δ	0	1	0
2	ω	1	0	0
3	τ, ϕ, ψ, δ	0	1	0
4	τ, ϕ, ψ, δ	0	1	1
5	ω	0	1	1

The training is done using the Adam optimizer [24] using the default parameters and with gradient clipping of the norm set at 1. In all phases the model is trained until convergence.

Chapter 4

Results

We compare our model to DANN, AAN and Deep CORAL. For our implementation of DANN we use similar embedding and classification layers as for our own model, while using the same gradient reversal factor schedule as in the original DANN paper (see Appendix B for implementation details). For AAN we use the authors' GitHub code and default hyperparameters. The Deep CORAL model was pretrained on the source dataset and then trained jointly across both domains using the proposed loss function. For all models, we used hyperparameter optimization to ensure that any differences in performance were not due to hyperparameter choices.

4.1 Benchmark Datasets

We create three new datasets due to lack of properly sized evaluation datasets for domain adaptation in NLP (the AAN datasets have not been made available by the authors). An overview of the datasets can be seen in Table 4.1. For all datasets we use 5000 observations for the validation set and 15000 observations for the test set. To tokenize the texts and create vocabularies we use code from DeepMoji [13].

For sentiment classification we consider domain adaptation between Amazon reviews and tweets on Twitter as well as between three Amazon product categories. Previous work has also used Amazon product categories to evaluate domain adaptation [15, 7], but the review datasets used in these previous papers only had 2000

labeled source samples, making the dataset small for modern standards. We thus create our own binary classification Amazon review datasets based on previously released data [19]. Our datasets are substantially larger, making for a more realistic test case. The reviews are on an ordinal scale from 1 to 5, which we binarize by regarding reviews with scores below 3 as negative and above 3 as positive. Reviews with a score of 3 are discarded to establish a clear separation between negative and positive reviews. To make for a more challenging domain adaptation task, we use only reviews with less than 30 tokens and remove all *trivially easy* observations. These observations are defined as the ones that a bag-of-words logistic regression using only a vocabulary of the top 1000 tokens predicts with 99% confidence or more. In addition to creating balanced datasets, we also randomly sample observations from the Amazon categories to create a version of that dataset that is naturally unbalanced, varying from 78% to 93% of the reviews being positive depending on the product category (see Table 2.2 for details).

In agreement with prior work [29, 33], we define positive sentiment for tweets as those containing a positive emoji and, similarly, negative sentiment for those containing a negative emoji. To handle the noisy text on Twitter we use the tokenization scheme and vocabulary from DeepMoji [13], where words with 2 or more repeated characters are shortened to use the same token. Furtermore, all URLs, numbers, and @-mentions are replaced by special tokens.

For topic classifciation we consider domain adaptation between between Yahoo Answers and AG News [38] where the two overlapping topics in both domains, Science/Technology and Sports, are used as our two prediction classes. Both corpora are balanced. A large proportion of the observations had the website domain as part of the text, making it trivially easy to classify these observation (e.g. if the domain is www.spacescience.com). We thus remove the website domains from the texts as part of the preprocessing.

Table 4.1: Description of benchmark datasets.

Experiment	Study	Task	Classes	N_{train}	N_{val}	N_{test}	Bal	Un-bal
Twitter ↔ Amazon	[34]	Sentiment	2	200000	5000	15000	✓	
Amazon Categories	[19]	Sentiment	2	40000	5000	15000	✓	✓
AG News ↔ Yahoo	[38]	Topic	2	40000	5000	15000	✓	

4.2 Performance on Balanced and Unbalanced Datasets

To evaluate the performance on the datasets we use the area under the curve (AUC) of the receiver operating characteristic, which is suited for evaluating both balanced and unbalanced datasets. Table 4.2 shows that Prediction Propagation is the method of the three that achieves state-of-the-art performance across the most datasets. All the results are averaged across 3 runs. Additionally, we compute the standard error of the mean (SEM) of the AUC for each combination of method and dataset.

To our knowledge, DANN, AAN and Deep CORAL have only been benchmarked on domain adaptation tasks, where the distribution of labels is uniform for both the source and target domain. However, obtaining a balanced target dataset necessarily requires access to the labels, which would not be present in a real-world application of domain adaptation. We thus evaluate the methods, where the source domain is balanced (as often done for training) and the target domain is naturally unbalanced, i.e. the observations are sampled randomly from the original dataset. Table 4.3 shows that DANN, AAN and Deep CORAL perform substantially worse on these tasks with a unbalanced target domain, thereby confirming the issue discussed in the chapter on Desirable Properties for Domain Adaptation and formalized in Appendix A. Our method substantially outperforms DANN, AAN and Deep CORAL on these unbalanced datasets.

Table 4.2: Balanced tasks. AUC on the target dataset averaged across 3 runs. Higher is better.

	DANN	AAN	Deep CORAL	Our method
Amazon → Twitter	.734 ± .006	.752 ± .032	.718 ± .008	**.766 ± .004**
Twitter → Amazon	.822 ± .008	.851 ± .015	.851 ± .019	**.874 ± .016**
AG → Yahoo	.773 ± .026	**.913 ± .003**	.836 ± .005	.894 ± .004
Yahoo → AG	.948 ± .007	.965 ± .002	.969 ± .001	**.978 ± .003**
Books → Movies	**.947 ± .001**	.898 ± .004	.925 ± .006	.938 ± .001
Books → Clothing	.911 ± .004	.876 ± .005	**.922 ± .004**	.918 ± .001
Movies → Books	**.929 ± .002**	.903 ± .001	.916 ± .002	.924 ± .001
Movies → Clothing	**.918 ± .003**	.888 ± .012	.906 ± .007	.899 ± .002
Clothing → Books	.842 ± .014	.864 ± .007	.867 ± .027	**.900 ± .002**
Clothing → Movies	.896 ± .005	.846 ± .017	.889 ± .018	**.921 ± .003**

Table 4.3: Unbalanced tasks. AUC on the target dataset averaged across 3 runs. Higher is better.

	DANN	AAN	Deep CORAL	Our method
Books → Movies	.848 ± .008	.680 ± .067	.908 ± .018	**.918 ± .005**
Books → Clothing	.836 ± .003	.732 ± .061	.885 ± .008	**.901 ± .001**
Movies → Books	.816 ± .003	.614 ± .021	.902 ± .005	**.912 ± .006**
Movies → Clothing	.819 ± .004	.786 ± .023	.886 ± .019	**.898 ± .005**
Clothing → Books	.724 ± .008	.693 ± .077	.751 ± .001	**.848 ± .005**
Clothing → Movies	.743 ± .010	.738 ± .033	.862 ± .003	**.870 ± .003**

4.3 Analysis

In this section we analyze the importance of our modeling choices and provide some insight into what our model learns.

We analyze our underlying premise of the backpropagation through the classifier being crucial for the performance of our model (see Methods chapter). We run the balanced Amazon → Twitter task, where we stop the gradient from backpropagating from the decoder to the classifier. The AUC on $\mathcal{D}_T$ drops from .766 to .731, which is below the current state-of-the-art methods, thereby emphasizing the importance of backpropagating gradients through the classifier to update our encoder.

Our method does not have any training hyperparameters, but instead has two hyperparameters related to the architecture: neighborhood size k and global dimensionality l. Table 4.4 shows a grid search over hyperparameter values with the results

being averaged across 3 runs. We find that the model is relatively insensitive for changes in hyperparameter values but performs best using $k = 3$ and $l = 8$ for this task. To avoid excessive hyperparameter tuning, we use these hyperparameter values across all tasks and analysis experiments.

Table 4.4: Grid search over neighborhood size k and size of global representation l. AUC for Amazon → Twitter.

	l			
k	0	4	8	16
1	.747	.756	.741	.742
3	.753	.741	**.766**	.742
5	.753	.756	.742	.760

To understand the impact of each of our training phases we evaluate the sentiment and word reconstruction accuracies on the target domain for the balanced task Amazon → Twitter. As seen in Table 4.5 each phase improves the performance. Phase 4 substantially improves the word reconstruction accuracy, which is intuitive as this is the first phase, where the target domain reconstruction loss, $\mathcal{L}_{target}$ is used. Phase 1 and 4 take the longest with the remaining three phases accounting for less than 25% of the overall training time.

Table 4.5: Accuracies for sentiment prediction and word reconstruction on $\mathcal{D}_T$ for the balanced task Amazon → Twitter.

Phase	Sentiment	Words
1	–	$14.5 \pm .1$
2	$61.8 \pm .4$	–
3	$62.8 \pm .5$	$14.5 \pm .2$
4	$68.1 \pm .1$	$29.1 \pm .6$
5	$68.5 \pm .2$	$29.2 \pm .3$

The Methods chapter describes how our method learns label-related information about the target domain, $\mathcal{D}_T$, which is important for domain adaptation tasks. To see the degree to which our method learns more label-related information about the target domain than the competing method, DANN, we analyze their performance on a subset

of the test set, O_T, that contains observations with information only present in the target domain. To create the subset O_T, we count the word occurrences in the training set for the two domains as $C(\mathcal{D}_S)$ and $C(\mathcal{D}_T)$. We then identify all observations O_T in the test set containing a word that fulfills $C(\mathcal{D}_S) = 0$ and $C(\mathcal{D}_T) > 0$. Table 4.6 shows that the difference in performance between DANN and our method is larger for these observations O_T than for the test set in general, thereby suggesting that the overall performance gain seen in Table 4.2 and 4.3 are likely due to our method learning more label-related information about $\mathcal{D}_T$ than DANN.

Table 4.6: AUCs for subset of test set O_T that contain words not in $\mathcal{D}_S$ for Amazon $\rightarrow$ Twitter.

	PredProp	DANN	Diff
Test set	.766 ± .002	.734 ± .003	.031 ± .003
O_T	.816 ± .002	.765 ± .003	**.051 ± .005**

Chapter 5

Conclusion

An important paradigm within domain adaptation for NLP is adversarial training with methods such as DANN, ADDA, Deep DC and AAN yielding the best performance on the typical benchmark datasets. We show that these recent approaches based on adversarial training with gradient-reversal have an important limitation for real-world applications in that they cannot effectively model domains with different label distributions. This is formalized in Appendix A and shown experimentally in Table 4.3.

Motivated by the limitations of the existing domain adaptation methods, we introduced the Prediction Propagation method that does not use adversarial training. Our method uses word-level reconstruction to improve the label classifier on domains without labels. The new neighborhood encoding architecture and multi-phase training enables us to force a label prediction to remain a good estimate of the label while training without any classification loss, thereby allowing us to use the reconstruction loss to learn label-related information about $\mathcal{D}_T$. Our method is able to model domains with different label distributions and obtains state-of-the-art performance across three challenging domain adaption datasets, thereby proving its usefulness. We release our new benchmark datasets and documented code for our new method.

Appendix A

Optimal Reweighing of Data from Different Distributions

Following Ben-David et al. [2], we define a domain as a pair consisting of a domain $\mathcal{X}$ and a distribution $\mathcal{D}$, along with a labeling function $f : \mathcal{X} \to [0, 1]$, which can optionally take fractional values if the label occurs non-deterministically. Considering a hypothesis function $h : \mathcal{X} \to \{0, 1\}$ that maps any input from the space of inputs $\mathcal{X}$ to a binary classification output, we can define the error $\epsilon_D(h, f)$ as the probability that the hypothesis function disagrees with the labeling function on the domain D:

$$\epsilon_D(h, f) = \mathbb{E}_{x \sim \mathcal{D}_D}[|h(x) - f(x)|] \tag{A.1}$$

The corresponding empirical error is given by $\hat{\epsilon}_D(f, h)$. Additionally, we may also evaluate $\epsilon_\alpha(h)$, which is the empirical α-weighted error of function h for $\alpha = (\alpha_i)_{i=1}^N$, and $\sum_i \alpha_i = 1$:

$$\epsilon_\alpha(h) = \sum_{j=1}^{N} \frac{\alpha_j}{m_j} \sum_{x \in S_j} |h(x) - f_j(x)| \tag{A.2}$$

We denote the source domain by $\mathcal{D}_S$ and target domain by $\mathcal{D}_T$. We now state an important theorem in understanding differently balanced data distributions in domain adaptation. [2] Let $\mathcal{H}$ be a hypothesis class of VC dimension d. For each

$j \in \{1, 2, ...N\}$, let S_j be a labeled sample of size $\beta_j m$ generated from drawing β_j points from $\mathcal{D}_j$ and labeling them according to f_j. If $\hat{h} \in \mathcal{H}$ is the empirical minimizer of $\hat{\epsilon_\alpha}(h)$ for a fixed vector α on these samples and $h_T^* = \min_{h \in \mathcal{H}} \epsilon_T(h)$ is the target error minimizer, then for any $\delta \in (0, 1)$, with probability at least $(1 - \delta)$: $\epsilon_T(\hat{h}) \leq \epsilon_T(h_T^*) + 4\sqrt{(\sum_{j=1}^N \frac{\alpha_j^2}{\beta_j}) \frac{2d \log(2(m+1)) + \log \frac{4}{\delta}}{m}} + \sum_{j=1}^N \alpha_j (2\lambda_j + d_{\mathcal{H}\Delta\mathcal{H}}(D_j, D_T))$, where $\lambda_j = \min_{h \in \mathcal{H}} \{\epsilon_T(h) + \epsilon_j(h)\}$

It is known from [2, 3] that the excess risk $\epsilon_T(h_T^*) - \epsilon_T(\hat{h})$ by an optimal choice of α and low values of $\sum_j \alpha_j d_{\mathcal{H}\Delta\mathcal{H}}(D_j, D_T)$. Typically, to make the bound tighter, the cost of making an error in each source domain α_j is chosen by calculating a "distance" of each sample in the source dataset from the target datasets, to minimize the excess risk. This technique has been adopted by numerous instance-weighting domain adaptation techniques [21, 4, 23, 9]. As a first step, we partition the data into sub-domains, weighted by the relative number of samples β_j introduced by each subdomain. One partition for the empirical source data distribution could be that we assume it is composed of two separate distributions $\mathcal{D}_{S_0}$ and $\mathcal{D}_{S_1}$ corresponding to the classes 0 and 1 respectively, meaning that we have $\mathcal{D}_S = \mathcal{D}_{S_0} \cup \mathcal{D}_{S_1}$.

However, we can create any number of subpartitions of the source domain based on the precision we wish to approximate the target domain with, and the amount of information we have about the domains. As an extreme case, instead of considering two subdomains $\mathcal{D}_{S_0}$ and $\mathcal{D}_{S_1}$, one can also employ an *instance-based* reweighting rule, where each training point is considered to be its own domain. In the general case, we can assume k partitions, each with $(b_i m)_{i=1}^k$ elements. Since the target label distribution is not known *a priori*, we calculate the expected risk over the entire target distribution $\mathcal{D}_T$, which can be given by Theorem A. We can then estimate α_i from a chosen, optimizable "distance" measure between the source and target domains.

This method of instance-based reweighing can be applied to techniques that attempt to reduce the excess risk by minimizing an empirical approximation of $\sum_j \alpha_j d_{\mathcal{H}\Delta\mathcal{H}}(D_j, D_T)$ as well (one of which is the DANN approach by [1]). However, to optimize this exactly, the knowledge of which domain each feature representation sample belongs to is required in order to estimate a "distance" measure based on the

membership of the training sample. Since by definition, *gradient-reversal* techniques aim to create a *domain-invariant* feature representation, we can say that there is no information about the domain D in the feature representation X. In terms of mutual information between the random variables X_θ (since the features are a learnable function of the inputs, given parameters θ) and D, we can write:

$$I(D; X_{\theta^*}) = 0 \implies H(X_{\theta^*}) = H(X_{\theta^*}|D) \tag{A.3}$$

Where θ^* is the set of optimal parameters for a method employing the *gradient-reversal* approach to learn a domain-invariant representation. Consequently, the feature representation X_{θ^*} cannot be used for differentiating between domains (since the information gained on observing the feature itself is zero, by Equation A.3), or doing an instance-based reweighting (where each point is its own source domain).

Thus, the formulation penalizes each point equally (following $\alpha \sim \mathcal{U}$), which can lead to inferior performance whenever the empirical label distributions of $\mathcal{D}_S$ and $\mathcal{D}_T$ diverge significantly, since the uniformly-distributed α values will, in expectation, predict the majority class (which is not an issue when the empirical label distributions of the source and target domains are similar). Our Results chapter substantiates this insight by evaluating two domain adaptation methods using gradient reversal on domains with different label distributions.

Appendix B

DANN Baseline Implementation

We attempt to keep our DANN implementation as close as possible to our Prediction Propagation method. For the DANN model we use an embedding layer with 64 dimensions. Similarly, we use the same classification function as for Prediction Propagation except that the fully-connected layer for DANN uses the hyperbolic tangent function rather than ReLU to ensure proper gradient propagation back through the model when training adversarially. The feature vector that the DANN model attempts to make domain-invariant through adversarial training is a new 128-dimensional vector obtained by adding an additional fully-connected layer to clf_ω. The adversarial classifier is a multi-layer perceptron architecture with two fully-connected layers each with 1024 units, dropout of $p = 0.5$, and the hyperbolic tangent activation function. Lastly, another fully-connected output layer is used to predict the domain of the input observation. We use the same schedule as proposed in the DANN paper [15] to slowly increase the gradient reversal factor.

Bibliography

[1] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. In *Second Workshop on Transfer and Multi-Task Learning: Theory meets Practice (NIPS 2014)*, 2014.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[3] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[4] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM, 2007.

[5] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

[6] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[7] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *International Conference on Machine Learning (ICML)*, 2012.

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 2011.

[9] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[10] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 2006.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[12] Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M Bender. Towards linguistically generalizable nlp systems: A workshop and shared task. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[13] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[14] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for domain adaptation. In *International Conference on Learning Representations (ICLR)*, 2018.

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 2016.

[16] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice.* CRC press, 1995.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[18] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[19] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *International Conference on World Wide Web (WWW)*, 2016.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[21] Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.

[22] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations (ICLR)*, 2017.

[23] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271, 2007.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[25] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[26] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning (ICML)*, 2015.

[27] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[29] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2017.

[31] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[32] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision (ECCV) 2016 Workshops*, 2016.

[33] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Association for Computational Linguistics (ACL)*, 2014.

[34] Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, 63(1):163–173, 2012.

[35] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *International Conference in Computer Vision (ICCV)*, 2015.

[36] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*. ACM, 2008.

[38] Xiang Zhang and Yann Lecun. Text understanding from scratch. *arXiv: Learning*, 2015.

[39] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *International Conference on Machine Learning (ICML)*, 2017.

[40] Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics (TACL)*, 2017.